Source Code Audit of BIND 9
for Internet Systems Corporation

Final Report and Management Summary

2024-02-02

PUBLIC

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

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

Revision	*Date*	*Change*	*Author(s)*
1	2023-06-30	Report First Sprint	E. Sesterhenn, M. Vervier
2	2023-12-05	Final Report and Management Summary	E. Sesterhenn, M. Vervier
3	2024-02-13	Public Release	E. Sesterhenn, M. Vervier

Contents

Dashboard

Target	
Customer	Internet Systems Corporation
Name	BIND 9
Type	DNS Server
Version	9.19.13 and 9.19.17
Engagement	
Type	Security Source Code Audit
Consultants	2: Eric Sesterhenn and Markus Vervier
Engagement Effort	60 person-days, 2023-05-29 to 2023-11-30
Total issues found	8

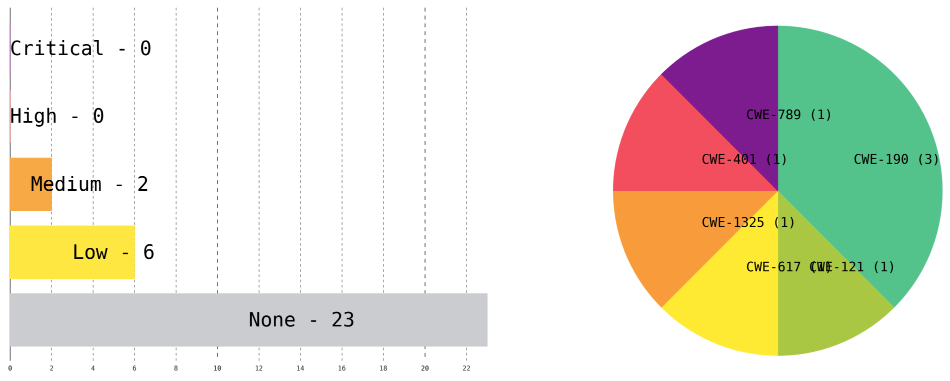


Figure 1: Issue Overview (l: Severity, r: CWE Distribution)

1 Executive Summary

From June to November 2023, X41 D-Sec GmbH performed a security source code audit against ISC BIND 9 to identify vulnerabilities and weaknesses in the DNS server, tools and libraries.

A total of eight vulnerabilities were discovered during the test by X41. None were rated as critical, none were classified as high severity, two as medium, and six as low. Additionally, 23 issues without a direct security impact were identified.

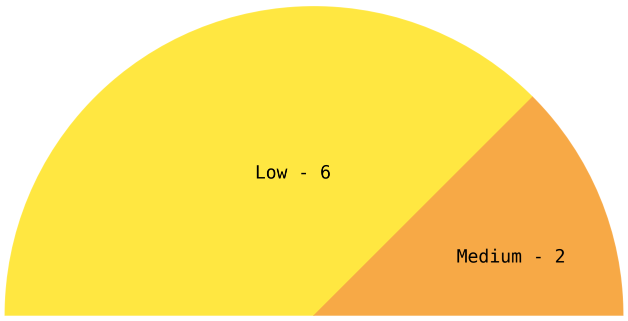


Figure 1.1: Issues and Severity

BIND 9 is an open source DNS server accompanied by libraries and tools that enable users to resolve domain names. It is a critical part of the current Internet infrastructure and therefore subject to scrutiny. Vulnerabilities could have a large impact on the worldwide Internet. Due to this, the codebase was regularly tested before with various fuzzing harnesses and inspected with various static analyzers and analysis tools before this audit and seems to be in good shape.

The test was performed by two experienced security experts between 2023-05-29 and 2023-11-

30. ISC provided the source code and a communication channel with selected developers with whom the communication during the project was flawless.

The most severe issue discovered allows an attacker to crash the named DNS server on some setups by sending maliciously formatted data to the command channel, which will exhaust the available stack memory when parsed. This can be triggered pre-authentication but installations might protect that service using network level access control and is tracked as CVE-2023-3341.

Other security relevant issues are related to the handling of integers such as truncation or overflow of length values. It is recommended to also review the informational issues that are related to hardening or issues that might become security relevant in the future.

The code was audited in a first phase to verify the most obvious attack surface and to validate the absence of commonly present vulnerability classes that are often found in complex C code. In a second phase a deeper investigation occured for more obscure bugs and logic issues.

In conclusion, the quality of the source code is far above average in comparison with projects of a similar size. While vulnerabilities were found during this audit, the code has been hardened and could serve as an example on how to develop C code with security in mind. X41 attributes the usage of safe patterns for concurrency and safe memory access to be a main contributing factor in elemination or at least reduction of the occurence of certain bug classes relaed to unsafe memory operations.

Due to the constantly changing nature of such large code bases and the time-boxed nature of security reviews, X41 recommends to perform independent review on a regular basis. This prevents the manifestation of more complex vulnerabilities and architectural flaws or otherwise unexpected behaviors.

2 Introduction

X41 reviewed the source code of the BIND 9 DNS[1] daemon and associated libraries and tools, which perform DNS queries and parse responses to resolve host names.

They are considered sensitive because the DNS is a critical part of the Internet infrastructure and several security mechanisms depend on DNS working correctly. Additionally, the libraries and tools BIND 9 provides are used in a wide range of operating systems and devices.

Attackers could try to abuse implementation-specific issues to gain RCE[2] on the server. Logic bugs might be used for DoS[3] attacks, which attackers could leverage to prevent access to a wide range of systems by disrupting name resolution.

2.1 Methodology

The review was based on a source code review against the latest development versions (9.19.13 in the first phase and 9.19.17 in the second phase) of the BIND 9 DNS daemon.

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[4].

X41 adheres to established standards for source code reviewing and penetration testing. These are in particular the *CERT Secure Coding*[5] standards and the *Study - A Penetration Testing Model*[6] of the German Federal Office for Information Security.

The workflow of source code reviews is shown in figure 2.1. In an initial, informal workshop

[1] Domain Name System
[2] Remote Code Execution
[3] Denial of Service
[4] `https://portswigger.net/burp`
[5] `https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards`
[6] `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1`

regarding the design and architecture of the application a basic threat model is created. This is used to explore the source code for interesting attack surface and code paths. These are then audited manually and with the help of tools such as static analyzers and fuzzers. The identified issues are documented and can be used in a gap analysis to highlight changes to previous audits.

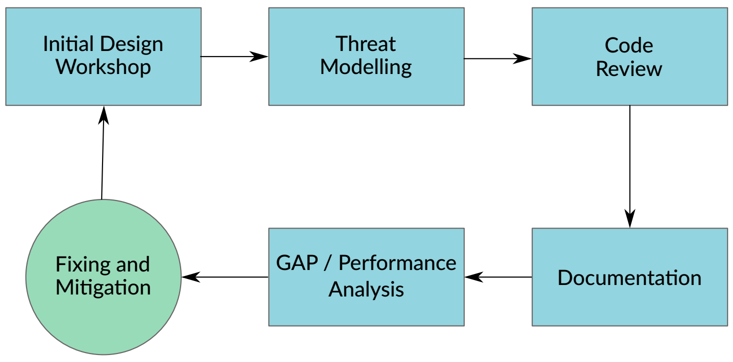


Figure 2.1: Code Review Methodology

2.2 Findings Overview

DESCRIPTION	SEVERITY	ID	GL	REF
Integer Overflow in http_calloc()	LOW	BND-CA-23-01	4120	4.1.1
Integer Overflow in mallocx()	LOW	BND-CA-23-02	4121	4.1.2
Integer Overflow in resize()	LOW	BND-CA-23-03	4122	4.1.3
Buffer Overflow in process_request()	LOW	BND-CA-23-04	4124	4.1.4
Negative Content-Length Leads to abort()	LOW	BND-CA-23-05	4125	4.1.5
Stack Exhaustion in Command Channel	MEDIUM	BND-CA-23-06	4152	4.1.6
OpenSSL Error Queue Not Emptied	LOW	BND-CA-23-07	4157	4.1.7
Unbounded Token Parsing	MEDIUM	BND-CA-23-08	4339	4.1.8
Memory Leak Due to realloc() Misuse	NONE	BND-CA-23-100	4174	4.2.1
ISC Memory API	NONE	BND-CA-23-101	n/a	4.2.2
Journal File Handling Missing Sanity Checks	NONE	BND-CA-23-102	4175	4.2.3
Unchecked malloc()	NONE	BND-CA-23-103	4077	4.2.4
Stack Exhaustion in Config Parser	NONE	BND-CA-23-104	4176	4.2.5
Connection Flags Mixup	NONE	BND-CA-23-105	4126	4.2.6
Use of Magic Numbers	NONE	BND-CA-23-106	n/a	4.2.7
NULL Pointer Dereference on Wrong API Usage	NONE	BND-CA-23-107	4177	4.2.8
Invalid Free in Low Memory Situation	NONE	BND-CA-23-108	4179	4.2.9
Possible Truncation in dns_keymgr_status()	NONE	BND-CA-23-109	4180	4.2.10
Newline and ANSI Escape Code Injection via CC	NONE	BND-CA-23-110	4181	4.2.11
Name Buffer Truncation	NONE	BND-CA-23-111	4186	4.2.12
Misaligned Structure Causes Exception	NONE	BND-CA-23-112	4187	4.2.13
Race in dns_tsigkey_find()	NONE	BND-CA-23-113	4182	4.2.14
Pointers Dereferenced before Being Checked	NONE	BND-CA-23-114	4432	4.2.15
Files Created with World Read/Write Permissions	NONE	BND-CA-23-115	4443	4.2.16
Locking Inconsistencies in Cache Implementation	NONE	BND-CA-23-116	4340	4.2.17
Endless Loop via GENERATE	NONE	BND-CA-23-117	4353	4.2.18
Endless Loop via INCLUDE	NONE	BND-CA-23-118	4357	4.2.19
Supplied Buffer Too Large	NONE	BND-CA-23-119	4433	4.2.20
Dead Code in DNSTAP Helper	NONE	BND-CA-23-120	4406	4.2.21
Unused AES Functions	NONE	BND-CA-23-121	4421	4.2.22
Usage of isc_safe_memwipe()	NONE	BND-CA-23-122	4435	4.2.23

Table 2.1: Security-Relevant Findings, GL lists the ISC GitLab IDs

2.3 Scope

The code audit was performed on versions 9.19.13[7] and 9.19.17[8] of the ISC BIND 9 DNS daemon, libraries and tools. The source code consists of around 275.000 lines of C code including tests.

Third party libraries that BIND 9 depends on, such as OpenSSL[9], LMDB[10], Libxml[11], JSON-C[12] or nghttp2[13], were not covered by this audit. Contributed code in `contrib/` was not inspected in depth.

A Mattermost chat group was created for direct communication between the developers and the testers. Additionally, issues were reported during the test in the ISC GitLab[14] as confidential issues.

2.4 Coverage

A security assessment attempts to find the most important or sometimes as many of the existing problems as possible, though it is practically never possible to rule out the possibility of additional weaknesses being found in the future.

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

The code was inspected for security issues such as integer overflows, OOB[15] reads and writes, hash table degradation, reference counter overflows, use-after-free and double free bugs. Beyond looking for common C level implementation bugs, the team investigated the codebase for bug patterns that might affect BIND 9 specifically.

The use of ***isc_buffer_*()*** and ***isc_mem_*()*** primitives were audited for misuse. Possible log file injections via ***isc_log_write()*** were checked. The DNS cookie[16] implementation was audited for logic issues. Access control and separation of trust domain issues were investigated. Race conditions and issues leading to unbound recursive calls were investigated. The conditions of the assertion functions such as ***ISC_REQUIRE()*** were audited for integer overflows and truncations. The handling and deletion of key material was inspected for common errors. Type casts were checked

[7] `https://downloads.isc.org/isc/bind9/9.19.13/`
[8] `https://downloads.isc.org/isc/bind9/9.19.17/`
[9] `https://www.openssl.org/`
[10] `http://www.lmdb.tech/doc/`
[11] `https://gitlab.gnome.org/GNOME/libxml2/-/wikis/home`
[12] `https://github.com/json-c/json-c`
[13] `https://nghttp2.org/`
[14] `https://gitlab.isc.org/isc-projects/bind9`
[15] Out-of-Bounds
[16] `https://www.ietf.org/rfc/rfc7873.txt`

for errors such as truncation or type confusions. EDNS[17] padding as well as IXFR[18] and AXFR[19] handling was inspected for logic issues.

Handling of locks and mutexes was investigated for potential race conditions, especially related to concurrent query processing and other asynchronuous operations heavily used in the code. Concurrency and access patterns such as event loops (`libuv`), lockfree access patterns, and userspace RCU[20] (`liburcu`) were audited.

X41 also investigated advanced scenarios and issues such as crashes due to incorrect memory alignment and other more uncommon problems related to compiler optimizations.

Besides manual analysis, dedicated code analysis tools were used during the process that are described in the following subsections.

2.4.1 Static Analysis

The source code was analyzed using several static analyzers such as joern.io[21], semgrep[22], weggli[23], cppcheck[24], gcc 12 analyzer[25] and the Clang Static Analyzer[26] using the built-in queries and manually created ones. Additionally, the warnings on the Intel oneAPI compilers were analyzed.

To support manual analysis and to make the analysis more efficient and repeatable, the team built CodeQL[27] queries that can find data-flows leading to vulnerabilities. In particular the following were investigated:

- Query to find correct usage of mutexes to verify manual review results (see appendix A.2.1)
- Query to find tainted length values used in ***isc_buffer_get*()*** function calls (see appendix A.2.2)
- Query to find ***printf()*** family function calls that use tainted data (see appendix A.2.3) with the intention to identify format string issues
- Query to find tainted calls of ISC memory wrapper functions (see appendix A.2.4)
- Query to find unsafe dereference related to RCU (see appendix A.2.5)
- Query to identify recursive calls of type *A()->B()->A()* (see appendix A.2.6)

[17] Extended DNS
[18] Incremental Zone Transfer
[19] Asynchronous Full Transfer Zone
[20] Read, Copy, Update
[21] `https://joern.io/`
[22] `https://semgrep.dev/`
[23] `https://github.com/weggli-rs/weggli`
[24] `https://cppcheck.sourceforge.io/`
[25] `https://gcc.gnu.org/wiki/StaticAnalyzer`
[26] `https://clang-analyzer.llvm.org/`
[27] `https://codeql.github.com`

2.4.2 Fuzz Testing

Additionally, fuzz testing using the already existing harnesses shipped with bind[28][29] and hong-fuzz[30] was performed and additional fuzz harnesses were created. These could be implemented quickly due to the already working setup and were added whenever a promising function was encountered during the audit.

This led to the creation of the following fuzzing harnesses:

- ***isc_url_parse()*** (see appendix A.1.1)
- ***phr_parse_request()*** (see appendix A.1.2)
- ***isc_regex_validate()*** (see appendix A.1.3)
- ***isc_utf8_valid()*** (see appendix A.1.4)
- ***isccc_cc_fromwire()*** (see appendix A.1.5)
- ***irs_resconf_load()*** (see appendix A.1.6)
- `dig/host` AFL++ ARGV Harness (see appendix A.1.7)

Furthermore, some speedups were implemented and memory leaks in existing harnesses removed (see appendix A.1.9).

Additionally, as an alternative strategy, template-based fuzzing using Scapy[31] was performed using a straightforward approach (see appendix A.1.8). Some stress-testing was applied to the bad-cache implemementation to identify possible errors in concurrent accesses (see appendix A.1.10).

The fuzzing did not trigger security relevant bugs, but caused parsing errors to show up in the logging output. These errors were regarded as being handled gracefully.

Suggestions for next steps in securing this scope can be found in section 2.5.

2.5 Recommended Further Tests

After investigating the most exposed and obvious attack surface, X41 recommends to perform a second iteration of the review that concentrates the efforts on more complex attack vectors and a deeper analysis of the code base.

[28] `https://gitlab.isc.org/isc-projects/bind9/-/tree/main/fuzz`
[29] `https://gitlab.isc.org/isc-projects/bind9/-/blob/main/bin/named/fuzz.c`
[30] `https://github.com/google/honggfuzz/tree/master/examples/bind`
[31] `https://scapy.net`

Further steps would be the audit of ACL[32] and limit enforcing when these are configured.

Another avenue for attacks that are hard to identify and mitigate can be compiler or operating system level bugs or unexpected behavior, which can lead to vulnerabilities that affect BIND 9. An example for such bugs was even identified in this initial review and is described in the informational finding 4.2.13.

Code that resides in `contrib/` was not inspected in-depth for this audit. X41 recommends to audit this code as well.

X41 recommends to mitigate the issues described in this report. Afterwards, CVE[33] IDs[34] should be requested and customers should be informed (e.g. via a changelog or a special note for issues with higher severity) to ensure that they can make an informed decision about upgrading or other possible mitigations.

[32] Access Control List
[33] Common Vulnerabilities and Exposures
[34] Identifiers

3 Rating Methodology for Security Vulnerabilities

Security vulnerabilities are given a purely technical rating by the testers as they are discovered during the test. Business factors and financial risks for Internet Systems Corporation are beyond the scope of a penetration test which focuses entirely on technical factors. Yet 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.

In total, five different ratings exist, which are as follows:

Severity Rating
None
Low
Medium
High
Critical

A low rating indicates that the vulnerability is either very hard for an attacker to exploit due to special circumstances, or that the impact of exploitation is limited, whereas findings with a medium rating are more likely to be exploited or have a higher impact. High and critical ratings are assigned when the testers deem the prerequisites realistic or trivial and the impact significant or very significant.

Findings with the rating 'none' are called informational findings and 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.

Common Weakness Enumeration

The CWE[1] is a set of software weaknesses that allows the categorization of vulnerabilities and weaknesses in software. If applicable, X41 provides the CWE-ID for each vulnerability that is discovered during a test.

CWE is a very powerful method to categorize a vulnerability and to give general descriptions and solution advice on recurring vulnerability types. CWE is developed by *MITRE*[2]. More information can be found on the CWE website at `https://cwe.mitre.org/`.

[1] Common Weakness Enumeration

[2] `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

The following subsections describe findings with a direct security impact that were discovered during the test.

4.1.1 BND-CA-23-01: Integer Overflow in http_calloc()

Severity:	LOW
CWE:	190 – Integer Overflow or Wraparound
GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4120
Affected Component:	lib/isc/netmgr/http.c:http_calloc()

4.1.1.1 Description

In ***http_calloc()***, two `size_t` variables are multiplied, which might lead to an overflow during the calculation. When that happens, less memory than expected by the caller is allocated, which might lead to a heap-based buffer overflow (see listing 4.1).

```c
static void *
http_calloc(size_t n, size_t sz, isc_mem_t *mctx) {
        const size_t msize = n * sz;
        void *data = isc_mem_allocate(mctx, msize);

        memset(data, 0, msize);
        return (data);
}
```

Listing 4.1: Integer Overflow in http_calloc()

Since this overflow happens in a core function with many possible callers, this is not considered informational.

4.1.1.2 Solution Advice

X41 recommends to add an overflow check for the multiplication and return `NULL` on overflow similar to the check implemented in ***isc__uv_calloc()***.

4.1.2 BND-CA-23-02: Integer Overflow in mallocx()

Severity:	LOW
CWE:	190 – Integer Overflow or Wraparound
GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4121
Affected Component:	lib/isc/jemalloc_shim.h:mallocx()

4.1.2.1 Description

In ***mallocx()***, an addition is performed to create space to store *size_info* in front of the allocated memory. This addition is not checked and could overflow for large values of *size*. When that happens, less memory than expected by the caller is allocated, which might lead to a heap-based buffer overflow (see listing 4.2).

```
static inline void *
mallocx(size_t size, int flags) {
        void *ptr = NULL;

        size_info *si = malloc(size + sizeof(*si));
        INSIST(si != NULL);

        si->size = size;
        ptr = &si[1];

        if ((flags & MALLOCX_ZERO) != 0) {
                memset(ptr, 0, size);
        }

        return (ptr);
}
```

Listing 4.2: Integer Overflow in mallocx()

Since this overflow happens in a core function, this is not considered informational.

4.1.2.2 Solution Advice

X41 recommends to add an overflow check for the addition and return *NULL* on overflow similar to the check implemented in ***isc__uv_calloc()***.

4.1.3 BND-CA-23-03: Integer Overflow in resize()

Severity:	LOW
CWE:	190 – Integer Overflow or Wraparound
GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4122
Affected Component:	lib/isc/heap.c:resize()

4.1.3.1 Description

In ***resize()*** of the heap implementation, the heap is grown, but no integer overflow checks are performed on the addition for *new_size* or the multiplication for the actual size (see listing 4.3).

```
static void
resize(isc_heap_t *heap) {
        void **new_array;
        unsigned int new_size;

        REQUIRE(VALID_HEAP(heap));

        new_size = heap->size + heap->size_increment;
        new_array = isc_mem_get(heap->mctx, new_size * sizeof(void *));
```

Listing 4.3: Integer Overflow in resize()

Since the overflow can only occur after previous resizes succeed, which occur in steps of *heap->size_increment*, the likelihood of this being exploitable is low.

4.1.3.2 Solution Advice

X41 recommends to add an overflow check for the addition and multiplication similar to the check implemented in ***isc__uv_calloc()***.

4.1.4 BND-CA-23-04: Buffer Overflow in process_request()

Severity:	LOW
CWE:	121 – Stack-based Buffer Overflow
GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4124
Affected Component:	lib/isc/httpd.c:process_request()

4.1.4.1 Description

A stack-based buffer overflow exists in `lib/isc/httpd.c`, in function ***process_request()***. When the *If-Modified-Since* HTTP[1] header is long enough, the ***memmove()*** writes out of bounds and corrupts stack data as seen in listing 4.4.

```c
} else if (name_match(header, "If-Modified-Since")) {
    char timestamp[ISC_FORMATHTTPTIMESTAMP_SIZE + 1];
    memmove(timestamp, header->value, header->value_len);
    timestamp[header->value_len] = 0;

    /* Ignore the value if it can't be parsed */
    (void)isc_time_parsehttptimestamp(
        timestamp, &httpd->if_modified_since);
}
```

Listing 4.4: Buffer Overflow in process_request()

With the default compilation settings, the overflow can overwrite the *headers* and *num_headers* values on the stack as well as canaries and return addresses.

The code in listing 4.5 can be used to trigger the overflow:

```python
import socket

HOST = "127.0.0.1"
PORT = 8080

depth = 100
payload = b'A' * depth

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    data = b'GET / HTTP/1.1\r\nHost: localhost:8080\r\n'
```

[1] HyperText Transfer Protocol

```
    data = b''.join([data, b'If-Modified-Since: ', payload, b'\r\n'])
    data = b''.join([data, b'\r\n'])
    s.connect((HOST, PORT))
    s.sendall(data)
    s.recv(10)
```

Listing 4.5: HTTP Overflow PoC

Since the statistics server should only be accessible by trusted parties and requires authentication, this issue is considered having a low impact. This is reflected in the BIND 9 documentation[2] as well:

> An issue in the statistics channel would be considered a security issue only if it could be exploited by unprivileged users circumventing the access control list. In other words, any issue in the statistics channel that could be used to access information unavailable otherwise, or to crash named, is not considered a security issue if it can be avoided through the use of a secure configuration.

4.1.4.2 Solution Advice

X41 recommends to add a size check to verify that the source buffer fits into *timestamp*.

[2] `https://bind9.readthedocs.io/en/v9_18_12/reference.html#statistics-channels-block-definition-and-usage`

4.1.5 BND-CA-23-05: Negative Content-Length Leads to abort()

Severity:	LOW
CWE:	617 – Reachable Assertion
GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4125
Affected Component:	lib/isc/httpd.c:process_request()

4.1.5.1 Description

In `lib/isc/httpd.c`, in ***process_request()*** a HTTP request is parsed and the HTTP *`Content-Length`* header evaluated (see listing 4.6). If the integer in that field is negative or large enough, the addition to *`httpd->consume`* might overflow and trigger an ***INSIST(httpd->consume != 0)*** assertion in ***prepare_response()***. This will cause an ***abort()*** and stop the process.

```c
ssize_t content_len = 0;
bool keep_alive = false;

isc_time_set(&httpd->if_modified_since, 0, 0);

for (size_t i = 0; i < num_headers; i++) {
    struct phr_header *header = &headers[i];

    if (name_match(header, "Content-Length")) {
        char *endptr;
        content_len = (size_t)strtoul(header->value, &endptr,
                                 10);
        /* Consistency check, if we consumed all numbers */
        if ((header->value + header->value_len) != endptr) {
            return (ISC_R_RANGE);
        }
...
/* Consume the request's data, which we do not use. */
httpd->consume += content_len;
...
```

Listing 4.6: Negative Content-Length Leads to abort()

This should only be possible when the attacker is also able to send enough (*`content_len`* bytes) data to the service.

Since the statistics server should only be accessible by trusted parties and requires authentication this issue is considered having a low impact.

4.1.5.2 Solution Advice

X41 recommends to add a size sanity check or catch the overflow during the addition.

4.1.6 BND-CA-23-06: Stack Exhaustion in Command Channel

Severity:	MEDIUM
CWE:	1325 – Improperly Controlled Sequential Memory Allocation
GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4152
Affected Component:	lib/isccc:value_fromwire()

4.1.6.1 Description

The command channel library libisccc parses the full TCP[3] packet before performing the actual authentication. The packet is structured into binary data, tables and lists. By structuring the packet maliciously, repeated recursive calls to ***value_fromwire()*** and ***table_fromwire()*** can be triggered. These functions are shown in listing 4.7.

```
static isc_result_t
value_fromwire(isccc_region_t *source, isccc_sexpr_t **valuep) {
        unsigned int msgtype;
        uint32_t len;
        isccc_sexpr_t *value;
        isccc_region_t active;
        isc_result_t result;

        if (REGION_SIZE(*source) < 1 + 4) {
                return (ISC_R_UNEXPECTEDEND);
        }
        GET8(msgtype, source->rstart);
        GET32(len, source->rstart);
        if (REGION_SIZE(*source) < len) {
                return (ISC_R_UNEXPECTEDEND);
        }
        active.rstart = source->rstart;
        active.rend = active.rstart + len;
        source->rstart = active.rend;
        if (msgtype == ISCCC_CCMSGTYPE_BINARYDATA) {
                value = isccc_sexpr_frombinary(&active);
                if (value != NULL) {
                        *valuep = value;
                        result = ISC_R_SUCCESS;
                } else {
                        result = ISC_R_NOMEMORY;
                }
        } else if (msgtype == ISCCC_CCMSGTYPE_TABLE) {
                result = table_fromwire(&active, NULL, 0, valuep);
```

[3] Transmission Control Protocol

```
        } else if (msgtype == ISCCC_CCMSGTYPE_LIST) {
                result = list_fromwire(&active, valuep);
        } else {
                result = ISCCC_R_SYNTAX;
        }

        return (result);
}

static isc_result_t
table_fromwire(isccc_region_t *source, isccc_region_t *secret,
               uint32_t algorithm, isccc_sexpr_t **alistp) {
        char key[256];
        uint32_t len;
        isc_result_t result;
        isccc_sexpr_t *alist, *value;
        bool first_tag;
        unsigned char *checksum_rstart;

        REQUIRE(alistp != NULL && *alistp == NULL);

        checksum_rstart = NULL;
        first_tag = true;
        alist = isccc_alist_create();
        if (alist == NULL) {
                return (ISC_R_NOMEMORY);
        }

        while (!REGION_EMPTY(*source)) {
                GET8(len, source->rstart);
                if (REGION_SIZE(*source) < len) {
                        result = ISC_R_UNEXPECTEDEND;
                        goto bad;
                }
                GET_MEM(key, len, source->rstart);
                key[len] = '\0'; /* Ensure NUL termination. */
                value = NULL;
                result = value_fromwire(source, &value);
                if (result != ISC_R_SUCCESS) {
                        goto bad;
                }
...
```

Listing 4.7: Stack Exhaustion in CC

On the test machine, each iteration between two calls to ***table_fromwire()*** required 432 bytes of stack memory. If enough calls can be triggered, this might lead to exhaustion of the available stack memory and cause a segmentation fault. The amount of iterative calls is limited by the parameter to ***isccc_ccmsg_setmaxsize()*** which is *32768* for BIND `named`. This is not enough to exhaust the 8MB of process stack usually configured on Linux-based systems. It can be triggered on Microsoft Windows systems where the stack size is 1MB by default. Systems where `ulimit`[4] is used to restrict stack usage are affected as well. BIND 9.18 on FreeBSD is also affected by this.

It was tested with the debug build of the latest[5] BIND 9 version that supports MS Windows, the error is shown in listing 4.8:

```
Unhandled exception at 0x00007FFCB43676C0 (libisccc.dll) in named.exe: 0xC00000FD: Stack overflow
↪ (parameters: 0x0000000000000001, 0x000000F149503F10).
```

Listing 4.8: Stack Exhaustion on Microsoft Windows

The issue can be triggered by sending the maliciously formatted data to the `rdnc` port as shown in listing 4.9:

```python
import socket

HOST = "127.0.0.1"
PORT = 953

depth = 4500
# should not be more than isccc_ccmsg_setmaxsize(&conn->ccmsg, 32768);
total_len = 10 + (depth * 7) - 6

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    data = b''.join([
        total_len.to_bytes(4, 'big'),   # <total lenght>
        b'\x00\x00\x00\x01',            # <version>
        b'\x01\x41',                    # <size><name>
    ])

    for i in range(depth, 0, -1):
        l = (i - 1) * 7
        t = b''.join([
                b'\x02',                        # ISCCC_CCMSGTYPE_TABLE
                l.to_bytes(4, 'big'),           # <size>
            b'\x01\x41',            # <size><name>
                ])
        data = b''.join([data, t])

```

[4] https://man7.org/linux/man-pages/man3/ulimit.3.html
[5] https://downloads.isc.org/isc/bind9/cur/9.16/BIND9.16.41.debug.x64.zip

```
    s.connect((HOST, PORT))
    s.sendall(data)
```

Listing 4.9: Stack Exhaustion in CC PoC

Usually TCP port 953 should not be reachable by untrusted parties due to firewalling and the BIND 9 configuration, but since authentication is performed as well, this might not always be the case.

4.1.6.2 Solution Advice

X41 recommends to further reduce the size passed to ***isccc_ccmsg_setmaxsize()*** or limit the number of recursive iterations. Another way to reduce the impact of this would be to dynamically allocate the key buffer on the heap instead of the stack.

4.1.7 BND-CA-23-07: OpenSSL Error Queue Not Emptied

Severity:	LOW
CWE:	401 – Improper Release of Memory Before Removing Last Reference ('Memory Leak')
GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4157
Affected Component:	lib/dns/rdata.c:check_private()

4.1.7.1 Description

OpenSSL 3 uses a queue to track errors[6]. The memory for errors in this queue is allocated via ***isc__tls_malloc_ex()*** and freed in three cases:

1. When the queue is emptied via ***ERR_get_error_all()*** and similar[7], for example via a call to ***dst__openssl_toresult()***
2. Via ***OPENSSL_cleanup()*** on a clean shutdown during library unload or when called from ***isc__tls_shutdown()***
3. ***isc__tls_shutdown()*** when the memory pool is destroyed via ***isc_mem_destroy()***

Some parts of the code clean that queue directly after an error happened (see listing 4.10).

```
status = EVP_PKEY_fromdata(
        ctx, retpkey, private ? EVP_PKEY_KEYPAIR : EVP_PKEY_PUBLIC_KEY,
        params);
if (status != 1) {
        DST_RET(dst__openssl_toresult2("EVP_PKEY_fromdata",
                                       DST_R_OPENSSLFAILURE));
}
ret = ISC_R_SUCCESS;
```

Listing 4.10: OpenSSL Error Queue Handling

When tracing the binary via `valgrind`[8] these leaks do not show up due to the cleanup on library unload.

[6] https://subscription.packtpub.com/book/web-development/9781800560345/9/ch09lvl1sec65/understanding-the-openssl-error-queue
[7] https://www.openssl.org/docs/man3.1/man3/ERR_get_error.html
[8] https://valgrind.org/

For the following trace (see listing 4.11) no cleanup seems to occur and therefore there should be a memory leak.

```
==137103== 16 bytes in 1 blocks are still reachable in loss record 33 of 718
==137103==    at 0x48407B4: malloc (vg_replace_malloc.c:381)
==137103==    by 0x4B4A6A9: mallocx (jemalloc_shim.h:65)
==137103==    by 0x4B4A858: mem_get (mem.c:304)
==137103==    by 0x4B4BF3A: isc__mem_allocate (mem.c:785)
==137103==    by 0x4B5FB11: isc__tls_malloc_ex (tls.c:129)
==137103==    by 0x54F1AF5: CRYPTO_strdup (in /usr/lib/x86_64-linux-gnu/libcrypto.so.3)
==137103==    by 0x54A8167: ERR_set_debug (in /usr/lib/x86_64-linux-gnu/libcrypto.so.3)
==137103==    by 0x53AFACC: ASN1_get_object (in /usr/lib/x86_64-linux-gnu/libcrypto.so.3)
==137103==    by 0x53A9040: d2i_ASN1_OBJECT (in /usr/lib/x86_64-linux-gnu/libcrypto.so.3)
==137103==    by 0x4950D55: check_private (rdata.c:624)
==137103==    by 0x49865A5: fromwire_rrsig (rrsig_46.c:345)
==137103==    by 0x49ADF93: dns_rdata_fromwire (rdata.c:824)
```

Listing 4.11: Allocation Due to ERR_set_debug()

A patch was applied to some fuzzing harnesses to be able to catch these errors during fuzz testing (see listing 4.12).

```
--- dns_message_parse.c.orig    2023-06-21 06:21:00.172793050 +0200
+++ dns_message_parse.c    2023-06-21 20:19:01.586207696 +0200
@@ -25,6 +25,10 @@

 #include <dns/message.h>

+#include <assert.h>
+#include <openssl/err.h>
+
+
 #include "fuzz.h"

 bool debug = false;
@@ -180,6 +184,19 @@ cleanup:
     if (message != NULL) {
         dns_message_detach(&message);
     }
+        if (ERR_peek_error() != 0) {
+        const char *file = NULL;
+        int line = 0;
+        const char *func = NULL;
+        const char *data2 = NULL;
+        int flags = 0;
+        long x = 1;
+        while (x != 0) {
```

```
+             x= ERR_get_error_all(&file, &line, &func, &data2, &flags);
+             printf("%s:%d:%s -> %s\n", file, line, func, data2);
+         }

+         abort();
+     }
      return (0);
 }
```

Listing 4.12: Enable Fuzzer to Catch OpenSSL Error Queue Leaks

Another place where similar code could be added to help debugging would be ***isc__tls_shutdown()*** in `lib/isc/tls.c` before the appropriate cleanup functions are called.

4.1.7.2 Solution Advice

X41 recommends to always empty the OpenSSL error queue after OpenSSL errors occurred by calling ***ERR_clear_error()***[9] or similar.

[9] `https://www.openssl.org/docs/manmaster/man3/ERR_clear_error.html`

4.1.8 BND-CA-23-08: Unbounded Token Parsing

Severity:	MEDIUM
CWE:	789 – Uncontrolled Memory Allocation
GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4339
Affected Component:	lib/isc/lex.c:pushandgrow()

4.1.8.1 Description

During the code review it was found that the lexer code in file `lib/isc/lex.c` is affected by crashes and infinite loops when parsing very long strings.

In particular the function ***pushandgrow()***, which is used to enlarge buffers during parsing, can lead to a failed allocation on 32-bit architectures and a subsequent ***abort()*** call:

```
static isc_result_t
pushandgrow(isc_lex_t *lex, inputsource *source, int c) {
    if (isc_buffer_availablelength(source->pushback) == 0) {
        isc_buffer_t *tbuf = NULL;
        unsigned int oldlen;
        isc_region_t used;
        isc_result_t result;

        oldlen = isc_buffer_length(source->pushback);
        isc_buffer_allocate(lex->mctx, &tbuf, oldlen * 2); // MARK integer overflow
        isc_buffer_usedregion(source->pushback, &used);
        result = isc_buffer_copyregion(tbuf, &used);
        INSIST(result == ISC_R_SUCCESS);
        tbuf->current = source->pushback->current;
        isc_buffer_free(&source->pushback);
        source->pushback = tbuf;
    }
    isc_buffer_putuint8(source->pushback, (uint8_t)c);
    return (ISC_R_SUCCESS);
}
```

Listing 4.13: Unbounded Buffer Size Increase

When parsing a zone file with a very long token, the lexer will enter the function ***pushandgrow()*** repeatedly and double the buffer allocation each time the buffer space is exhausted. This quickly leads to a failed allocation on 32-bit architectures, since the maximum single allocation size is at most 4GB due to the limited address space. In practice, it is even much less for most allocators.

An example is shown in the following listing:

```
$ ./bin/dnssec/dnssec-signzone-gdb db.long
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
        <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from /home/user/src-new/bind9/bin/dnssec/.libs/dnssec-signzone...
gdb-peda$ r
Starting program: /home/user/src-new/bind9/bin/dnssec/.libs/dnssec-signzone db.long
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
jemalloc_shim.h:68: INSIST(ptr != ((void *)0)) failed, back trace
/home/user/src-new/bind9/lib/isc/.libs/libisc-9.19.18-dev.so(+0x27859)[0xf7f76859]
/home/user/src-new/bind9/lib/isc/.libs/libisc-9.19.18-dev.so(isc_assertion_failed+0x26)[0xf7f7678
↪ a]
/home/user/src-new/bind9/lib/isc/.libs/libisc-9.19.18-dev.so(+0x4222a)[0xf7f9122a]
/home/user/src-new/bind9/lib/isc/.libs/libisc-9.19.18-dev.so(+0x4238b)[0xf7f9138b]
/home/user/src-new/bind9/lib/isc/.libs/libisc-9.19.18-dev.so(isc__mem_get+0x58)[0xf7f923d0]
/home/user/src-new/bind9/lib/isc/.libs/libisc-9.19.18-dev.so(+0x390f5)[0xf7f880f5]
/home/user/src-new/bind9/lib/isc/.libs/libisc-9.19.18-dev.so(+0x39fb8)[0xf7f88fb8]
/home/user/src-new/bind9/lib/isc/.libs/libisc-9.19.18-dev.so(isc_lex_gettoken+0x2e0)[0xf7f89334]
/home/user/src-new/bind9/lib/dns/.libs/libdns-9.19.18-dev.so(+0x771c9)[0xf7c771c9]
/home/user/src-new/bind9/lib/dns/.libs/libdns-9.19.18-dev.so(+0x78fb7)[0xf7c78fb7]
/home/user/src-new/bind9/lib/dns/.libs/libdns-9.19.18-dev.so(dns_master_loadfile+0x8b)[0xf7c8066c]
/home/user/src-new/bind9/lib/dns/.libs/libdns-9.19.18-dev.so(dns_db_load+0xbd)[0xf7c403e8]
/home/user/src-new/bind9/bin/dnssec/.libs/dnssec-signzone(+0xc13c)[0x5656113c]
/home/user/src-new/bind9/bin/dnssec/.libs/dnssec-signzone(main+0xfaa)[0x565646a6]
/lib/i386-linux-gnu/libc.so.6(__libc_start_main+0x106)[0xf7a32e46]
/home/user/src-new/bind9/bin/dnssec/.libs/dnssec-signzone(_start+0x31)[0x56559e51]

Program received signal SIGABRT, Aborted.
[----------------------------------registers-----------------------------------]
EAX: 0x0
EBX: 0x2
ECX: 0xffffbfdc --> 0x0
EDX: 0x0
ESI: 0x8
EDI: 0x0
EBP: 0xffffbfdc --> 0x0
```

```
ESP: 0xffffbfc0 --> 0xffffbfdc --> 0x0
EIP: 0xf7fd0559 (<__kernel_vsyscall+9>: pop    ebp)
EFLAGS: 0x200246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0xf7fd0553 <__kernel_vsyscall+3>:    mov    ebp,esp
   0xf7fd0555 <__kernel_vsyscall+5>:    sysenter
   0xf7fd0557 <__kernel_vsyscall+7>:    int    0x80
=> 0xf7fd0559 <__kernel_vsyscall+9>:    pop    ebp
   0xf7fd055a <__kernel_vsyscall+10>:   pop    edx
   0xf7fd055b <__kernel_vsyscall+11>:   pop    ecx
   0xf7fd055c <__kernel_vsyscall+12>:   ret
   0xf7fd055d:  nop
[------------------------------------stack-------------------------------------]
0000| 0xffffbfc0 --> 0xffffbfdc --> 0x0
0004| 0xffffbfc4 --> 0x0
0008| 0xffffbfc8 --> 0xffffbfdc --> 0x0
0012| 0xffffbfcc --> 0xf7a48e02 (<__GI_raise+194>:      mov    eax,DWORD PTR [esp+0x10c])
0016| 0xffffbfd0 --> 0xf7fe26e9 (<_dl_fixup+9>: add    ebx,0x1a917)
0020| 0xffffbfd4 --> 0xf7fc8000 --> 0x78e18
0024| 0xffffbfd8 --> 0xf7fad5a8 (", back trace")
0028| 0xffffbfdc --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGABRT
0xf7fd0559 in __kernel_vsyscall ()
gdb-peda$ bt
#0  0xf7fd0559 in __kernel_vsyscall ()
#1  0xf7a48e02 in __libc_signal_restore_set (set=0xffffbfdc) at
↪   ../sysdeps/unix/sysv/linux/internal-signals.h:86
#2  __GI_raise (sig=0x6) at ../sysdeps/unix/sysv/linux/raise.c:48
#3  0xf7a31306 in __GI_abort () at abort.c:79
#4  0xf7f76792 in isc_assertion_failed (file=0xf7fb2917 "jemalloc_shim.h", line=0x44,
↪   type=isc_assertiontype_insist,
    cond=0xf7fb2904 "ptr != ((void *)0)") at assertions.c:49
#5  0xf7f9122a in mallocx (size=0x8000002c, flags=0x0) at jemalloc_shim.h:68
#6  0xf7f9138b in mem_get (ctx=0x56583b60, size=0x8000002c, flags=0x0) at mem.c:305
#7  0xf7f923d0 in isc__mem_get (ctx=0x56583b60, size=0x8000002c, flags=0x0) at mem.c:744
#8  0xf7f880f5 in isc_buffer_allocate (mctx=0x56583b60, dbufp=0xffffc334, length=0x80000000) at
↪   ./include/isc/buffer.h:1085
#9  0xf7f88fb8 in pushandgrow (lex=0x5658bf30, source=0x565832e0, c=0x41) at lex.c:327
#10 0xf7f89334 in isc_lex_gettoken (lex=0x5658bf30, options=0x137, tokenp=0xffffcd44) at lex.c:451
#11 0xf7c771c9 in gettoken (lex=0x5658bf30, options=0x137, token=0xffffcd44, eol=0x1,
↪   callbacks=0xffffce84) at master.c:341
#12 0xf7c78fb7 in load_text (lctx=0x5658b520) at master.c:1090
#13 0xf7c8066c in dns_master_loadfile (master_file=0xffffd549 "db.long", top=0x56582c60,
↪   origin=0x56582c60, zclass=0x1,
    options=0x0, resign=0x0, callbacks=0xffffce84, include_cb=0x0, include_arg=0x0,
    ↪   mctx=0x56583b60, format=dns_masterformat_text,
    maxttl=0x0) at master.c:2637
#14 0xf7c403e8 in dns_db_load (db=0x56582c50, filename=0xffffd549 "db.long",
↪   format=dns_masterformat_text, options=0x0) at db.c:316
```

```
#15 0x5656113c in loadzone (file=0xffffd549 "db.long", origin=0xffffd549 "db.long", rdclass=0x1,
↪  db=0x5656d5cc <gdb>)
    at dnssec-signzone.c:2579
#16 0x565646a6 in main (argc=0x0, argv=0xffffd38c) at dnssec-signzone.c:3862
#17 0xf7a32e46 in __libc_start_main (main=0x565636fc <main>, argc=0x2, argv=0xffffd384,
↪  init=0x565669b0 <__libc_csu_init>,
    fini=0x56566a10 <__libc_csu_fini>, rtld_fini=0xf7fe3230 <_dl_fini>, stack_end=0xffffd37c) at
    ↪  ../csu/libc-start.c:308
#18 0x56559e51 in _start ()
gdb-peda$
```

Listing 4.14: Failed Allocation on 32-Bit Architectures

As a proof of concept, a zone file triggering this behavior can be created using the command:

- `perl -e ’print "AA"x0x80000000’ > db.long`

The tool `bin/dnssec/ddnssec-signzone` can then be used to trigger the crash. It shares the same code path with `named` and attackers could try to provide malicious zone files to it.

On 64-bit architectures, the parsing code will enter an infinite loop due to unsigned integer truncation.

Arithmetic overflows caused by large amounts of data are notoriously hard to find using general purpose dynamic analysis methods such as fuzzing.

4.1.8.2 Solution Advice

X41 recommends to limit the maximum token size to a sane amount. Furthermore, it is recommended to replace all allocation size calculations in the lexer by using the new ***isc_mem_callocate()*** function which can detect arithmetic overflows. Besides the examples shown above, there are also other parts of the lexer code that could be prone to similar bugs such as the function shown in listing 4.15.

```
static isc_result_t
grow_data(isc_lex_t *lex, size_t *remainingp, char **currp, char **prevp) {
    char *tmp;

    tmp = isc_mem_get(lex->mctx, lex->max_token * 2 + 1);
    memmove(tmp, lex->data, lex->max_token + 1);
    *currp = tmp + (*currp - lex->data);
    if (*prevp != NULL) {
        *prevp = tmp + (*prevp - lex->data);
    }
    isc_mem_put(lex->mctx, lex->data, lex->max_token + 1);
    lex->data = tmp;
    *remainingp += lex->max_token;
    lex->max_token *= 2;
    return (ISC_R_SUCCESS);
}
```

Listing 4.15: Allocation Sizes Calculated Using Unsafe Integer Arithmetic

It is recommended to refactor the parsing code and potentially use parser generators to auto generate the parsing code.

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 BND-CA-23-100: Memory Leak Due to realloc() Misuse

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4174
Affected Component:	contrib/dlz/modules/bdbhpt/dlz_bdbhpt_dynamic.c

4.2.1.1 Description

In ***dlz_allnodes()*** and ***dlz_lookup()***, buffers are resized using ***realloc()***. When the system is low on memory, ***realloc()*** will return *NULL* but will not free the original memory. The pointer *tmp* is overwritten by the return value of ***realloc()*** (as seen in listing 4.16) and no other reference to that buffer exists.

Therefore the memory allocated before is no longer referenced and leaks, which increases the memory pressure on the system even further. The relevant code is shown in the following listing 4.16:

```
tmp = realloc(tmp, dns_data.size + 1);
if (tmp == NULL) {
    goto allnodes_cleanup;
}
```

Listing 4.16: realloc() Misuse

Since this issue is in the contributed and not the main code, it is considered an informational issue. Code in `contrib/` is not actively maintained by Internet Systems Corporation.

4.2.1.2 Solution Advice

X41 recommends to use a temporary pointer to obtain the return value of ***realloc()***.

4.2.2 BND-CA-23-101: ISC Memory API

GitLab Issue:	n/a
Affected Component:	lib/isc/include/isc/mem.h

4.2.2.1 Description

The ISC memory API[10] functions ***isc_mem_get()*** and ***isc_mem_allocate()*** are used to allocate arrays. In several of these cases, the multiplications for these happen without an integer overflow check:

```
lib/isc/netmgr/netmgr.c:        isc_mem_get(mctx, netmgr->nloops * sizeof(netmgr->workers[0]));
lib/isc/lex.c:    tmp = isc_mem_get(lex->mctx, lex->max_token * 2 + 1);
lib/isc/symtab.c:    symtab->table = isc_mem_get(mctx, size * sizeof(eltlist_t));
lib/isc/symtab.c:    newtable = isc_mem_get(symtab->mctx, newsize * sizeof(eltlist_t));
lib/isc/tls.c:    locks = isc_mem_getx(isc__tls_mctx, nlocks * sizeof(locks[0]),
lib/isc/commandline.c:        *argvp = isc_mem_get(mctx, n * sizeof(char *));
lib/dns/ssu.c:        rule->types = isc_mem_get(mctx, ntypes * sizeof(*rule->types));
lib/dns/remote.c:        remote->ok = isc_mem_get(mctx, count * sizeof(bool));
lib/dns/rdataset.c:            in = isc_mem_get(cctx->mctx, count * sizeof(*in));
lib/dns/rdataset.c:            out = isc_mem_get(cctx->mctx, count * sizeof(*out));
lib/dns/diff.c:    v = isc_mem_get(diff->mctx, length * sizeof(dns_difftuple_t *));
lib/dns/rdataslab.c:    x = isc_mem_get(mctx, nalloc * sizeof(struct xrdata));
lib/dns/rdataslab.c:    offsettable = isc_mem_getx(mctx, nalloc * sizeof(unsigned int),
lib/dns/rdataslab.c:    offsettable = isc_mem_getx(mctx, mcount * sizeof(unsigned int),
lib/dns/ipkeylist.c:    addrs = isc_mem_get(mctx, n * sizeof(isc_sockaddr_t));
lib/dns/ipkeylist.c:    sources = isc_mem_get(mctx, n * sizeof(isc_sockaddr_t));
lib/dns/ipkeylist.c:    keys = isc_mem_get(mctx, n * sizeof(dns_name_t *));
lib/dns/ipkeylist.c:    tlss = isc_mem_get(mctx, n * sizeof(dns_name_t *));
lib/dns/ipkeylist.c:    labels = isc_mem_get(mctx, n * sizeof(dns_name_t *));
lib/dns/dispatch.c:        v4ports = isc_mem_get(mgr->mctx, sizeof(in_port_t) * nv4ports);
lib/dns/dispatch.c:        v6ports = isc_mem_get(mgr->mctx, sizeof(in_port_t) * nv6ports);
lib/dns/dispatch.c:    dset->dispatches = isc_mem_get(mctx, sizeof(dns_dispatch_t *) * n);
lib/dns/acl.c:    acl->elements = isc_mem_getx(mctx, n * sizeof(acl->elements[0]),
lib/dns/badcache.c:    bc->table = isc_mem_getx(bc->mctx, sizeof(bc->table[0]) * size,
lib/dns/badcache.c:    bc->tlocks = isc_mem_getx(bc->mctx, sizeof(bc->tlocks[0]) * size,
lib/dns/badcache.c:    newtable = isc_mem_getx(bc->mctx, sizeof(dns_bcentry_t *) * newsize,
lib/dns/badcache.c:    newlocks = isc_mem_get(bc->mctx, sizeof(isc_mutex_t) * newsize);
lib/dns/master.c:    newlist = isc_mem_get(mctx, new_len * sizeof(*newlist));
lib/dns/master.c:    newlist = isc_mem_get(mctx, new_len * sizeof(*newlist));
lib/dns/dnssec.c:    data = isc_mem_get(mctx, n * sizeof(dns_rdata_t));
lib/dns/zoneverify.c:    dstkeys = isc_mem_get(vctx->mctx, sizeof(*dstkeys) * count);
lib/dns/zone.c:    argv = isc_mem_get(zone->mctx, dbargc * sizeof(*argv));
lib/ns/update.c:        rules = isc_mem_getx(mctx, sizeof(*rules) * ruleslen,
lib/ns/query.c:    aaaaok = isc_mem_get(client->manager->mctx, sizeof(bool) * count);
bin/nsupdate/nsupdate.c:        servers = isc_mem_get(gmctx, ns_alloc * sizeof(isc_sockaddr_t));
```

[10] Application Programming Interface

```
bin/nsupdate/nsupdate.c:        servers = isc_mem_get(gmctx, ns_alloc * sizeof(isc_sockaddr_t));
bin/nsupdate/nsupdate.c:    servers = isc_mem_getx(gmctx, ns_alloc * sizeof(isc_sockaddr_t),
bin/dnssec/dnssec-cds.c:    keytable = isc_mem_getx(mctx, sizeof(keytable[0]) * nkey, ⌋
↪  ISC_MEM_ZERO);
bin/dnssec/dnssec-cds.c:    algo = isc_mem_getx(mctx, nkey * sizeof(algo[0]), ISC_MEM_ZERO);
bin/dnssec/dnssec-cds.c:    arrdata = isc_mem_get(mctx, n * sizeof(dns_rdata_t));
bin/dnssec/dnssec-cds.c:    ds = isc_mem_get(mctx, n * sizeof(dns_rdata_ds_t));
bin/dnssec/dnssec-signzone.c:    wassignedby = isc_mem_get(mctx, arraysize * sizeof(bool));
bin/dnssec/dnssec-signzone.c:    nowsignedby = isc_mem_get(mctx, arraysize * sizeof(bool));
bin/named/zoneconf.c:            types = isc_mem_get(mctx, n * sizeof(*types));
```

Listing 4.17: Example isc_mem_get() Calls

A full list (similar to the one shown in listing 4.17) can be found with the weggli query shown in listing 4.18:

```
weggli --unique -R 'a!=^[A-Z_]+$' 'isc_mem_get($a * _);' ~/bind-9.19.13
```

Listing 4.18: weggli Query to Identify Multiplication in Allocation

This informational issue was not reported as a separate issue on GitLab but discussed in another issue[11] and addressed in merge request 8007[12].

4.2.2.2 Solution Advice

X41 recommends to improve the API by adding functions similar to ***calloc()*** that handle the multiplication and overflow check in a generic way.

[11] https://gitlab.isc.org/isc-projects/bind9/-/issues/4120#note_379884
[12] https://gitlab.isc.org/isc-projects/bind9/-/merge_requests/8007

4.2.3 BND-CA-23-102: Journal File Handling Missing Sanity Checks

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4175
Affected Component:	lib/dns/journal.c:journal_open()

4.2.3.1 Description

The journal file handling code is missing sanity checks to prevent OOB reads and integer overflows that might result in OOB writes.

The calculation for *rawbytes* might overflow on 32-bit systems and less memory is allocated for *j->rawindex* and *j->index* than expected as shown in listing 4.19:

```
rawbytes = j->header.index_size * sizeof(journal_rawpos_t);
j->rawindex = isc_mem_get(mctx, rawbytes);

CHECK(journal_read(j, j->rawindex, rawbytes));

j->index = isc_mem_get(mctx, j->header.index_size *
                     sizeof(journal_pos_t));

p = j->rawindex;
for (i = 0; i < j->header.index_size; i++) {
    j->index[i].serial = decode_uint32(p);
    p += 4;
    j->index[i].offset = decode_uint32(p);
    p += 4;
}
```

Listing 4.19: Journal Handling

Additionally, the calls to ***decode_uint32()*** might read more data than available as shown in listing 4.20:

```
=================================================================
==136920==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x614000000400 at pc
↪ 0x7f66822e43b5 bp 0x7ffe1b4ffa10 sp 0x7ffe1b4ffa08
READ of size 1 at 0x614000000400 thread T0
    #0 0x7f66822e43b4 in decode_uint32 /home/eric/code/bind-9.19.13/lib/dns/journal.c:114:21
    #1 0x7f66822e43b4 in journal_open /home/eric/code/bind-9.19.13/lib/dns/journal.c:717:25
    #2 0x7f66822e25a8 in dns_journal_open /home/eric/code/bind-9.19.13/lib/dns/journal.c:775:11
    #3 0x7f66822ec85f in dns_journal_print /home/eric/code/bind-9.19.13/lib/dns/journal.c:1639:11
    #4 0x4ca513 in main /home/eric/code/bind-9.19.13/bin/tools/named-journalprint.c:125:12
    #5 0x7f66816b2d09 in __libc_start_main csu/../csu/libc-start.c:308:16
    #6 0x420489 in _start (/usr/bin/named-journalprint+0x420489)

0x614000000400 is located 0 bytes to the right of 448-byte region [0x614000000240,0x614000000400)
allocated by thread T0 here:
    #0 0x49a15d in malloc (/usr/bin/named-journalprint+0x49a15d)
    #1 0x7f6681adb290 in mallocx /home/eric/code/bind-9.19.13/lib/isc/./jemalloc_shim.h:65:14
    #2 0x7f6681adb290 in mem_get /home/eric/code/bind-9.19.13/lib/isc/mem.c:304:8
    #3 0x7f6681adb290 in isc__mem_get /home/eric/code/bind-9.19.13/lib/isc/mem.c:667:8

SUMMARY: AddressSanitizer: heap-buffer-overflow
↪   /home/eric/code/bind-9.19.13/lib/dns/journal.c:114:21 in decode_uint32
Shadow bytes around the buggy address:
  0x0c287fff8030: 00 00 00 00 00 00 00 00 00 00 fa fa fa fa fa fa
  0x0c287fff8040: fa fa fa fa fa fa fa fa 00 00 00 00 00 00 00 00
  0x0c287fff8050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c287fff8060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c287fff8070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c287fff8080:[fa]fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff8090: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff80a0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff80b0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff80c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c287fff80d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
```

Listing 4.20: ASan Output for Journal OOB Read

Since journal files are considered trusted in general, this is considered an informational note.

4.2.3.2 Solution Advice

X41 recommends to add safeguards and implement fuzz testing for journal file handling.

4.2.4 BND-CA-23-103: Unchecked malloc()

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4077
Affected Component:	lib/isc/thread.c:thread_wrap()

4.2.4.1 Description

In ***thread_wrap()***, the return value of ***malloc()*** is not checked and can potentially cause a *NULL* pointer dereference in low memory situations as seen in listing 4.21:

```
static struct thread_wrap *
thread_wrap(isc_threadfunc_t func, void *arg) {
        struct thread_wrap *wrap = malloc(sizeof(*wrap));
        *wrap = (struct thread_wrap){
                .func = func,
                .arg = arg,
        };

        return (wrap);
}
```

Listing 4.21: Unchecked malloc()

This issue is not likely to be enforceable by an attacker and in case of ***malloc()*** failing, the server would shutdown anyway. Therefore, this is considered an informational finding.

When reporting this issue it was noticed that this issue was reported in GitLab before as #4077[13].

4.2.4.2 Solution Advice

X41 recommends to handle the low memory situation more gracefully and print the proper error and perform a cleanup before the exit.

[13] https://gitlab.isc.org/isc-projects/bind9/-/issues/4077

4.2.5 BND-CA-23-104: Stack Exhaustion in Config Parser

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4176
Affected Component:	isccfg/parser.c:cfg_parse_listelt()

4.2.5.1 Description

When the configuration file contains too many opening brackets, the configuration parser will iterate until the stack is exhausted and the application crashes (see listing 4.22):

```
AddressSanitizer:DEADLYSIGNAL
=================================================================
==881685==ERROR: AddressSanitizer: stack-overflow on address 0x7fff92547ee8 (pc 0x00000049be0e bp
↪  0x7fff92548730 sp 0x7fff92547ef0 T0)
    #0 0x49be0e in __asan_memmove (/usr/bin/named-checkconf+0x49be0e)
    #1 0x7f322de80661 in isc_buffer_compact
    ↪  /home/eric/code/bind-9.19.13/lib/isc/./include/isc/buffer.h:1076:9
    #2 0x7f322de80661 in isc_lex_gettoken /home/eric/code/bind-9.19.13/lib/isc/lex.c:399:2
    #3 0x7f322d406da3 in cfg_gettoken /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:3474:11
    #4 0x7f322d4256be in cfg_peektoken /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:3537:2
    #5 0x7f322d4256be in parse_addrmatchelt
    ↪  /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:1715:2
    #6 0x7f322d414bf4 in cfg_parse_obj /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:248:11
    #7 0x7f322d414bf4 in cfg_parse_listelt
    ↪  /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:1989:11
    #8 0x7f322d412b58 in parse_list /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:2024:3
    #9 0x7f322d412b58 in cfg_parse_bracketed_list
    ↪  /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:2069:2
    #10 0x7f322d4258a3 in cfg_parse_obj /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:248:11
...
    #494 0x7f322d412b58 in parse_list /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:2024:3
    #495 0x7f322d412b58 in cfg_parse_bracketed_list
    ↪  /home/eric/code/bind-9.19.13/lib/isccfg/parser.c:2069:2

SUMMARY: AddressSanitizer: stack-overflow (/usr/bin/named-checkconf+0x49be0e) in __asan_memmove
==881685==ABORTING
```

Listing 4.22: Unbound Recursion in Configuration Parser

Since the configuration is considered trusted, this is considered an informational finding.

4.2.5.2 Solution Advice

X41 recommends to limit the iteration depth to a sane amount.

4.2.6 BND-CA-23-105: Connection Flags Mixup

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4126
Affected Component:	lib/isc/httpd.c:process_request()

4.2.6.1 Description

In `lib/isc/httpd.c` in function ***process_request()***, an HTTP request is parsed and the headers are evaluated.

When the client sets two *`Connection`* headers, the *`HTTPD_CLOSE`* and *`HTTPD_KEEPALIVE`* flags might be set on *`httpd->flags`* as shown in listing 4.23:

```
#define HTTPD_CLOSE          0x0001 /* Got a Connection: close header */
#define HTTPD_FOUNDHOST      0x0002 /* Got a Host: header */
#define HTTPD_KEEPALIVE      0x0004 /* Got a Connection: Keep-Alive */

...

} else if (name_match(header, "Connection")) {
    if (value_match(header, "close")) {
        httpd->flags |= HTTPD_CLOSE;
    } else if (value_match(header, "keep-alive")) {
        keep_alive = true;
    }

...

switch (httpd->minor_version) {
case 0:
    if (keep_alive == true) {
        httpd->flags |= HTTPD_KEEPALIVE;
    } else {
        httpd->flags |= HTTPD_CLOSE;
    }
    break;
```

Listing 4.23: Connection Flags Mixup

This currently has no impact but might lead to logic issues in the future when more features are added to the code.

4.2.6.2 Solution Advice

X41 recommends to clear the opposite flag when a new `Connection` header is read.

4.2.7 BND-CA-23-106: Use of Magic Numbers

GitLab Issue:	n/a
Affected Component:	lib/dns/catz.c, lib/dns/request.c, lib/dns/db.c

4.2.7.1 Description

A few places in the code use magic numbers[14] instead of using ***sizeof()*** or a named define.

Using defines or ***sizeof()*** makes the code easier to understand and audit as well as more robust to changes in the future as shown in the following listings:

```
static isc_result_t
catz_process_version(dns_catz_zone_t *catz, dns_rdataset_t *value) {
...
    char t[16];
...
        if (rdatastr.length > 15) {
                result = ISC_R_BADNUMBER;
                goto cleanup;
        }
```

Listing 4.24: Use of Magic Numbers in cat.c

```
isc_result_t
dns_request_createraw(dns_requestmgr_t *requestmgr, isc_buffer_t *msgbuf,
                      const isc_sockaddr_t *srcaddr,
                      const isc_sockaddr_t *destaddr,
                      dns_transport_t *transport,
                      isc_tlsctx_cache_t *tlsctx_cache, unsigned int options,
                      unsigned int timeout, unsigned int udptimeout,
                      unsigned int udpretries, isc_loop_t *loop, isc_job_cb cb,
                      void *arg, dns_request_t **requestp) {

...
        isc_buffer_usedregion(msgbuf, &r);
        if (r.length < DNS_MESSAGE_HEADERLEN || r.length > 65535) {
                result = DNS_R_FORMERR;
                goto cleanup;
        }

```

[14] https://en.wikipedia.org/wiki/Magic_number_(programming)

```
        if ((options & DNS_REQUESTOPT_TCP) != 0 || r.length > 512) {
                tcp = true;
                request->timeout = timeout * 1000;
        } else {
```

Listing 4.25: Use of Magic Numbers in request.c

```
isc_result_t
dns_db_getsoaserial(dns_db_t *db, dns_dbversion_t *ver, uint32_t *serialp) {
...
        INSIST(rdata.length > 20);
        isc_buffer_init(&buffer, rdata.data, rdata.length);
        isc_buffer_add(&buffer, rdata.length);
        isc_buffer_forward(&buffer, rdata.length - 20);
```

Listing 4.26: Use of Magic Numbers in db.c

```
memset(rdatalist->upper, 0xeb, sizeof(rdatalist->upper));
...
REQUIRE(rdatalist->upper[0] == 0xea);
```

Listing 4.27: Use of Magic Numbers in rdatalist.c

4.2.7.2 Solution Advice

X41 recommends to replace magic numbers with defines or calls to ***sizeof()***.

4.2.8 BND-CA-23-107: NULL Pointer Dereference on Wrong API Usage

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4177
Affected Component:	contrib/dlz/modules/sqlite3/dlz_sqlite3_dynamic.c:sqlite3_get_resultset()

4.2.8.1 Description

In `contrib/dlz/modules/sqlite3/dlz_sqlite3_dynamic.c` in function ***sqlite3_get_resultset()***, a wrong use of the API causes a goto *`cleanup`*. At this point the pointer *`dbi`* is initialized with *`NULL`* and dereferenced in the cleanup phase (see listing 4.28).

```c
static isc_result_t
sqlite3_get_resultset(const char *zone, const char *record, const char *client,
                      unsigned int query, void *dbdata, sqlite3_res_t **rsp) {
        isc_result_t result;
        dbinstance_t *dbi = NULL;
        sqlite3_instance_t *db = (sqlite3_instance_t *)dbdata;
        char *querystring = NULL;
        sqlite3_res_t *rs = NULL;
        int qres = 0;

        if ((query == COUNTZONE && rsp != NULL) ||
            (query != COUNTZONE && (rsp == NULL || *rsp != NULL)))
        {
                db->log(ISC_LOG_DEBUG(2), "Invalid result set pointer.");
                result = ISC_R_FAILURE;
                goto cleanup;
        }
...
cleanup:
        if (dbi->zone != NULL) {
                free(dbi->zone);
                dbi->zone = NULL;
        }
        if (dbi->record != NULL) {
                free(dbi->record);
                dbi->record = NULL;
        }
        if (dbi->client != NULL) {
                free(dbi->client);
                dbi->client = NULL;
        }

        /* release the lock so another thread can use this dbi */
        (void)dlz_mutex_unlock(&dbi->lock);

        if (querystring != NULL) {
                free(querystring);
        }

        return (result);
}
```

Listing 4.28: NULL Pointer Dereference on Wrong API Usage

Since this issue is in the contributed code and not the main code, it is considered an informational issue.

4.2.8.2 Solution Advice

X41 recommends to add an additional check against *NULL* before dereferencing *dbi*.

4.2.9 BND-CA-23-108: Invalid Free in Low Memory Situation

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4179
Affected Component:	contrib/dlz/modules/bdbhpt/dlz_bdbhpt_dynamic.c:dlz_allowzonexfr()

4.2.9.1 Description

In `contrib/dlz/modules/bdbhpt/dlz_bdbhpt_dynamic.c` in the function ***dlz_allowzonexfr()***, the memory of *`data`* is only initialized after *`key`* has been set up. When ***strdup()*** fails, e.g. due to a low memory situation, the value *`data.data`* might be passed to ***free()***, while still being uninitialized and pointing to arbitrary memory on the stack as seen in listing 4.29. This might result in freeing used and allocated memory, in a double free or segfault when non-accessible memory is accessed.

```
isc_result_t
dlz_allowzonexfr(void *dbdata, const char *name, const char *client) {
        isc_result_t result;
        bdbhpt_instance_t *db = (bdbhpt_instance_t *)dbdata;
        DBT key, data;

        /* check to see if we are authoritative for the zone first. */
#if DLZ_DLOPEN_VERSION >= 3
        result = dlz_findzonedb(dbdata, name, NULL, NULL);
#else  /* if DLZ_DLOPEN_VERSION >= 3 */
        result = dlz_findzonedb(dbdata, name);
#endif /* if DLZ_DLOPEN_VERSION >= 3 */
        if (result != ISC_R_SUCCESS) {
                return (ISC_R_NOTFOUND);
        }

        memset(&key, 0, sizeof(DBT));
        key.flags = DB_DBT_MALLOC;
        key.data = strdup(name);
        if (key.data == NULL) {
                result = ISC_R_NOMEMORY;
                goto xfr_cleanup;
        }
        key.size = strlen(key.data);

        memset(&data, 0, sizeof(DBT));

...

xfr_cleanup:
        /* free any memory duplicate string in the key field */
        if (key.data != NULL) {
                free(key.data);
        }

        /* free any memory allocated to the data field. */
        if (data.data != NULL) {
                free(data.data);
        }

        return (result);
}
```

Listing 4.29: Invalid Free in Low Memory Situation

Since this issue is in the contributed code and not the main code, it is considered an informational issue.

4.2.9.2 Solution Advice

X41 recommends to initialize `data` earlier in the function.

4.2.10 BND-CA-23-109: Possible Truncation in dns_keymgr_status()

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4180
Affected Component:	lib/dns/keymgr.c:dns_keymgr_status()

4.2.10.1 Description

The function ***isc_buffer_printf()*** returns the error value *ISC_R_NOSPACE* when the input buffer is not large enough to contain the result. Callers that do not provide a sufficiently large buffer and do not check the return value might operate on truncated strings without noticing. One such caller is ***dns_keymgr_status()*** which is called from ***named_server_dnssec()*** with a *4096* byte buffer as shown in listing 4.30.

```
isc_result_t
named_server_dnssec(named_server_t *server, isc_lex_t *lex,
                    isc_buffer_t **text) {
...
        char output[4096];
...
                dns_keymgr_status(kasp, &keys, now, &output[0], sizeof(output));

...

void
dns_keymgr_status(dns_kasp_t *kasp, dns_dnsseckeylist_t *keyring,
                  isc_stdtime_t now, char *out, size_t out_len) {
        isc_buffer_t buf;
        char timestr[26]; /* Minimal buf as per ctime_r() spec. */

        REQUIRE(DNS_KASP_VALID(kasp));
        REQUIRE(keyring != NULL);
        REQUIRE(out != NULL);

        isc_buffer_init(&buf, out, out_len);

        // policy name
        isc_buffer_printf(&buf, "dnssec-policy: %s\n", dns_kasp_getname(kasp));
        isc_buffer_printf(&buf, "current time:  ");
        isc_stdtime_tostring(now, timestr, sizeof(timestr));
        isc_buffer_printf(&buf, "%s\n", timestr);

        for (dns_dnsseckey_t *dkey = ISC_LIST_HEAD(*keyring); dkey != NULL;
             dkey = ISC_LIST_NEXT(dkey, link))
        {
                char algstr[DNS_NAME_FORMATSIZE];
                bool ksk = false, zsk = false;
                isc_result_t ret;

                if (dst_key_is_unused(dkey->key)) {
                        continue;
                }

                // key data
                dns_secalg_format((dns_secalg_t)dst_key_alg(dkey->key), algstr,
                                  sizeof(algstr));
                isc_buffer_printf(&buf, "\nkey: %d (%s), %s\n",
                                  dst_key_id(dkey->key), algstr,
                                  keymgr_keyrole(dkey->key));

                // publish status
                keytime_status(dkey->key, now, &buf,
                               "  published:      ", DST_KEY_DNSKEY,
                               DST_TIME_PUBLISH);
...
```

Listing 4.30: Possible Truncation in dns_keymgr_status()

4.2.10.2 Solution Advice

X41 recommends to either verify the return values or use a dynamic buffer.

4.2.11 BND-CA-23-110: Newline and ANSI Escape Code Injection via CC

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4181
Affected Component:	bin/named/control.c:named_control_docommand(), bin/delv/delv.c:main()

4.2.11.1 Description

When invalid commands are received, they are logged into a file, or when `named` is started with the *-g* parameter, to *stdout*. No sanitation is performed before the command data is output into the log as seen in listing 4.31.

```
} else {
        isc_log_write(named_g_lctx, NAMED_LOGCATEGORY_GENERAL,
                      NAMED_LOGMODULE_CONTROL, ISC_LOG_WARNING,
                      "unknown control channel command '%s'", command);
        result = DNS_R_UNKNOWNCOMMAND;
}
```

Listing 4.31: Newline and ANSI Escape Character Injection via CC

This can be abused by an authenticated attacker to inject ANSI[15] escape codes[16] into the output by calling `rndc` with a maliciously crafted command such as the one seen in listing 4.32.

```
/sbin/rndc -s 127.0.0.1 -p 953 -k /etc/bind/rndc.key `echo -e "'\x0a\x0d<fakedatehere> Normal
↪  \e[1mBold"`
```

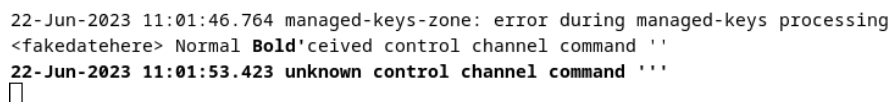
Listing 4.32: Using rndc to Inject Newlines and ANSI Escape Characters

This allows attackers to inject newlines for fake log entries or add arbitrary formatting to the data when viewed in a console as shown in figure 4.1.

Since this requires authentication as an already privileged user, this is considered an informational issue.

A similar issue exists in `bin/delv/delv.c` in the function ***main()*** when tracing is enabled via `+ns` and the *server* value is attacker controlled as shown in listing 4.33.

[15] American National Standards Institute
[16] https://en.wikipedia.org/wiki/ANSI_escape_code



```
22-Jun-2023 11:01:46.764 managed-keys-zone: error during managed-keys processing
<fakedatehere> Normal Bold'ceived control channel command ''
22-Jun-2023 11:01:53.423 unknown control channel command '''
```

Figure 4.1: Log Output with Injected Bold Code

```c
if (fulltrace && server != NULL) {
    delv_log(ISC_LOG_WARNING,
         "WARNING: using internal name server mode: "
         "'@%s' will be ignored",
         server);
}
```

Listing 4.33: Injection in delv.c

Since the *server* parameter should usually not be attacker controlled this is considered informational, but it might be when `delv` is called from a web-interface.

4.2.11.2 Solution Advice

X41 recommends to sanitize the data before sending it to ***isc_log_write()***, similar to the escaping in ***dns_name_totext2()***.

4.2.12 BND-CA-23-111: Name Buffer Truncation

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4186
Affected Component:	lib/isc/loop.c:dns_zonemgr_create()

4.2.12.1 Description

A truncation of the name of memory pools was found which might lead to unintended behavior or incorrect debugging output.

A memory pool structure *isc_mempool* has a member field *name* with a capacity of 16 bytes as shown in listing 4.34:

```
struct isc_mempool {
    /* always unlocked */
    unsigned int magic;
    isc_mem_t *mctx;           /*%< our memory context */
    ISC_LINK(isc_mempool_t) link; /*%< next pool in this mem context */
    element *items;               /*%< low water item list */
    size_t size;               /*%< size of each item on this pool */
    size_t allocated;           /*%< # of items currently given out */
    size_t freecount;           /*%< # of items on reserved list */
    size_t freemax;               /*%< # of items allowed on free list */
    size_t fillcount;           /*%< # of items to fetch on each fill */
    /*%< Stats only. */
    size_t gets; /*%< # of requests to this pool */
    /*%< Debugging only. */
    char name[16]; /*%< printed name in stats reports */
};
```

Listing 4.34: Name Structure Member Declaration in File lib/isc/mem.c

In the function ***dns_zonemgr_create()***, a string of size 16 without the terminating *NUL* byte is passed on to function ***isc_mem_setname()***, leading to silent truncation of the last character in that string as shown in the following listing 4.35:

```
for (size_t i = 0; i < zmgr->workers; i++) {
    isc_mem_create(&zmgr->mctxpool[i]);
    isc_mem_setname(zmgr->mctxpool[i], "zonemgr-mctxpool"); ⌋
↪  // MARK truncation / off by one (namebuffer is 16 bytes only)
}
```

Listing 4.35: Terminating NUL Byte Truncation

This issue is informational since the truncation has no security implications, but could lead to incorrect assumptions or functionality defects.

4.2.12.2 Solution Advice

X41 recommends to either increase the buffer size or shorten the name value, but to also add an assertion to the ***isc_mem_create()*** function that ensures the *name* size is larger than zero and less than 16 bytes without the terminating *NUL* byte.

4.2.13 BND-CA-23-112: Misaligned Structure Causes Exception

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4187
Affected Component:	lib/dns/rbtdb.c:allocate_version() (and others)

4.2.13.1 Description

The `named` process crashes with a segementation fault when compiled with the optimization flags *-march=native -ftree-slp-vectorize* and *jemalloc* disabed. During tests, X41 found that compilers clang[17] version 16 and GCC[18] versions 13.2.1 (Fedora 38) and 10.4 (Debian 11) were affected.

On a recent Intel CPU system, a crash can be reproduced reliably when compiling with the following commands shown in listing 4.36:

```
export CFLAGS="-O1 -g -march=native -ftree-slp-vectorize
export CXXFLAGS=$CFLAGS
make clean
./configure --without-jemalloc && make -j16 2>&1 | tee compile-warnings-log-$(date +%s.%N).txt
```

Listing 4.36: Compile with Vector Optimizations Enabled via -ftree-slp-vectorize

It was found that for *clang-16*, the issue is present even when compiling with flags *-O1 -march=native* only.

When starting `named` with the example configuration via the command line `./named -f -d10 -M fill`, a crash occurs (see listing 4.37):

```
Thread 1 "named" received signal SIGSEGV, Segmentation fault.
0x00007ffff7903136 in allocate_version (mctx=mctx@entry=0x5555556da340, serial=serial@entry=1,
↪  references=references@entry=1, writer=writer@entry=false) at rbtdb.c:1282
1282            ISC_LIST_INIT(version->resigned_list);
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
───────────────────────[ REGISTERS / show-flags off / show-compact-regs off
↪  ]──────────────────────────
*RAX  0x5555556dff20 <- 0x0
*RBX  0x5555556dfb10 <- 0xbebebebe00000001
 RCX  0x0
*RDX  0x28
*RDI  0x5555556dff20 <- 0x0
 RSI  0x0
```

[17] `https://clang.org`
[18] `https://gcc.gnu.org`

```
*R8   0x5555556dff20 <- 0x0
*R9   0x7ffff720cbe0 (main_arena+96) -> 0x5555556ff3f0 <- 0x211b0cca04000100
*R10  0x4000000
 R11  0x0
*R12  0x1
*R13  0x5555556da340 <- 0x44d656d43
 R14  0x0
*R15  0x1
*RBP  0x7fffffff89d0 <- 0x0
*RSP  0x7fffffff89a0 -> 0x5555556dad60 -> 0x5555556dfa90 <- 0x5242542b /* '+TBR' */
*RIP  0x7ffff7903136 (allocate_version+122) <- vmovdqa ymmword ptr [rbx + 0x20], ymm0
--------------------------[ DISASM / x86-64 / set emulate on ]-------------------------------
 > 0x7ffff7903136 <allocate_version+122>    vmovdqa ymmword ptr [rbx + 0x20], ymm0
   0x7ffff790313b <allocate_version+127>    mov    qword ptr [rbx + 0x40], -1
   0x7ffff7903143 <allocate_version+135>    mov    qword ptr [rbx + 0x48], -1
   0x7ffff790314b <allocate_version+143>    mov    rax, rbx
   0x7ffff790314e <allocate_version+146>    add    rsp, 8
   0x7ffff7903152 <allocate_version+150>    pop    rbx
   0x7ffff7903153 <allocate_version+151>    pop    r12
   0x7ffff7903155 <allocate_version+153>    pop    r13
   0x7ffff7903157 <allocate_version+155>    pop    r14
   0x7ffff7903159 <allocate_version+157>    pop    r15
   0x7ffff790315b <allocate_version+159>    pop    rbp
-------------------------------[ SOURCE (CODE) ]-------------------------------------
In file: /home/user/src/bind9-clean/lib/dns/rbtdb.c
   1277         version->glue_table = isc_mem_getx(mctx, size, ISC_MEM_ZERO);
   1278
   1279         version->writer = writer;
   1280         version->commit_ok = false;
   1281         ISC_LIST_INIT(version->changed_list);
 > 1282         ISC_LIST_INIT(version->resigned_list);
   1283         ISC_LINK_INIT(version, link);
   1284
   1285         return (version);
   1286 }
   1287
```

Listing 4.37: Segmentation Fault in allocate_version()

The crash occurs in function ***allocate_version()*** although the code looks unsuspicious as seen in listing 4.38:

```
static rbtdb_version_t *
allocate_version(isc_mem_t *mctx, rbtdb_serial_t serial,
         unsigned int references, bool writer) {
    rbtdb_version_t *version;
    size_t size;

    version = isc_mem_get(mctx, sizeof(*version));
    version->serial = serial;

    isc_refcount_init(&version->references, references);
    isc_rwlock_init(&version->glue_rwlock);

    version->glue_table_bits = ISC_HASH_MIN_BITS;
    version->glue_table_nodecount = 0U;

    size = ISC_HASHSIZE(version->glue_table_bits) *
           sizeof(version->glue_table[0]);
    version->glue_table = isc_mem_getx(mctx, size, ISC_MEM_ZERO);

    version->writer = writer;
    version->commit_ok = false;
    ISC_LIST_INIT(version->changed_list); ⌋
↪   // MARK incorrect optimization of these consecutive ISC_LIST_INIT statements
    ISC_LIST_INIT(version->resigned_list);
    ISC_LINK_INIT(version, link);

    return (version);
}
```

Listing 4.38: Code Pattern Example in Function allocate_version()

After extensive debugging it was found that an unaligned pointer is used in a x86_64 vector instruction:

`vmovdqa ymmword ptr [rbx + 0x20], ymm0`

Further investigation reveals that this seems to be caused by a misaligned address used in instruction *vmovdqa*, which requires a 32-bit or 64-bit alignment depending on the target architecture. The instruction is introduced due to automatic vectorization optimizations caused by the flags `-march=native -ftree-slp-vectorize`, which cause the compiler to use the native instruction set of the detected architecture and to apply auto-vectorization[19] performance optimizations.

[19] `https://gcc.gnu.org/projects/tree-ssa/vectorization.html`

Making changes to the order and interleaving of statements in the C code, which should not have any effect on the list operation semantics, makes the crash disappear as shown in listing 4.39:

```
ISC_LIST_INIT(version->changed_list);
volatile int noop = 1; if (noop) { // MARK no effect / NOOP to break the optimization
        ISC_LIST_INIT(version->resigned_list);
}
```

Listing 4.39: Prevent Optimization

However, additional crashes appear in the code shown in the following listing 4.40:

```
   2586                         /*
   2587                          * We're rolling back this transaction.
   2588                          */
   2589                         cleanup_list = version->changed_list;
   2590                         ISC_LIST_INIT(version->changed_list);
   2591                         resigned_list = version->resigned_list;
 > 2592                         ISC_LIST_INIT(version->resigned_list);
   2593                         rollback = true;
   2594                         cleanup_version = version;
   2595                         rbtdb->future_version = NULL;
   2596                 }
   2597         } else {
```

Listing 4.40: Another Crash in File lib/dns/rbtdb.c

When changing the alignments in the *rbtdb_version* struct via compiler attributes, the crash disappears as shown in the following code listing 4.41:

```
typedef struct rbtdb_version {
    /* Not locked */
    rbtdb_serial_t serial;
    dns_rbtdb_t *rbtdb;
    /*
     * Protected in the refcount routines.
     * XXXJT: should we change the lock policy based on the refcount
     * performance?
     */
    isc_refcount_t references;
    /* Locked by database lock. */
    bool writer;
    bool commit_ok;
    struct {} __attribute__ ((aligned (32))); // MARK force alignment
    rbtdb_changedlist_t changed_list;
    struct {} __attribute__ ((aligned (32)));
```

Listing 4.41: Patch to Enforce Correct Alignment

X41 considers the root cause to be the byte alignment of size 32, which exceeds the value of *max_align_t*[20].

Alignment values above *max_align_t* lead to *implementation defined behavior* according to section 6.2.8 of *ISO/IEC 9899:201x*[21]:

> An extended alignment is represented by an alignment greater than _Alignof (max_align_t). It is implementation-defined whether any extended alignments are supported and the contexts in which they are supported. A type having an extended alignment requirement is an over-aligned type.

Memory allocated using ***malloc*** will be aligned at least as strictly as *max_align_t*, but in this case the alignment required is greater.

Since this issue causes a crash already at startup, it is most likely not security relevant currently. Should it happen that an optimization is done in a part of the code that is not executed at startup, but could be reachable by attacker controller input, this issue could escalate into a denial of service issue.

4.2.13.2 Solution Advice

X41 recommends to either remove the specification of *alignas(ISC_OS_CACHELINE_SIZE)* that leads to extended alignment, or to change the allocation function from ***malloc*** to ***posix_memalign***[22].

[20] https://en.cppreference.com/w/c/types/max_align_t
[21] https://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
[22] https://man7.org/linux/man-pages/man3/posix_memalign.3.html

4.2.14 BND-CA-23-113: Race in dns_tsigkey_find()

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4182
Affected Component:	lib/dns/tsig.c:dns_tsigkey_find()

4.2.14.1 Description

There is a race condition which will trigger ***ISC_LINK_INSIST()*** when ***dns_tsigkey_find()*** is called twice for the same generated and expired key.

In the function ***dns_tsigkey_find()*** in `lib/dns/tsig.c`, a lookup for a key is performed based on the name and algorithm. If the found key is expired, it is removed from the keyring. These operations are protected by the *ring->lock* and depending on the access this lock is held in read or write mode as seen in listing 4.42.

```
isc_result_t
dns_tsigkey_find(dns_tsigkey_t **tsigkey, const dns_name_t *name,
                 const dns_name_t *algorithm, dns_tsig_keyring_t *ring) {
...
        RWLOCK(&ring->lock, isc_rwlocktype_read);
        key = NULL;
        result = dns_rbt_findname(ring->keys, name, 0, NULL, (void *)&key);
...
        if (key->inception != key->expire && isc_serial_lt(key->expire, now)) {
                /*
                 * The key has expired.
                 */
                RWUNLOCK(&ring->lock, isc_rwlocktype_read);
                RWLOCK(&ring->lock, isc_rwlocktype_write);
                remove_fromring(key);
                RWUNLOCK(&ring->lock, isc_rwlocktype_write);
                return (ISC_R_NOTFOUND);
        }
...
        RWUNLOCK(&ring->lock, isc_rwlocktype_read);
        adjust_lru(key);
        *tsigkey = key;
        return (ISC_R_SUCCESS);
}
```

Listing 4.42: Lock Handling in dns_tsigkey_find()

When the process gets interrupted after the call to ***RWUNLOCK(&ring->lock, isc_rwlocktype_read)***, and before the call to ***RWLOCK(&ring->lock, isc_rwlocktype_write)***, the call to ***remove_fromring()*** will be performed twice. For generated keys, the function ***remove_fromring()*** will try to unlink that key for each caller (see listing 4.43), which will trigger ***ISC_LINK_INSIST(ISC_LINK_LINKED(elt, link))***.

```
static void
remove_fromring(dns_tsigkey_t *tkey) {
        if (tkey->generated) {
                ISC_LIST_UNLINK(tkey->ring->lru, tkey, link);
                tkey->ring->generated--;
        }
        (void)dns_rbt_deletename(tkey->ring->keys, &tkey->name, false);
}
```

Listing 4.43: ISC_LIST_UNLINK() Called in remove_fromring()

4.2.14.2 Solution Advice

X41 recommends to ensure that the key still exists in the list in ***remove_fromring()*** or before calling the function. Additionally, the helper function ***UPGRADELOCK()*** could be used to clarify the code flow.

4.2.15 BND-CA-23-114: Pointers Dereferenced before Being Checked

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4432
Affected Component:	lib/isc/netmgr/streamdns.c:isc_nm_streamdnsconnect()

4.2.15.1 Description

In several places, pointers are dereferenced before being checked against *NULL*. As seen in listing 4.44, the pointer *mgr* is dereferenced to assign *worker* and then checked for validity, which includes a *NULL* pointer check. In case *mgr* is *NULL*, invalid memory is getting read which likely leads to a crash instead of a more controlled abort.

```
void
isc_nm_streamdnsconnect(isc_nm_t *mgr, isc_sockaddr_t *local,
                        isc_sockaddr_t *peer, isc_nm_cb_t cb, void *cbarg,
                        unsigned int timeout, isc_tlsctx_t *ctx,
                        isc_tlsctx_client_session_cache_t *client_sess_cache) {
        isc_nmsocket_t *nsock = NULL;
        isc__networker_t *worker = &mgr->workers[isc_tid()];

        REQUIRE(VALID_NM(mgr));
```

Listing 4.44: mgr Dereferenced Then Validated

Similar code exists in ***isc_nm_listenstreamdns()***, ***isc_nm_tcpconnect()***, ***isc_nm_listentls()***, ***isc_nm_tlsconnect()***, ***isc_nm_tcpconnect()*** and ***isc_nm_udpconnect()***.

sock is used in a similar pattern in ***isc__nm_udp_send()***. The *stats* pointer in ***dns_dnssecsignstats_increment()*** and ***dns_dnssecsignstats_clear()*** is accessed in the same way.

4.2.15.2 Solution Advice

X41 recommends to change the order of the validation and access of the pointers to ensure correct error messages in these failure cases.

4.2.16 BND-CA-23-115: Files Created with World Read/Write Permissions

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4443
Affected Component:	lib/isc/file.c:isc_file_openunique()

4.2.16.1 Description

During the code review it was found that the function ***isc_file_openunique()*** tries to create files with permission mode *0666* as shown in listing 4.45.

```
        isc_result_t
        isc_file_openunique(char *templet, FILE **fp) {
                int mode = S_IWUSR | S_IRUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH;
                return (isc_file_openuniquemode(templet, mode, fp));
        }
```

Listing 4.45: File Created with Mode 0666, World Read/Write

Unless a more restrictive *umask* is set, this results in the created file to be world read- and writable for any user on the system. The function ***isc_file_openunique()*** is involved in the creation of temporary files, zone files, and configuration files.

On nearly all modern systems, the umask[23] will be restrictive, mitigating a security impact because it will turn off corresponding bits requested in the file mode.

4.2.16.2 Solution Advice

It is recommended to remove the overly broad file mode permissions. Since on all modern systems a restrictive umask is set by default, the code will have not any effect and should be safe to remove.

[23] https://man.freebsd.org/cgi/man.cgi?query=umask&sektion=2

4.2.17 BND-CA-23-116: Locking Inconsistencies in Cache Implementation

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4340
Affected Component:	lib/dns/cache.c

4.2.17.1 Description

Several inconsistencies exist in the `lib/dns/cache.c` in regards to locking.

dns_cache_setcachesize() locks `cache->lock` to protect `cache->size`. But setting and retrieving this variable is basically a noop, since it is never used for anything meaningful.

`cache->serve_stale_ttl` is protected by `cache->lock` as well in ***dns_cache_setservestalettl()*** but not in ***cache->serve_stale_ttl()***, which can be called via ***dns_cache_flush()***.

Furthermore, ***dns_cache_getservestalerefresh()*** uses ***dns_db_getservestalerefresh()*** to return the value instead of the locally stored copy in `cache`.

dns_db_getservestalerefresh() does not do any locking in the variable it sets and might therefore race against ***dns_db_getservestalerefresh()***.

4.2.17.2 Solution Advice

X41 recommends to cleanup the locking and check which functions can be removed or need to be reimplemented.

4.2.18 BND-CA-23-117: Endless Loop via GENERATE

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4353
Affected Component:	lib/dns/master.c:genname()

4.2.18.1 Description

In `lib/dns/master.c`, in function ***genname()***, `it` and `delta` are both of type `int`. There is an issue when `delta` is negative (which nothing prevents) when calling ***nibbles()***. The implemented check (see listing 4.46) does not create an error, since a negative value is always smaller than `INT_MAX - it` for a positive value of `it`.

```c
/*
 * 'it' is >= 0 so we don't need to check for
 * underflow.
 */
if ((it > 0 && delta > INT_MAX - it)) {
    return (ISC_R_RANGE);
}
if (nibblemode) {
    n = nibbles(numbuf, sizeof(numbuf), width,
            mode[0], it + delta);
} else {
    n = snprintf(numbuf, sizeof(numbuf), fmt,
             it + delta);
}
```

Listing 4.46: Insufficient Check in genname()

In ***nibbles()***, the right shift of `value` will sign extend, and `value` will never reach 0, which causes an infinite loop (see listing 4.47).

```c
static unsigned int
nibbles(char *numbuf, size_t length, unsigned int width, char mode, int value) {
        unsigned int count = 0;

        /*
         * This reserve space for the NUL string terminator.
         */
        if (length > 0U) {
                *numbuf = '\0';
                length--;
        }
        do {
                char val = hex[(value & 0x0f) + ((mode == 'n') ? 0 : 16)];
                value >>= 4;
                if (length > 0U) {
                        *numbuf++ = val;
                        *numbuf = '\0';
                        length--;
                }
                if (width > 0) {
                        width--;
                }
                count++;
                /*
                 * If width is non zero then we need to add a label separator.
                 * If value is non zero then we need to add another label and
                 * that requires a label separator.
                 */
                if (width > 0 || value != 0) {
                        if (length > 0U) {
                                *numbuf++ = '.';
                                *numbuf = '\0';
                                length--;
                        }
                        if (width > 0) {
                                width--;
                        }
                        count++;
                }
        } while (value != 0 || width > 0);
        return (count);
}
```

Listing 4.47: Right Shift in nibbbles()

Additionally, when *width* is a large number, the processing can take some time as well, but will terminate eventually.

Since `GENERATE` is only allowed in trusted zone files, it should not be possible for an attacker to trigger this.

4.2.18.2 Solution Advice

X41 recommends to convert *value* to *unsigned int*.

4.2.19 BND-CA-23-118: Endless Loop via INCLUDE

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4357
Affected Component:	lib/dns/master.c:load_text()

4.2.19.1 Description

When loading a zonefile, e.g. via `./named-compilezone -d -i full -o /dev/null x input` we can enter another endless loop, when the file just contains `$INCLUDE .` statement.

The loop happens via the callchain seen in listing 4.48:

```
#0  load_text (lctx=0x7ffff1a54300) at master.c:1083
#1  0x00007ffff78042f6 in dns_master_loadfile (master_file=<optimized out>, top=<optimized out>,
↪   origin=<optimized out>, zclass=<optimized out>, options=<optimized out>, resign=<optimized
↪   out>, callbacks=<optimized out>,
    include_cb=<optimized out>, include_arg=<optimized out>, mctx=<optimized out>,
    ↪  format=<optimized out>, maxttl=<optimized out>) at master.c:2637
#2  0x00007ffff7bbd66b in zone_startload (db=0x7ffff1a18800, zone=<optimized out>,
↪   zone@entry=0x7ffff1a90000, loadtime=...) at zone.c:2641
#3  0x00007ffff7b6e362 in zone_load (zone=<optimized out>, flags=<optimized out>,
↪   locked=<optimized out>) at zone.c:2315
#4  0x000055555563cd3d in load_zone (mctx=<optimized out>, zonename=<optimized out>,
↪   filename=<optimized out>, fileformat=<optimized out>, classname=<optimized out>,
↪   maxttl=<optimized out>, zonep=<optimized out>)
    at check-tool.c:639
#5  0x000055555563afb0 in main (argc=<optimized out>, argv=<optimized out>) at
↪   named-checkzone.c:546
```

Listing 4.48: Zonefile Loading Callchain

As seen in listing 4.49, the code in ***load_text()*** contains an infinite loop:

```
while (true) {
...

        } else if (strcasecmp(DNS_AS_STR(token), "$INCLUDE") ==
               0)
        {
            COMMITALL;
...
            GETTOKEN(lctx->lex, ISC_LEXOPT_QSTRING, &token,
                 false);
            if (include_file != NULL) {
                isc_mem_free(mctx, include_file);
            }

...
next_line:;
}
```

Listing 4.49: load_text() infinite loop

Since ***gettoken()*** returns *ISC_R_NOTFILE*, the jump to *next_line* is taken and the parser ends up in an infinite loop as shown in listing 4.50.

```
#define GETTOKENERR(lexer, options, token, eol, err)                    \
        do {                                                            \
                result = gettoken(lexer, options, token, eol, callbacks); \
                switch (result) {                                       \
                case ISC_R_SUCCESS:                                     \
                        break;                                          \
                case ISC_R_UNEXPECTED:                                  \
                        goto insist_and_cleanup;                        \
                default:                                                \
                        if (MANYERRS(lctx, result)) {                   \
                                SETRESULT(lctx, result);                \
                                LOGIT(result);                          \
                                read_till_eol = true;                   \
                                err goto next_line;                     \
        ...
        } while (0)
#define GETTOKEN(lexer, options, token, eol) \
        GETTOKENERR(lexer, options, token, eol, {})
```

Listing 4.50: GETTOKEN Macro

4.2.19.2 Solution Advice

X41 recommends to catch the *ISC_R_NOTFILE* case in the same manner as *ISC_R_UNEXPECTED*.

4.2.20 BND-CA-23-119: Supplied Buffer Too Large

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4433
Affected Component:	bin/tests/wire_test.c

4.2.20.1 Description

The code in `wire_test.c` provides a *64*1024*-byte buffer to ***dns_message_renderbegin()***. This might trigger an error in a corner case, where the code is expecting to receive buffers that are not larger than 65536 bytes as shown in listing 4.51.

```
if (result != ISC_R_SUCCESS) {
    INSIST(st.used < 65536);
    dns_compress_rollback(
        msg->cctx, (uint16_t)st.used);
    *(msg->buffer) = st; /* rollback */
    msg->buffer->length += msg->reserved;
    msg->counts[sectionid] += total;
    maybe_clear_ad(msg, sectionid);
    return (result);
}
```

Listing 4.51: realloc() Misuse

4.2.20.2 Solution Advice

X41 recommends to reduce the buffer size by one to prevent the error from happening.

4.2.21 BND-CA-23-120: Dead Code in DNSTAP Helper

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4406
Affected Component:	bin/tools/dnstap-read.c

4.2.21.1 Description

The code in `bin/tools/dnstap-read.c` allocates memory for *b* in ***main()***, never uses it and correctly frees it again.

```
int
main(int argc, char *argv[]) {
...
    isc_buffer_t *b = NULL;
...
        for (;;) {
...
                if (b != NULL) {
                        isc_buffer_free(&b);
                }
                isc_buffer_allocate(mctx, &b, 2048);
                if (b == NULL) {
                        fatal("out of memory");
                }
...
cleanup:
...
        if (b != NULL) {
                isc_buffer_free(&b);
        }
        isc_mem_destroy(&mctx);

        exit(rv);
}
```

Listing 4.52: Dead Code in DNSTAP Helper

4.2.21.2 Solution Advice

X41 recommends to remove the dead code.

4.2.22 BND-CA-23-121: Unused AES Functions

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4421
Affected Component:	lib/isc/aes.c

4.2.22.1 Description

The AES[24] functions ***isc_aes256_crypt()*** and ***isc_aes192_crypt()*** in `lib/isc/aes.c` have no callers besides test code and are therefore dead code.

4.2.22.2 Solution Advice

X41 recommends to remove unused functions and dead code.

[24] Advanced Encryption Standard

4.2.23 BND-CA-23-122: Usage of isc_safe_memwipe()

GitLab Issue:	https://gitlab.isc.org/isc-projects/bind9/-/issues/4435
Affected Component:	lib/isc/safe.c

4.2.23.1 Description

The function ***isc_safe_memwipe()*** is used to wipe memory in a way that is not optimized away by a compiler. An optimizing compiler could remove a call to ***memset()*** if the memory area is passed to ***free()*** afterwards. The functions seems to be used to remove key material from memory that is passed to BIND 9 via files. It is likely that key material is also left in heap or stack variables of the lexer used to parse the files. This might cause a false sense of security since not all key data is removed from memory. In case the threat model sees this as an actual threat, more memory areas need to be wiped.

Nevertheless, removing key material already lowers the chances of key material leaking.

4.2.23.2 Solution Advice

X41 recommends to clarify the threat model and either remove the function ***isc_safe_memwipe()*** to avoid a false sense of security or enforce proper memory wiping in more places.

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

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

Acronyms

ACL	Access Control List	12
AES	Advanced Encryption Standard	73
ANSI	American National Standards Institute	53
API	Application Programming Interface	36
AXFR	Asynchronous Full Transfer Zone	10
CVE	Common Vulnerabilities and Exposures	12
CWE	Common Weakness Enumeration	14
DNS	Domain Name System	6
DoS	Denial of Service	6
EDNS	Extended DNS	10
HTTP	HyperText Transfer Protocol	19
ID	Identifier	12
IXFR	Incremental Zone Transfer	10
OOB	Out-of-Bounds	9
RCE	Remote Code Execution	6
RCU	Read, Copy, Update	10
TCP	Transmission Control Protocol	23

A Appendix

A.1 Fuzz Testing

Various fuzz harnesses already exist for the BIND 9 nameserver. Among these are those shipped with bind[1][2] and hongfuzz[3]. X41 created additional harnesses to either target functions more directly or to test parts that were not covered by the fuzz testing before. Due to the already working fuzzing setup, these could be quickly added whenever a promising target was encountered during the source code audit.

A.1.1 isc_url_parse() Fuzz Harness

The fuzz harness created for ***isc_url_parse()*** is shown in listing A.1.

```
/*
 * Copyright (C) Internet Systems Consortium, Inc. ("ISC")
 *
 * SPDX-License-Identifier: MPL-2.0
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, you can obtain one at https://mozilla.org/MPL/2.0/.
 *
 * See the COPYRIGHT file distributed with this work for additional
 * information regarding copyright ownership.
 */

#include <stddef.h>
#include <stdint.h>

#include <isc/url.h>

#include "fuzz.h"

int
LLVMFuzzerInitialize(int *argc ISC_ATTR_UNUSED, char ***argv ISC_ATTR_UNUSED) {
```

[1] https://gitlab.isc.org/isc-projects/bind9/-/tree/main/fuzz
[2] https://gitlab.isc.org/isc-projects/bind9/-/blob/main/bin/named/fuzz.c
[3] https://github.com/google/honggfuzz/tree/master/examples/bind

```

    return (0);
}

int
LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {

    isc_url_parser_t p;

    memset(&p, 0, sizeof(p));
    isc_url_parse(data, size, true, &p);
    memset(&p, 0, sizeof(p));
    isc_url_parse(data, size, false, &p);

    return (0);
}
```

Listing A.1: isc_url_parse() Fuzz Harness

A.1.2 phr_parse_request() Fuzz Harness

The fuzz harness created for ***phr_parse_request()*** is shown in listing A.2.

```
/*
 * Copyright (C) Internet Systems Consortium, Inc. ("ISC")
 *
 * SPDX-License-Identifier: MPL-2.0
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, you can obtain one at https://mozilla.org/MPL/2.0/.
 *
 * See the COPYRIGHT file distributed with this work for additional
 * information regarding copyright ownership.
 */

#include <stddef.h>
#include <stdint.h>

#include <picohttpparser.h>

#include "fuzz.h"

#define HTTP_HEADERS_NUM     100

int
LLVMFuzzerInitialize(int *argc ISC_ATTR_UNUSED, char ***argv ISC_ATTR_UNUSED) {

    return (0);
}

int
LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {
        const char *method = NULL;
        size_t method_len = 0;
        const char *path;
        size_t path_len = 0;
    int minor_version;
        struct phr_header headers[HTTP_HEADERS_NUM];
        size_t num_headers = HTTP_HEADERS_NUM;

    phr_parse_request((const char *) data, size, &method,
              &method_len, &path, &path_len,
              &minor_version, headers,
              &num_headers, size);

    return (0);
}
```

Listing A.2: phr_parse_request() Fuzz Harness

A.1.3 isc_regex_validate() Fuzz Harness

The fuzz harness created for ***isc_regex_validate()*** is shown in listing A.3.

```
/*
 * Copyright (C) Internet Systems Consortium, Inc. ("ISC")
 *
 * SPDX-License-Identifier: MPL-2.0
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, you can obtain one at https://mozilla.org/MPL/2.0/.
 *
 * See the COPYRIGHT file distributed with this work for additional
 * information regarding copyright ownership.
 */

#include <stddef.h>
#include <stdint.h>

#include <isc/regex.h>

#include "fuzz.h"

int
```

```c
LLVMFuzzerInitialize(int *argc ISC_ATTR_UNUSED, char ***argv ISC_ATTR_UNUSED) {

    return (0);
}

int
LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {


    char *r = malloc(size + 1);
    memcpy(r, data, size);
    r[size] = 0;

    isc_regex_validate(r);

    free(r);
    return (0);
}
```

Listing A.3: isc_regex_validate() Fuzz Harness

A.1.4 isc_utf8_valid() Fuzz Harness

The fuzz harness created for ***isc_utf8_valid()*** is shown in listing A.4.

```c
/*
 * Copyright (C) Internet Systems Consortium, Inc. ("ISC")
 *
 * SPDX-License-Identifier: MPL-2.0
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, you can obtain one at https://mozilla.org/MPL/2.0/.
 *
 * See the COPYRIGHT file distributed with this work for additional
 * information regarding copyright ownership.
 */

#include <stddef.h>
#include <stdint.h>

#include <isc/utf8.h>

#include "fuzz.h"

int
LLVMFuzzerInitialize(int *argc ISC_ATTR_UNUSED, char ***argv ISC_ATTR_UNUSED) {

    return (0);
}

int
LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {

    isc_utf8_valid(data, size);

    return (0);
}
```

Listing A.4: isc_utf8_valid() Fuzz Harness

A.1.5 isccc_cc_fromwire() Fuzz Harness

The fuzz harness created for ***isccc_cc_fromwire()*** is shown in listing A.5.

```
/*
 * Copyright (C) Internet Systems Consortium, Inc. ("ISC")
 *
 * SPDX-License-Identifier: MPL-2.0
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, you can obtain one at https://mozilla.org/MPL/2.0/.
 *
 * See the COPYRIGHT file distributed with this work for additional
 * information regarding copyright ownership.
 */

#include <stdbool.h>
#include <stdlib.h>

#include <isc/buffer.h>
#include <isc/mem.h>
#include <isc/util.h>

#include <isccc/alist.h>
#include <isccc/cc.h>
#include <isccc/ccmsg.h>
#include <isccc/sexpr.h>
#include <isccc/symtab.h>
#include <isccc/util.h>

#include "fuzz.h"

bool debug = false;

```

```c
int
LLVMFuzzerInitialize(int *argc, char ***argv) {
    UNUSED(argc);
    UNUSED(argv);
    return (0);
}

int
LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {

    isccc_sexpr_t *alistp = NULL;
    isccc_region_t source;

    source.rstart = (char *) data;
    source.rend = (char *) data + size;

    // fuzzing table_fromwire() directly would be more effective,
    // but its currently not exported

    isccc_cc_fromwire(&source, &alistp, 0, NULL);

    if (alistp)
        isccc_sexpr_free(&alistp);
    return 0;
}
```

Listing A.5: isccc_cc_fromwire() Fuzz Harness

A.1.6 irs_resconf_load() AFL Harness

The AFL harness created for ***irs_resconf_load()*** is shown in listing A.6.

```c
/*
 * Copyright (C) Internet Systems Consortium, Inc. ("ISC")
 *
 * SPDX-License-Identifier: MPL-2.0
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, you can obtain one at https://mozilla.org/MPL/2.0/.
 *
 * See the COPYRIGHT file distributed with this work for additional
 * information regarding copyright ownership.
 */


#include <stdlib.h>

```

```
#include <isc/mem.h>

#include <irs/resconf.h>

/*% Main processing routine for dig */
int
main(int argc, char **argv) {
    isc_result_t result;
        irs_resconf_t *resconf = NULL;
    static isc_mem_t *mctx = NULL;
    char *fn;

    if (argc < 2)
        return -1;
    fn = argv[1];

    isc_mem_create(&mctx);

        result = irs_resconf_load(mctx, fn, &resconf);

        if (resconf != NULL) {
                irs_resconf_destroy(&resconf);
        }


    return 0;
}
```

Listing A.6: irs_resconf_load() AFL Harness

A.1.7 dig/host AFL++ ARGV Harness

Only very light argument fuzzing was performed using AFL++ on the `dig` and `host` tools. The reasoning for this test was that they might be used on cheap routers with attacker-influenced command line arguments stemming from the web interface. The main parts of that harness can be seen in listing A.7.

```
...
#include "/home/eric/tools/AFLplusplus/utils/argv_fuzzing/argv-fuzz-inl.h"


/*% Main processing routine for dig */
int
main(int argc, char **argv) {
        isc_result_t result;
        irs_resconf_t *resconf = NULL;
        static isc_mem_t *mctx = NULL;
        char *fn;

        AFL_INIT_SET0("dig");

...
```

Listing A.7: dig/host AFL++ ARGV Harness

A.1.8 Scapy Template-Based Fuzzing

The following simple template-based fuzzer generates multiple packets to be sent in (nearly) parallel:

```python
#! /usr/bin/env python3

from scapy.all import DNS, DNSQR, IPv6, IPv6, sr1, UDP, fuzz, send

dns_pkt1 = fuzz(DNS(qd=fuzz(DNSQR(qname="kk.foobar"))))
dns_pkt11 = fuzz(DNS(qd=fuzz(DNSQR(qname="kk.foobar\x00"))))
dns_pkt12 = fuzz(DNS(qd=fuzz(DNSQR(qname="41.41.41.41"))))
dns_pkt13 = fuzz(DNS(qd=fuzz(DNSQR(qname="::ff"))))
dns_pkt2 = fuzz(DNS(qd=fuzz(DNSQR())))
dns_pkt3 = fuzz(DNS(rd=1, qd=DNSQR()))
while 1:
    send(IPv6(dst='::1')/UDP(sport=53, dport=53)/dns_pkt1, verbose=0)
    send(IPv6(dst='::1')/UDP(sport=53, dport=53)/dns_pkt11, verbose=0)
    send(IPv6(dst='::1')/UDP(sport=53, dport=53)/dns_pkt12, verbose=0)
    send(IPv6(dst='::1')/UDP(sport=53, dport=53)/dns_pkt13, verbose=0)
    send(IPv6(dst='::1')/UDP(sport=53, dport=53)/dns_pkt2, verbose=0)
    send(IPv6(dst='::1')/UDP(dport=53)/dns_pkt3, verbose=0)
```

Listing A.8: Scapy Fuzz Script

A.1.9 Speedup of dns_rdata_fromwire_text.c

The fuzzing harness `fuzz/dns_rdata_fromwire_text.c` can be sped up by a factor of 200 by moving some initialisation routines into ***LLVMFuzzerInitialize()*** as shown in listing A.9.

```diff
--- bind-9.19.17-orig/fuzz/dns_rdata_fromwire_text.c    2023-09-08 11:09:28.635415105 +0200
+++ bind-9.19.17/fuzz/dns_rdata_fromwire_text.c    2023-09-25 08:46:45.028588658 +0200
@@ -40,6 +40,8 @@ bool debug = false;

 static isc_mem_t *mctx = NULL;
 static isc_lex_t *lex = NULL;
+unsigned int types = 1;
```

```
+dns_rdatatype_t typelist[256] = { 1000 }; /* unknown */

 int
 LLVMFuzzerInitialize(int *argc ISC_ATTR_UNUSED, char ***argv ISC_ATTR_UNUSED) {
@@ -56,6 +58,24 @@ LLVMFuzzerInitialize(int *argc ISC_ATTR_
     isc_lex_setspecials(lex, specials);
     isc_lex_setcomments(lex, ISC_LEXCOMMENT_DNSMASTERFILE);

+    unsigned int t;
+    /*
+     * Append known types to list.
+     */
+    for (t = 1; t <= 0x10000; t++) {
+        char typebuf[256];
+        if (dns_rdatatype_ismeta(t)) {
+            continue;
+        }
+        dns_rdatatype_format(t, typebuf, sizeof(typebuf));
+        if (strncmp(typebuf, "TYPE", 4) != 0) {
+            /* Assert when we need to grow typelist. */
+            assert(types < sizeof(typelist) / sizeof(typelist[0]));
+            typelist[types++] = t;
+        }
+    }
+
+
     return (0);
 }

@@ -73,13 +93,13 @@ nullmsg(dns_rdatacallbacks_t *cb, const
     }
 }

+
 int
 LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {
     char totext[64 * 1044 * 4];
     dns_compress_t cctx;
     dns_rdatatype_t rdtype;
     dns_rdataclass_t rdclass;
-    dns_rdatatype_t typelist[256] = { 1000 }; /* unknown */
     dns_rdataclass_t classlist[] = { dns_rdataclass_in, dns_rdataclass_hs,
                      dns_rdataclass_ch, dns_rdataclass_any,
                      60 };
@@ -92,7 +112,7 @@ LLVMFuzzerTestOneInput(const uint8_t *da
     unsigned char fromwire[1024];
     unsigned char towire[1024];
     unsigned int classes = (sizeof(classlist) / sizeof(classlist[0]));
-    unsigned int types = 1, flags, t;
+    unsigned int flags;

     /*
```

```
      * First 2 bytes are used to select type and class.
@@ -102,21 +122,6 @@ LLVMFuzzerTestOneInput(const uint8_t *da
         return (0);
     }

-    /*
-     * Append known types to list.
-     */
-    for (t = 1; t <= 0x10000; t++) {
-        char typebuf[256];
-        if (dns_rdatatype_ismeta(t)) {
-            continue;
-        }
-        dns_rdatatype_format(t, typebuf, sizeof(typebuf));
-        if (strncmp(typebuf, "TYPE", 4) != 0) {
-            /* Assert when we need to grow typelist. */
-            assert(types < sizeof(typelist) / sizeof(typelist[0]));
-            typelist[types++] = t;
-        }
-    }

    /*
     * Random type and class from a limited set.
```

Listing A.9: Speedup of dns_rdata_fromwire_text.c

Additionally, it is recommended to add a call to ***isc_lex_close()*** to match the calls to ***isc_lex_openbuffer()*** to avoid memory leaks. The same memory leaks appear in `isc_lex_getmastertoken.c`.

A.1.10 badcache Stress Testing

Stress testing was performed against the badcache implementation by modifying `tests/dns/badcache_test.c`. The modified test makes modifications to a shared cache in parallel.

```c
/*
 * Copyright (C) Internet Systems Consortium, Inc. ("ISC")
 *
 * SPDX-License-Identifier: MPL-2.0
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, you can obtain one at https://mozilla.org/MPL/2.0/.
 *
 * See the COPYRIGHT file distributed with this work for additional
 * information regarding copyright ownership.
 */

#include <inttypes.h>
#include <sched.h> /* IWYU pragma: keep */
#include <stdarg.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <isc/buffer.h>
#include <isc/mem.h>
#include <isc/os.h>
#include <isc/thread.h>
#include <isc/urcu.h>
#include <isc/util.h>
#include <isc/uv.h>
#include <isc/loop.h>
#include <isc/async.h>

#include <dns/fixedname.h>
#include <dns/badcache.h>
#include <dns/name.h>
#include <dns/rdatatype.h>

#define BADCACHE_TEST_FLAG 1 << 3

static void
basic(dns_badcache_t *bc) {
    dns_fixedname_t fname = { 0 };
    dns_name_t *name = dns_fixedname_initname(&fname);
    isc_stdtime_t now = isc_stdtime_now();
    uint32_t flags = BADCACHE_TEST_FLAG;

    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now + 60);

    flags = 0;
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    flags = 0;
    dns_badcache_find(bc, name, dns_rdatatype_a, &flags, now);
}

static void
expire(dns_badcache_t *bc) {
    dns_fixedname_t fname = { 0 };
    dns_name_t *name = dns_fixedname_initname(&fname);
    isc_stdtime_t now = isc_stdtime_now();
    uint32_t flags = BADCACHE_TEST_FLAG;

    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now + 60);
    dns_badcache_add(bc, name, dns_rdatatype_a, false, flags, now + 60);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags,
                      now + 61);

    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_badcache_find(bc, name, dns_rdatatype_a, &flags, now);
    dns_badcache_add(bc, name, dns_rdatatype_a, true, flags, now + 120);
    dns_badcache_find(bc, name, dns_rdatatype_a, &flags, now + 61);
}

static void
print(dns_badcache_t *bc) {
    dns_fixedname_t fname = { 0 };
    dns_name_t *name = dns_fixedname_initname(&fname);
    isc_stdtime_t now = isc_stdtime_now();
    isc_stdtime_t expire = now + 60;
    uint32_t flags = BADCACHE_TEST_FLAG;
    FILE *file = NULL;

    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_a, false, flags, expire);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, expire);

    file = fopen("/dev/null", "w");
    dns_badcache_print(bc, "badcache", file);
    fclose(file);
}

static void
flush(dns_badcache_t *bc) {
    dns_fixedname_t fname = { 0 };
    dns_name_t *name = dns_fixedname_initname(&fname);
    isc_stdtime_t now = isc_stdtime_now();
    uint32_t flags = BADCACHE_TEST_FLAG;

    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now + 60);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_badcache_flush(bc);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
}

static void
flushname(dns_badcache_t *bc) {
    dns_fixedname_t fname = { 0 };
    dns_name_t *name = dns_fixedname_initname(&fname);
    isc_stdtime_t now = isc_stdtime_now();
    uint32_t flags = BADCACHE_TEST_FLAG;

    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now + 60);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_name_fromstring(name, "sub.example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now + 60);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_name_fromstring(name, "sub.sub.example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now + 60);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_name_fromstring(name, "sub.example.com.", NULL, 0, NULL);
    dns_badcache_flushname(bc, name);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_badcache_find(bc, name, dns_rdatatype_a, &flags, now);
    dns_name_fromstring(name, "sub.sub.example.com.", NULL, 0, NULL);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
}

static void
flushtree(dns_badcache_t *bc) {
    dns_fixedname_t fname = { 0 };
    dns_name_t *name = dns_fixedname_initname(&fname);
    isc_stdtime_t now = isc_stdtime_now();
    uint32_t flags = BADCACHE_TEST_FLAG;

    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now + 60);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_name_fromstring(name, "sub.example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now + 60);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_name_fromstring(name, "sub.sub.example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now + 60);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_name_fromstring(name, "sub.example.com.", NULL, 0, NULL);
    dns_badcache_flushtree(bc, name);
    dns_name_fromstring(name, "sub.sub.example.com.", NULL, 0, NULL);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_name_fromstring(name, "sub.example.com.", NULL, 0, NULL);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);
    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags, now);

}

static void
purge(dns_badcache_t *bc) {
    dns_fixedname_t fname = { 0 };
    dns_name_t *name = dns_fixedname_initname(&fname);
    isc_stdtime_t now = isc_stdtime_now();
    uint32_t flags = BADCACHE_TEST_FLAG;

    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags,
                      now - 60);
    dns_name_fromstring(name, "sub.example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags,
                      now - 60);
    dns_name_fromstring(name, "sub.sub.example.com.", NULL, 0, NULL);
    dns_badcache_add(bc, name, dns_rdatatype_aaaa, false, flags, now);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags,
                      now - 60);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags,
                      now + 30);
    dns_name_fromstring(name, "sub.sub.example.com.", NULL, 0, NULL);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags,
                      now + 30);
    dns_name_fromstring(name, "sub.example.com.", NULL, 0, NULL);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags,
                      now + 30);
    dns_name_fromstring(name, "example.com.", NULL, 0, NULL);
    dns_badcache_find(bc, name, dns_rdatatype_aaaa, &flags,
                      now + 30);
}

isc_loopmgr_t *loopmgr = NULL;

void loop(dns_badcache_t *bc);
void loop(dns_badcache_t *bc) {
    int i = 0;
    for (i = 0; i < 1000; i++) {
        basic(bc);
        expire(bc);
        print(bc);
        flush(bc);
        flushname(bc);
        flushtree(bc);
        purge(bc);
    }
    isc_loopmgr_shutdown(loopmgr);
}
int main(__attribute__((unused)) int argc, __attribute__((unused)) char **argv) {
    dns_badcache_t *bc = NULL;
    isc_mem_t *mctx = NULL;

    isc_mem_create(&mctx);
    isc_loopmgr_create(mctx, 100, &loopmgr);

    bc = dns_badcache_new(mctx);

    isc_loopmgr_setup(loopmgr, (isc_job_cb) loop, bc);
    isc_loopmgr_run(loopmgr);


    dns_badcache_destroy(&bc);
    isc_loopmgr_shutdown(loopmgr);
}
```

Listing A.10: badcache Stress Testing

A.2 CodeQL Queries

The following queries were used to inspect the codebase and may be desirable to integrate into automated analysis pipelines:

A.2.1 Mutex DataFlow Query

```
iimport cpp
import semmle.code.cpp.dataflow.DataFlow

class Config extends DataFlow::Configuration {
  Config() { this = "MutexVarFlow" }

  override predicate isSource(DataFlow::Node source) {
    exists(FunctionCall alloc, DataFlow::Node v |
      alloc.getTarget().getName() = "pthread_mutex_init" and
      v.asExpr() = alloc.getArgument(0).getFullyConverted()
    )
  }

  override predicate isSink(DataFlow::Node sink) {
    exists(FunctionCall dealloc |
      dealloc.getTarget().getName() = "pthread_mutex_destroy" and
      sink.asExpr() = dealloc.getArgument(0).getFullyConverted()
    )
  }
}

from Config cfg, DataFlow::Node src, DataFlow::Node sink
where cfg.hasFlow(src, sink) and src.getLocation() != sink.getLocation()
select src, sink
```

Listing A.11: Mutex Dataflow to Find Correct Usage of Mutexes

A.2.2 Verifiation of Tainted Length Variable Dataflow

```
import cpp
import semmle.code.cpp.dataflow.TaintTracking

class BufferCall extends FunctionCall {
    BufferCall() {
    this.getTarget().getName().matches("isc_buffer_get%")
    or     this.getTarget().getName().matches("isc_buffer_pee%")
    or     this.getTarget().getName().matches("%read%")
    or     this.getTarget().getName().matches("%ntohs%")
    or     this.getTarget().getName().matches("%bswap%")
```

```
  }

  Expr getVal() {
    //result = this.getArgument(1)
    result = this.getAnArgument()
}
}


class MemCall extends FunctionCall {
    MemCall() {
      this.getTarget().hasName("memmove")
      or this.getTarget().hasName("memcpy")
      or this.getTarget().hasName("strncpy")
      or this.getTarget().hasName("strcpy")
    }

    Expr getFileDescriptor() {
      result = this.getArgument(0)
    }
  }

class Config extends TaintTracking::Configuration {
  Config() { this = "MemoryTaintFlow" }

  override predicate isSource(DataFlow::Node source) {
    exists(BufferCall call |
      source.asExpr() = call.getVal()
    )
  }

  override predicate isSink(DataFlow::Node sink) {
    exists(MemCall call |
      sink.asExpr() = call.getArgument(2) // or whichever argument you're interested in
    )
  }
}

from Config cfg, DataFlow::PathNode source, DataFlow::PathNode sink
where cfg.hasFlowPath(source, sink)
select sink.getNode(), source, source.getNode().asExpr().getEnclosingElement(), sink,
↪  sink.getNode().asExpr().getActualType(), source.getNode().asExpr().getActualType()
```

Listing A.12: Tainted Operation Length Value Dataflow

A.2.3 Tainted Printf Function Family Dataflows

```
import cpp
import semmle.code.cpp.dataflow.TaintTracking

class BufferCall extends FunctionCall {
    BufferCall() {
    this.getTarget().getName().matches("isc_buffer_get%")
    or     this.getTarget().getName().matches("isc_buffer_pee%")
    or     this.getTarget().getName().matches("%read%")
    or     this.getTarget().getName().matches("%ntohs%")
    or     this.getTarget().getName().matches("%bswap%")
  }

  Expr getVal() {
    //result = this.getArgument(1)
    result = this.getAnArgument()
}
}


class FormatCall extends FunctionCall {
    FormatCall() {
        this.getTarget().getName().matches("%printf%")
        and not this.getTarget().getName().matches("printf")
    }
    Expr getFormatString() {
            result = this.getArgument(1)
    }
  }

class Config extends TaintTracking::Configuration {
  Config() { this = "PrintfTaintFlow" }

  override predicate isSource(DataFlow::Node source) {
    exists(BufferCall call |
      source.asExpr() = call.getVal()
    )
  }

  override predicate isSink(DataFlow::Node sink) {
    exists(FormatCall call |
      sink.asExpr() = call.getFormatString() // or whichever argument you're interested in
    )
  }
}

from Config cfg, DataFlow::PathNode source, DataFlow::PathNode sink
where cfg.hasFlowPath(source, sink)
select sink.getNode(), source, source.getNode().asExpr().getEnclosingElement(), sink,
↪  sink.getNode().asExpr().getActualType(), source.getNode().asExpr().getActualType()
```

Listing A.13: Tainted Printf Function Family Dataflow

A.2.4 Tainted ISC Memory Wrapper Dataflows

```
import cpp
import semmle.code.cpp.dataflow.DataFlow

class Config extends DataFlow::Configuration {
  Config() { this = "GetAndNotPutFlow" }
  override predicate isSource(DataFlow::Node source) {
    exists(FunctionCall alloc, Assignment a, Variable v |
      (alloc.getTarget().getName() = "isc__mem_get" or alloc.getTarget().getName() =
      ↪  "isc__mem_getx") and
      a.getRValue() = alloc and
      v = a.getLValue().(VariableAccess).getTarget() and
      source.asExpr() = v.getAnAccess()
    )
  }


  override predicate isSink(DataFlow::Node sink) {
    exists(FunctionCall dealloc |
      (dealloc.getTarget().getName() = "mem_put" or dealloc.getTarget().getName() = "abort" or
      ↪  dealloc.getTarget().getName() = "ISC_LIST_APPEND")
       and
      sink.asExpr() = dealloc.getArgument(1)
    )
  }
}

from Config cfg, FunctionCall alloc, Assignment a, Variable v
where
  (alloc.getTarget().getName() = "isc__mem_get" or alloc.getTarget().getName() = "isc__mem_getx")
  ↪  and
  a.getRValue() = alloc and
  v = a.getLValue().(VariableAccess).getTarget()
    and not exists(DataFlow::Node src, DataFlow::Node sink |
    src.asExpr() = v.getAnAccess() and cfg.hasFlow(src, sink)
  )
select alloc, v, alloc.getLocation(), v.getLocation()
```

Listing A.14: Tainted ISC Memory Wrapper Dataflow

A.2.5 Query to Find Unsafe RCU Dereference Dataflows

```
import cpp
import semmle.code.cpp.dataflow.DataFlow

class RcuDereferenceFunction extends Function {
  RcuDereferenceFunction() { this.getName().matches("%r_deref%") }
    // TODO this is hackish since it overfits, but there
    // are potentially lots of wrappers for rcu_dereference
}

class Config extends DataFlow::Configuration {
  Config() { this = "RCUDereferenceFuzzyFlow" }

  override predicate isSource(DataFlow::Node source) {
    exists(Call call, RcuDereferenceFunction func |
      call.getTarget() = func and
      source.asExpr() = call
    )
  }

  override predicate isSink(DataFlow::Node sink) {
    exists(VariableAccess va | va = sink.asExpr())
  }
}

from Config cfg, DataFlow::Node source, DataFlow::Node sink
where cfg.hasFlow(source, sink)
select source, sink, "Variable influenced by a call to rcu_dereference() or a similar function"
```

Listing A.15: RCU Dereference Dataflows (Overfitting)

A.2.6 Query to Find Second Order Recursive Calls

```
import cpp

from Function f1, Function f2, Call call1, Call call2
where call1.getEnclosingFunction() = f1 and
      call1.getTarget() = f2 and
      call2.getEnclosingFunction() = f2 and
      call2.getTarget() = f1
select call1, call2, call1.getLocation()
```

Listing A.16: Second Order Recursive Calls: A()->B()->A()

A.2.7 Query (Path-Query) to Find Locks Held during While And For Loops

```
/**
* @kind path-problem
*/

import cpp
import semmle.code.cpp.dataflow.TaintTracking

class LockCall extends FunctionCall {
    LockCall() {
    this.getTarget().getName().matches("%\\_lock%")
  }

  Expr getVal() {
    //result = this.getArgument(1)
    result = this.getAnArgument()
}
}


class UnlockCall extends FunctionCall {
    UnlockCall() {
      this.getTarget().getName().matches("%\\_unlock%")
    }

    Expr getLockArg() {
      result = this.getArgument(0)
    }
  }

class Config extends TaintTracking::Configuration {
  Config() { this = "LockTaintFlow" }

  override predicate isSource(DataFlow::Node source) {
    exists(LockCall call |
      source.asExpr() = call.getAnArgument().getFullyConverted()
      or source.asExpr() = call.getAnArgument()
    )
  }

  override predicate isSink(DataFlow::Node sink) {
    exists(UnlockCall call |
      sink.asExpr() = call.getAnArgument().getFullyConverted()
      ↪  // or whichever argument you're interested in
      or sink.asExpr() = call.getAnArgument() // or whichever argument you're interested in
      )
  }
}

// UNUSED: this recursive one may allow to filter out paths that
```

```
  // have an unlock before the while, but it is very slow
  predicate hasUnlockOnPath(BasicBlock start, BasicBlock end) {
    // Base case: the start block is a block of a ForStmt
    start.getANode() instanceof UnlockCall
    or
    // Recursive case: there exists a block on the path from
    // start to end that is a block of a ForStmt
    exists(BasicBlock mid |
      start.getASuccessor() = mid and
      mid != end
      and hasUnlockOnPath(mid, end)
    )
  }



  query predicate edges(BasicBlock a, BasicBlock b) {
    b = a.getASuccessor() and
    not a.getANode() instanceof UnlockCall and
    not (a.getANode() instanceof WhileStmt or a.getANode() instanceof ForStmt) // and
  }

  from Config cfg, LockCall start, WhileStmt end, ControlFlowNode entryPoint
  where
  entryPoint = start.getEnclosingBlock() and
  edges+(entryPoint.getBasicBlock(), end.getBasicBlock())
  select end, entryPoint.getBasicBlock(), end.getBasicBlock(), "Lock taint" +
  ↪  entryPoint.toString()
```

Listing A.17: Query Finding Paths Between Lock->While, Lock->For, and Lock->Unlock