

[New issue](#)[Jump to bottom](#)

# Global Buffer overflow in gettoken at Main.c (Ver 1.91) #75

Closed Halcy0nic opened this issue on Oct 15 · 12 comments

Halcy0nic commented on Oct 15

Hi @sasagawa888,

I pulled down the most recent version of nprolog (Ver 1.91) and ran it through my fuzz tests. It looks like there is a global buffer overflow in gettoken at Main.c when you tell NPL to run a file in script mode.

```
└─$ ./npl -s output/default/crashes/id:000001,sig:11,src:000000+000002,time:157668,execs:302,op:splice,rep:16
zsh: segmentation fault  ./npl -s
```

I have attached most of the crash files for reproduction. If you compile the project with [AddressSanitizer](#) it can also detect the global overflow:

## Makefile

```
CC      = gcc
LIBS    = -lm -ldl -fsanitize=address

LIBSRASPI = -lm -ldl -lwiringPi -fsanitize=address
INCS      =
CFLAGS    = $(INCS) -Wall -O3 -fsanitize=address
DEST      = /usr/local/bin
```

## Running NPL in script mode

```

$ ./npl -s output/default/crashes/id:000001,sig:11,src:000000+000002,time:157668,execs:302,op:splice,rep:16
==4170466==ERROR: AddressSanitizer: global-buffer-overflow on address 0x55c6d7cc04d0 at pc 0x55c6d7c3b925 bp 0x7ffd6a8a5170 sp 0x7ffd6a8a5168
WRITE of size 1 at 0x55c6d7cc04d0 thread T0
#0 0x55c6d7c3b924 in gettoken (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x24924)
#1 0x55c6d7c40924 in parser (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x29924)
#2 0x55c6d7c4c4ea in b_consult (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x354ea)
#3 0x55c6d7c2cd8e in main (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x15d8e)
#4 0x7f68357e67fc in __libc_start_main ../csu/libc-start.c:332
#5 0x55c6d7c2d2c9 in _start (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x162c9)

0x55c6d7cc04d0 is located 48 bytes to the left of global variable 'record_pt' defined in 'main.c:24:5' (0x55c6d7cc0500) of size 4
0x55c6d7cc04d0 is located 0 bytes to the right of global variable 'stok' defined in 'main.c:27:7' (0x55c6d7cc03c0) of size 272
SUMMARY: AddressSanitizer: global-buffer-overflow (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x24924) in gettoken
Shadow bytes around the buggy address:
 0x0ab95af90040: 04 f9 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9 f9
 0x0ab95af90050: 04 f9 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9 f9
 0x0ab95af90060: 04 f9 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9 f9
 0x0ab95af90070: 04 f9 f9 f9 f9 f9 f9 00 00 00 00 00 00 00 00
 0x0ab95af90080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0ab95af90090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0ab95af900a0: 04 f9 f9 f9 f9 f9 f9 00 00 00 00 00 00 00 00
 0x0ab95af900b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0ab95af900c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0ab95af900d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0ab95af900e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack after return: f5
Stack use after scope: f8
Global redzone: f9
Global init order: f6
Poisoned by user: f7
Container overflow: fc
Array cookie: ac
Intra object redzone: bb
ASan internal: fe
Left alloca redzone: ca
Right alloca redzone: cb
Shadow gap: cc
==4170466==ABORTING

```

[crash.zip](#)

sasagawa888 commented 10 days ago

Owner

Sorry for the late reply. I missed it. I will consider

sasagawa888 commented 10 days ago

Owner

I changed the game called lights-out included in hiroi.pl for script mode and tried it. It worked fine. My environment is Linux-MINT. Which Prolog file did you run? please tell me in detail.

```
npl -s tests/hiroi.pl
```

```
11000
```

```
11011
```

```
00111
```

```
01110
```

```
01101
```

```
10110
```

```
01110
```

```
11100
```

```
11011
```

```
00011
```

```
01101
01110
00111
11011
11000

00011
11011
11100
01110
10110
sasagawa@sasagawa-Diginnos-PC:~/nprolog$
```

**Halcy0nic** commented 10 days ago • edited ▼

Author

Hi there! In the original message I attached a file for reproduction (named crash.zip). If you unzip this file you will find a script to run using NPL. After running this file in script mode, NPL will segfault. Compiling the project with address sanitizer (above) and running the attached NPL script will indicate where the global buffer overflow takes place.

I tested this on Ubuntu 18, 20, 16.04, and Debian. All 64 bit operating systems

**sasagawa888** commented 8 days ago

Owner

Thank you for your reply. I will try.

**sasagawa888** commented 8 days ago

Owner

Hi Halcy0nic.

I downloaded crash.zip and unzipped it. However, the contents could not be read. Could you please send the data again?

**Halcy0nic** commented 8 days ago • edited ▼

Author

Hi @sasagawa888!

I just pulled down the file again and it seems like it is working correctly. If you unzip the file it should have a folder named report and another folder named vuln underneath it. The contents of the script may not be human readable (meaning the bytes of the file have been modified) but can still be executed by running the following:

```
np1 -s [any file in the vuln directory]
```

Some of the output isn't human readable, because it was generated by the fuzzer I am using (AFL), and the file bytes have been modified.

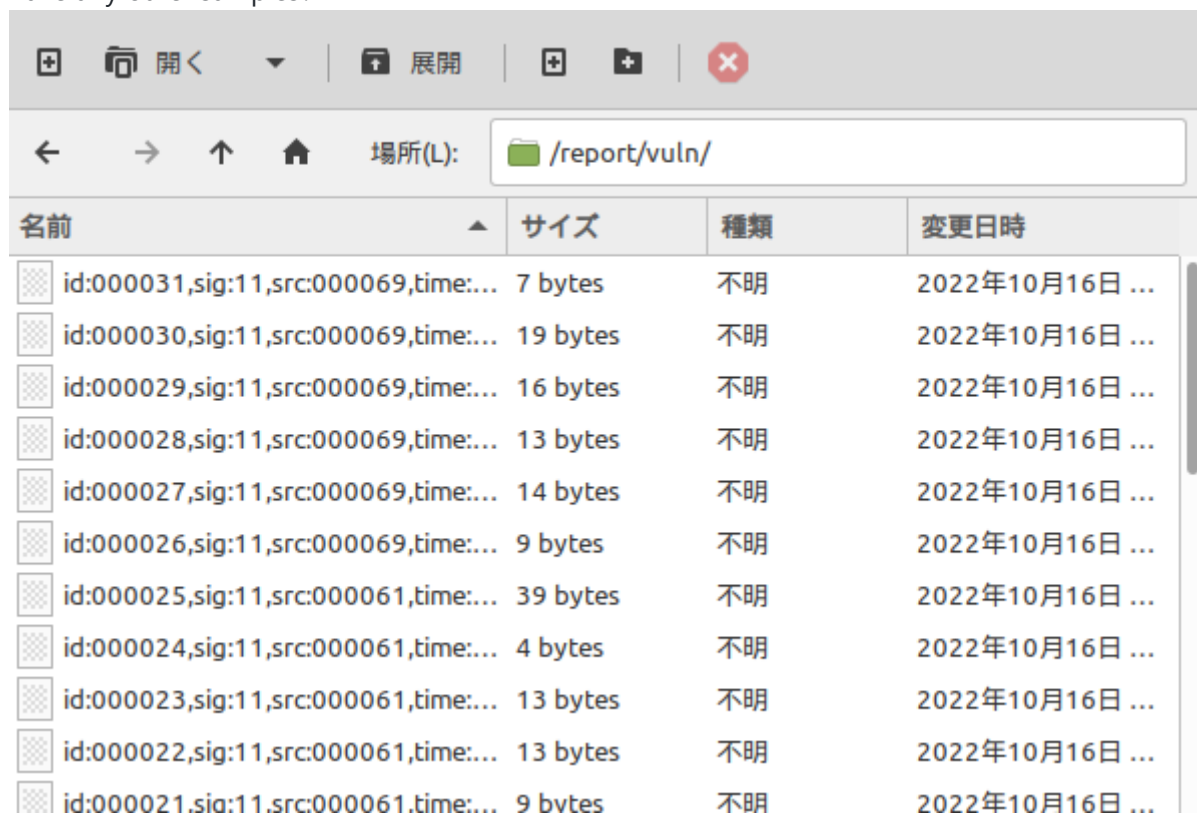
Are you able to see all of the files that start with the pattern id:000... in the report/vuln directory?

If not I will upload another sample.

sasagawa888 commented 8 days ago

Owner

When I unzipped the zip file, it looked like this: Cannot read inside the file. File name is also unknown. Do you have any other samples?



名前	サイズ	種類	変更日時
id:000031,sig:11,src:000069,time:...	7 bytes	不明	2022年10月16日 ...
id:000030,sig:11,src:000069,time:...	19 bytes	不明	2022年10月16日 ...
id:000029,sig:11,src:000069,time:...	16 bytes	不明	2022年10月16日 ...
id:000028,sig:11,src:000069,time:...	13 bytes	不明	2022年10月16日 ...
id:000027,sig:11,src:000069,time:...	14 bytes	不明	2022年10月16日 ...
id:000026,sig:11,src:000069,time:...	9 bytes	不明	2022年10月16日 ...
id:000025,sig:11,src:000061,time:...	39 bytes	不明	2022年10月16日 ...
id:000024,sig:11,src:000061,time:...	4 bytes	不明	2022年10月16日 ...
id:000023,sig:11,src:000061,time:...	13 bytes	不明	2022年10月16日 ...
id:000022,sig:11,src:000061,time:...	13 bytes	不明	2022年10月16日 ...
id:000021,sig:11,src:000061,time:...	9 bytes	不明	2022年10月16日 ...

Halcy0nic commented 8 days ago

Author

@sasagawa888

Here are a few new examples. If you unzip the 'testcases.zip' file:

[testcases.zip](#)

You will see a folder named 'testcases' with the following contents inside:

- test1.pl

- test2.pl
- test3.pl

## Viewing and executing test1.pl without AddressSanitizer

```
$ cat testcases/test1.pl
...
```

```
$ ./npl -s ./testcases/test1.pl
zsh: segmentation fault ./npl -s ./testcases/test1.pl
```

## Executing test1.pl with AddressSanitizer

```
==8598==ERROR: AddressSanitizer: global-buffer-overflow on address 0x557313a9c4d0 at pc 0x557313a17925 bp 0x7ffe376e6b40 sp 0x7ffe376e6b38
WRITE of size 1 at 0x557313a9c4d0 thread T0
#0 0x557313a17924 in gettoken (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x24924)
#1 0x557313a1c924 in parser (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x29924)
#2 0x557313a284ea in b_consult (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x354ea)
#3 0x557313a08d8e in main (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x15d8e)
#4 0x7fc6360b2209 in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
#5 0x7fc6360b22bb in __libc_start_main_impl ../csu/libc-start.c:389
#6 0x557313a092c9 in _start (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x162c9)

0x557313a9c4d0 is located 48 bytes to the left of global variable 'record_pt' defined in 'main.c:24:5' (0x557313a9c500) of size 4
0x557313a9c4d0 is located 0 bytes to the right of global variable 'stok' defined in 'main.c:27:7' (0x557313a9c3c0) of size 272
SUMMARY: AddressSanitizer: global-buffer-overflow (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x24924) in gettoken
Shadow bytes around the buggy address:
 0x0aaee274b840: 04 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9
 0x0aaee274b850: 04 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9
 0x0aaee274b860: 04 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9
 0x0aaee274b870: 04 f9 f9 f9 f9 f9 00 00 00 00 00 00 00
 0x0aaee274b880: 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0aaee274b890: 00 00 00 00 00 00 00 00 00[f9]f9 f9 f9 f9
 0x0aaee274b8a0: 04 f9 f9 f9 f9 f9 00 00 00 00 00 00 00
 0x0aaee274b8b0: 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0aaee274b8c0: 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0aaee274b8d0: 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0aaee274b8e0: 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack after return: f5
Stack use after scope: f8
Global redzone: f9
Global init order: f6
Poisoned by user: f7
Container overflow: fc
Array cookie: ac
Intra object redzone: bb
ASan internal: fe
Left alloca redzone: ca
Right alloca redzone: cb
Shadow gap: cc
==8598==ABORTING
```

## Viewing and executing test2.pl without AddressSanitizer

```
$ cat testcases/test2.pl
:-
$1 is N+u is N+u
```

```
$ ./npl -s ./testcases/test2.pl
zsh: segmentation fault ./npl -s ./testcases/test2.pl
```



## Executing test2.pl with AddressSanitizer

```
└─$ ./npl -s ./testcases/test2.pl
==8165==ERROR: AddressSanitizer: global-buffer-overflow on address 0x55ecedc5f4d0 at pc 0x55ecedbdb53d bp 0x7fff69e72a30 sp 0x7fff69e72a28
WRITE of size 1 at 0x55ecedc5f4d0 thread T0
#0 0x55ecedbdb53c in gettoken (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x2553c)
#1 0x55ecedbdf924 in parser (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x29924)
#2 0x55ecedbe0d46 in parser (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x2ad46)
#3 0x55ecedbeb4ea in b_consult (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x354ea)
#4 0x55ecedbcb8e in main (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x15d8e)
#5 0x7fa75efbb209 in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
#6 0x7fa75efbb2bb in __libc_start_main_impl ../csu/libc-start.c:389
#7 0x55ecedbcc2c9 in _start (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x162c9)

0x55ecedc5f4d0 is located 48 bytes to the left of global variable 'record_pt' defined in 'main.c:24:5' (0x55ecedc5f500) of size 4
0x55ecedc5f4d0 is located 0 bytes to the right of global variable 'stok' defined in 'main.c:27:7' (0x55ecedc5f3c0) of size 272
SUMMARY: AddressSanitizer: global-buffer-overflow (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x2553c) in gettoken
Shadow bytes around the buggy address:
 0x0abe1db83e40: 04 f9 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9 f9
 0x0abe1db83e50: 04 f9 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9 f9
 0x0abe1db83e60: 04 f9 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9 f9
 0x0abe1db83e70: 04 f9 f9 f9 f9 f9 f9 00 00 00 00 00 00 00 00
 0x0abe1db83e80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0abe1db83e90: 00 00 00 00 00 00 00 00 00 00 00[f9]f9 f9 f9 f9
 0x0abe1db83ea0: 04 f9 f9 f9 f9 f9 f9 00 00 00 00 00 00 00 00
 0x0abe1db83eb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0abe1db83ec0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0abe1db83ed0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0abe1db83ee0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack after return: f5
Stack use after scope: f8
Global redzone: f9
Global init order: f6
Poisoned by user: f7
Container overflow: fc
Array cookie: ac
Intra object redzone: bb
ASan internal: fe
Left alloca redzone: ca
Right alloca redzone: cb
Shadow gap: cc
==8165==ABORTING
```

## Viewing and executing test3.pl without AddressSanitizer

```
└─$ cat testcases/test3.pl
:-
%1-
$1 :-:-
%.

1-
```

```
└─$ ./npl -s ./testcases/test3.pl
zsh: segmentation fault ./npl -s ./testcases/test3.pl
```

## Executing test3.pl with AddressSanitizer

```

$ ./npl -s ./testcases/test3.pl

==7530==ERROR: AddressSanitizer: global-buffer-overflow on address 0x5561e15f14d0 at pc 0x5561e156d53d bp 0x7fff78bacc70 sp 0x7fff78bacc68
WRITE of size 1 at 0x5561e15f14d0 thread T0
#0 0x5561e156d53c in gettoken (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x2553c)
#1 0x5561e1571924 in parser (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x29924)
#2 0x5561e1572d46 in parser (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x2ad46)
#3 0x5561e157d4ea in b_consult (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x354ea)
#4 0x5561e155dd8e in main (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x15d8e)
#5 0x7fe121641209 in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
#6 0x7fe1216412bb in __libc_start_main_impl ../csu/libc-start.c:389
#7 0x5561e155e2c9 in _start (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x162c9)

0x5561e15f14d0 is located 48 bytes to the left of global variable 'record_pt' defined in 'main.c:24:5' (0x5561e15f1500) of size 4
0x5561e15f14d0 is located 0 bytes to the right of global variable 'stok' defined in 'main.c:27:7' (0x5561e15f13c0) of size 272
SUMMARY: AddressSanitizer: global-buffer-overflow (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x2553c) in gettoken
Shadow bytes around the buggy address:
 0x0aacbc2b6240: 04 f9 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9 f9
 0x0aacbc2b6250: 04 f9 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9 f9
 0x0aacbc2b6260: 04 f9 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9 f9 f9
 0x0aacbc2b6270: 04 f9 f9 f9 f9 f9 f9 00 00 00 00 00 00 00 00
 0x0aacbc2b6280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0aacbc2b6290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0aacbc2b62a0: 04 f9 f9 f9 f9 f9 f9 00 00 00 00 00 00 00 00
 0x0aacbc2b62b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0aacbc2b62c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0aacbc2b62d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0aacbc2b62e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack after return: f5
Stack use after scope: f8
Global redzone: f9
Global init order: f6
Poisoned by user: f7
Container overflow: fc
Array cookie: ac
Intra object redzone: bb
ASan internal: fe
Left alloca redzone: ca
Right alloca redzone: cb
Shadow gap: cc
==7530==ABORTING

```

sasagawa888 commented 7 days ago

Owner

I tested.

An error occurred in test1.pl.

test2.pl Test3.pl becomes a segmentation fault. The reason is that \$ is a symbol representing a string and the terminating \$ is not given. I needed to detect file\_end and make it an error.

N-Prolog is ARITY-Prolog compatible. Not ISO-Prolog.

N-Prolog Ver 1.91

?- X = \$abcd\$.

X = \$abcd\$ .

yes

?-

```
sasagawa@DESKTOP-0D0L605:~/nprolog$ cat tests/test1.pl
```

```
...
```

```
sasagawa@DESKTOP-0D0L605:~/nprolog$ npl -s tests/test1.pl
```

```
Syntax error expected operator
```

```
around here line=1 column=0
```

```
?- halt.
```

```
Not callable ?- .halt
```

```
?- halt.
```

```
- good bye -
```

```
sasagawa@DESKTOP-0D0L605:~/nprolog$ cat tests/test2.pl
:-
$1 is N+u is N+u.
sasagawa@DESKTOP-0D0L605:~/nprolog$ npl -s tests/test2.pl
Segmentation fault (core dumped)
sasagawa@DESKTOP-0D0L605:~/nprolog$ cat tests/test3.pl
:-
%1-
$1 :-:-
%.
sasagawa@DESKTOP-0D0L605:~/nprolog$ npl -s tests/test3.pl
Segmentation fault (core dumped)
sasagawa@DESKTOP-0D0L605:~/nprolog$
```

Halcy0nic commented 7 days ago

Author

Glad you were able to reproduce.

Just a heads up, it will segfault and cause a global buffer overflow even without the '\$' character. Here is an example (attached):

[test4.zip](#)

```
$ cat testcases/test4.pl
ffAon(Nf)'
```

```
$ ./npl -s testcases/test4.pl
zsh: segmentation fault ./npl -s testcases/test4.pl
```

## Executing with AddressSanitizer



```

$ ./npl -s testcases/test4.pl
==105440==ERROR: AddressSanitizer: global-buffer-overflow on address 0x55a1ee51c4d0 at pc 0x55a1ee497925 bp 0x7ffce9127700 sp 0x7ffce91276f8
WRITE of size 1 at 0x55a1ee51c4d0 thread T0
#0 0x55a1ee497924 in gettoken (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x24924)
#1 0x55a1ee49c924 in parser (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x29924)
#2 0x55a1ee4a84ea in b_consult (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x354ea)
#3 0x55a1ee488d8e in main (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x15d8e)
#4 0x7f33a8c2b209 in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
#5 0x7f33a8c2b2bb in __libc_start_main_impl ../csu/libc-start.c:389
#6 0x55a1ee4892c9 in _start (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x162c9)

0x55a1ee51c4d0 is located 48 bytes to the left of global variable 'record_pt' defined in 'main.c:24:5' (0x55a1ee51c500) of size 4
0x55a1ee51c4d0 is located 0 bytes to the right of global variable 'stok' defined in 'main.c:27:7' (0x55a1ee51c3c0) of size 272
SUMMARY: AddressSanitizer: global-buffer-overflow (/home/kali/projects/fuzzing/fuzz_targets/nprolog/npl+0x24924) in gettoken
Shadow bytes around the buggy address:
 0x0ab4bdc9b840: 04 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9
 0x0ab4bdc9b850: 04 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9
 0x0ab4bdc9b860: 04 f9 f9 f9 f9 f9 04 f9 f9 f9 f9 f9
 0x0ab4bdc9b870: 04 f9 f9 f9 f9 f9 00 00 00 00 00 00
 0x0ab4bdc9b880: 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0ab4bdc9b890: 00 00 00 00 00 00 00 00 00[f9]f9 f9 f9 f9
 0x0ab4bdc9b8a0: 04 f9 f9 f9 f9 f9 00 00 00 00 00 00
 0x0ab4bdc9b8b0: 00 00 00 00 00 00 00 00 00 00 00 00
 0x0ab4bdc9b8c0: 00 00 00 00 00 00 00 00 00 00 00 00
 0x0ab4bdc9b8d0: 00 00 00 00 00 00 00 00 00 00 00 00
 0x0ab4bdc9b8e0: 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack after return: f5
Stack use after scope: f8
Global redzone: f9
Global init order: f6
Poisoned by user: f7
Container overflow: fc
Array cookie: ac
Intra object redzone: bb
ASan internal: fe
Left alloca redzone: ca
Right alloca redzone: cb
Shadow gap: cc
==105440==ABORTING

```

sasagawa888 commented 7 days ago

Owner

Reproduced. The reason is that I didn't expect the case without the period. Finding file\_end should be an error.

```

sasagawa@sasagawa-Diginnos-PC:~/nprolog$ cat tests/test4.pl
ffAon(Nf)'sasagawa@sasagawa-Diginnos-PC:~/nprolog$ npl -s tests/test4.pl
Segmentation fault (core dumped)
sasagawa@sasagawa-Diginnos-PC:~/nprolog$

```

sasagawa888 commented 6 days ago

Owner

I fixed it. Please continue testing.

 sasagawa888 closed this as completed 5 days ago

Assignees

No one assigned

---

Labels

None yet

---

Projects

None yet

---

Milestone

No milestone

---

Development

No branches or pull requests

---

2 participants

