Some dereference failures found during software vulnerabilities investigation on linenoise and lua lib dependences

Edit advisory

Closed | Moderate | **janislley** opened **GHSA-jfp8-58rm-3f8m** on Jul 11 · 3 comments

Package

linenoise and lua

Affected versions

7.0.11

Patched versions

None

janislley opened on Jul 11 • edited

Description

Hello.

I am performing some tests on redis as a security investigative report.
During the tests, potential software vulnerabilities were found.
To identify this kind of vulnerabilities it was used the tool ESBMC-WR: https://github.com/janislley/esbmc-wr

More about the tool: https://arxiv.org/pdf/2102.02368.pdf

Tests were performed in the latest redis version.
Please let me know if you need more reports or information regarding the tests.

For this report, issues were found on linenoise lib dependence.
Check complete details:

Issue 01: dereference failure: invalid pointer on linenoise.c

```
########################################
[FILE] deps/linenoise/linenoise.c
[ARGS] ['--unwind', '1', '--no-unwinding-assertions']
[FUNCTION] refreshSingleLine
########################################
```

State 1 file linenoise.c line 523 function refreshSingleLine thread 0
Violated property:
file linenoise.c line 523 function refreshSingleLine
dereference failure: invalid pointer

VERIFICATION FAILED

Issue 02: dereference failure: invalid pointer on linenoise.c

```
########################################
[FILE] deps/linenoise/linenoise.c
[ARGS] ['--unwind', '1', '--no-unwinding-assertions']
[FUNCTION] refreshMultiLine
########################################
```

State 8 file linenoise.c line 569 function refreshMultiLine thread 0
rows = 47869973 (00000010 11011010 01110000 00010101)

State 14 file linenoise.c line 570 function refreshMultiLine thread 0
rpos = 62 (00000000 00000000 00000000 00111110)

State 16 file linenoise.c line 573 function refreshMultiLine thread 0
old_rows = 0 (00000000 00000000 00000000 00000000)

State 19 file linenoise.c line 474 function abInit thread 0
ab->b = 0

State 20 file linenoise.c line 475 function abInit thread 0
ab->len = 0 (00000000 00000000 00000000 00000000)

State 23 file linenoise.c line 479 function abAppend thread 0
new = INVALID4294967295

State 24 file linenoise.c line 483 function abAppend thread 0
ab->b = INVALID4294967295

State 25 file linenoise.c line 484 function abAppend thread 0
ab->len = ab.len

State 30 file linenoise.c line 479 function abAppend thread 0
new = 0

State 34 file linenoise.c line 479 function abAppend thread 0
new = INVALID4294967295

State 36 file string.c line 264 function memcpy thread 0
Violated property:
file string.c line 264 function memcpy
dereference failure: invalid pointer

VERIFICATION FAILED

Issue 03: dereference failure: NULL pointer on linenoise.c

###
[FILE] deps/linenoise/linenoise.c
[ARGS] ['--unwind', '1', '--no-unwinding-assertions']
[FUNCTION] completeLine
###

State 2 file linenoise.c line 372 function completeLine thread 0

Violated property:
file linenoise.c line 372 function completeLine
dereference failure: NULL pointer

VERIFICATION FAILED

Issue 04: dereference failure: invalid pointer freed on linenoise.c

###
[FILE] deps/linenoise/linenoise.c
[ARGS] ['--unwind', '1', '--no-unwinding-assertions']
[FUNCTION] linenoiseFree
###

State 1 file linenoise.c line 1101 function linenoiseFree thread 0

Violated property:
file linenoise.c line 1101 function linenoiseFree
dereference failure: invalid pointer freed

VERIFICATION FAILED

Issue 05: dereference failure: Access to object out of bounds on deps/lua/src/lstrlib.c

```
########################################
[FILE] deps/lua/src/lstrlib.c
[ARGS] ['--unwind', '1', '--no-unwinding-assertions']
[FUNCTION] str_match
########################################

State 1 file lstrlib.c line 497 function str_find_aux thread 0
s = &ms

State 2 file lstrlib.c line 498 function str_find_aux thread 0
p = INVALID16422

State 3 file lstrlib.c line 499 function str_find_aux thread 0
init = 9220539694901624831 (01111111 11110101 11101111 11111111 11111111 11111111 11111111 11111111)

State 5 file lstrlib.c line 514 function str_find_aux thread 0
anchor = 0 (00000000 00000000 00000000 00000000)

State 6 file lstrlib.c line 515 function str_find_aux thread 0
s1 = &ms + 9220539694901624831

State 7 file lstrlib.c line 516 function str_find_aux thread 0
ms.L = invalid-object

State 8 file lstrlib.c line 517 function str_find_aux thread 0
ms.src_init = &ms

State 9 file lstrlib.c line 518 function str_find_aux thread 0
ms.src_end = &ms + 9221133989023534562

State 10 file lstrlib.c line 521 function str_find_aux thread 0
ms.level = 0 (00000000 00000000 00000000 00000000)

State 18 file lstrlib.c line 187 function check_capture thread 0
l = 38 (00000000 00000000 00000000 00100110)

State 23 file lstrlib.c line 357 function match_capture thread 0

Violated property:
file lstrlib.c line 357 function match_capture
dereference failure: Access to object out of bounds

VERIFICATION FAILED
```

janislley added as a collaborator on Jul 11

janislley was credited as a reporter on Jul 11

janislley accepted credit on Jul 11

Decline credit

yossigo commented on Jul 11

@**janislley** My comment on your other report applies here as well.

janislley commented on Jul 11

Hello @**yossigo**
Sure, let me explain in detail:

Issue 01 - In the refreshSingleLine function, there is no initial value verification for `l->prompt` . So, in an extreme case, it can return on an invalid pointer.

To prevent this, it's a good practice to always ensure that the pointers you're dereferencing are valid. This could involve checking whether l and l->prompt are not null before the strlen(l->prompt) call, and ensuring that l->prompt points to a null-terminated string.

Issue 02 - I confirmed it as a false positive.

Issue 03 - This issue might occurs on line `completionCallback(ls->buf,&lc)` .

Here, `ls` is a pointer to a `linenoiseState` structure and `ls->buf` is being passed as a parameter to the `completionCallback` function. So, if `ls` or `ls->buf` is NULL, you would get a null pointer dereference.

This means that there are two potential reasons for null pointer dereference:

- If ls is NULL, `ls->buf` will try to dereference a null pointer.
- If ls is not NULL, but `ls->buf` is NULL, and the `completionCallback` function tries to dereference `ls->buf` without first checking that it's not NULL, then that would also lead to a null pointer dereference.

If the static code analysis tool is telling you that there is a potential null pointer dereference, then it's very likely that one of these situations could occur based on the current code and how the function is being used in your codebase. To handle this, you should add null-checks before the `completionCallback(ls->buf,&lc)` line to make sure neither ls nor `ls->buf` is NULL.

Issue 04 - The provided code snippet free(ptr) on its own does not necessarily indicate a problem. It is a false positive.

janislley changed the title ~~**Some dereference failures found during software vulnerabilities investigation on linenoise lib dependence**~~ **Some dereference failures found during software vulnerabilities investigation on linenoise and lua lib dependences** on Jul 11

oranagra commented on Jul 11

@janislley

the point of this advisory is report a bug that can result in a security issue, not report bad coding practices.
if you or that tool can provide a code path that could lead to referencing a NULL or an uninitialized pointer, we would like know what is the code path (without that, the analysis report isn't very useful).
but the fact that the function doesn't validate the input pointer is not NULL doesn't indicate a bug.
also, checking for NULL can obviously be ineffective since it could also be a pointer that looks valid but points to an invalid memory.
p.s. IIRC, strlen doesn't check for null pointer either.

yossigo closed this on Sep 2

Severity

Moderate 5.9 / 10

CVSS base metrics

Attack vector	Local
Privileges required	Low
User interaction	None
Scope	None
Confidentiality	Unchanged
Integrity	Low
Availability	Low
Availability	Low

e severe if no privileges are required

CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L

CVE ID

No known CVE

Weaknesses

CWE-476 CWE-822

Credits

janislley Reporter

Collaborators

Only the following users and teams can see and collaborate on this advisory:

redis owners

janislley Author Remove

Publishers

Only the following users and teams can publish this advisory:

redis owners