LiteDNS: Out-of-Bounds Read in DNS Name Parsing Leads to Denial of Service (DoS)

Written by : Antonius (w1sdom)
Web : www.bluedragonsec.com
Github : https://github.com/bluedragonsecurity
Vulnerability discovered by : Antonius
Date of Discovery : March 6 2026

LiteDNS is a lightweight, C-based DNS server that supports basic DNS query resolution for A, AAAA, CNAME, NS, MX, and TXT records. This dns server runs on Linux platform.

The project url is at https://github.com/TanishqNanavati/LiteDNS

There is a remotely exploited vulnerability is this dns server that can causes a denial of service.

Vulnerability has been tested on this environments :

Tested Environment

- Operating System: CentOS Stream 9
- Architecture: x86_64
- Kernel: Linux `5.14.0-635.el9.x86_64`
- Working Directory During Testing: `/var/root/LiteDNS`
- Build Configuration: AddressSanitizer-enabled build

Here is the asan log :

```
Query from 127.0.0.1:52109 (8 bytes)
Decoded Buffer :
Response sent (17 bytes)

Query from 127.0.0.1:58730 (8 bytes)
Decoded Buffer :
Response sent (17 bytes)

Query from 127.0.0.1:60533 (4 bytes)
Decoded Buffer :
Response sent (17 bytes)

Query from 127.0.0.1:47818 (8 bytes)
Decoded Buffer :
Response sent (17 bytes)

Query from 127.0.0.1:53626 (16 bytes)
AddressSanitizer:DEADLYSIGNAL
=================================================================
==1021829==ERROR: AddressSanitizer: SEGV on unknown address 0x10007fff82ae (pc 0x000
==1021829==The signal is caused by a READ memory access.
    #0 0x20348f in decode_domain src/dns_parser.c:9
    #1 0x203e87 in parse_question src/dns_parser.c:57
    #2 0x2060bd in build_response src/dns_response.c:75
    #3 0x2066d2 in main /var/root/LiteDNS/main.c:53
    #4 0x7ffff662a60f in __libc_start_call_main (/lib64/libc.so.6+0x2a60f)
    #5 0x7ffff662a6bf in __libc_start_main@GLIBC_2.2.5 (/lib64/libc.so.6+0x2a6bf)
    #6 0x203354 in _start (/var/root/LiteDNS/dns-server+0x203354)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV src/dns_parser.c:9 in decode_domain
==1021829==ABORTING
```

Vulnerability discovered during fuzzing. Here is the last dns packet that causes crash :

```
sh-5.1# xxd /tmp/last_packet.dns
00000000: 1234 0100 0001 0000 0000 0000 0241 42ff  .4...........AB.
sh-5.1#
```

Packet in hex : 12 34 01 00 00 01 00 00 00 00 00 00 02 41 42 FF

This packet is **16 bytes total**. It is structured as:

12-byte DNS header

4-byte beginning of the Question section

So the layout is:

```
[ DNS Header: 12 bytes ]          [ Question section begins: 4 bytes ]

12 34 01 00 00 01 00 00 00 00 00 00          02 41 42 FF
```

A valid DNS question must contain:

- a complete **QNAME**
- a **QTYPE** field (2 bytes)
- a **QCLASS** field (2 bytes)

This packet does not contain a complete question, it's a truncated dns query. It has only:

- a header
- the start of a QNAME
- then an incomplete trailing byte that looks like the beginning of a compression pointer

The Compression Pointer

After the label `"AB" (the byte sequence : 0x41 and 0x42)`, a valid DNS name encoding must continue with an End of Name (0x00) or a Compression Pointer.

When the top 2 bits sets to 1 it means a compression pointer. The

binary of 0xff is 11111111, so it meets the criteria of a compression pointer since the top 2 bits sets to 1 : 11xxxxxx

Why this Compression Pointer Triggers a Denial of Service on LiteDNS ?

Because if `FF` is treated as a compression pointer start, the parser expects:

- current byte = first pointer byte
- next byte = second pointer byte

But the packet ends immediately after 0xFF. What happens to parser in the background ?

Step 1

It starts reading the question name at offset 12 (0x02)

```
12 34 01 00 00 01 00 00 00 00 00 00 02 41 42 FF
```

so the parser determines the label length is 2.

Step 2

The parser reads the next 2 bytes sequences : 41 and 42, so the label is “AB”

Step 3

Now the offset reaches FF, since FF is a compression pointer, the parser treats it as a pointer byte.

Step 4

The parser then tries to read the next byte in order to complete the pointer, unfortunetly there is no next byte inside the dns packet hence it causes the parser to read past the received packet boundary.

Step 6

After that, the derived offset may become attacker-influenced garbage, and the parser may continue reading from an invalid location, eventually causing a crash.

POC

https://github.com/bluedragonsecurity/LiteDNS_out_of_bounds_read_vulnerability/blob/main/poc.c

References

https://github.com/bluedragonsecurity/LiteDNS_out_of_bounds_read_vulnerability

https://medium.com/@w1sdom/litedns-out-of-bounds-read-in-dns-name-parsing-leads-to-denial-of-service-e4a41a7efa49