

PICOC Null Pointer Dereference Denial of Service

PICOC Suffers from a Denial of Service (CWE476) vulnerability as a result of a Null Pointer Dereference. Any project or library that uses Picoc also suffers from this issue. An example of this would be picoc-js (<https://www.npmjs.com/package/picoc-js>). As a result PICOC will immediately segfault.

Reproduction Steps

1. Create a file to be executed by the PICOC interpreter

```
$ touch vulncode
```

2. Add the following code to the file:

```
printf("Before Crash\n");  
**4%;  
printf("This code won't execute because of the crash\n");
```

3. Execute PICOC against the file:

```
$ ./picoc -s vulncode
```

4. You will receive a segfault and the program will crash. This is a result of a null pointer dereference that is not caught or handled by the interpreter. The vulnerable line of code can be seen below:

```
**4%;
```

Solution

Adding a few if statements that verify the pointer is not NULL before usage will solve this problem. You can find more information about this here:

https://owasp.org/www-community/vulnerabilities/Null_Dereference

To upload designs, you'll need to enable LFS and have an admin enable hashed storage. [More information](#)

Tasks  0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items  0

Link issues together to show that they're related or that one is blocking others. [Learn more.](#)

Activity



Halcy0nic @Halcy0nic · 4 months ago

Author

GDB Trace:

```

Starting program: /home/hack/projects/fuzzing/p100/p100 vulnCrash
Program received signal SIGSEGV, Segmentation fault.
VariableReferencePointer (PointerVal=0x020200, DerefVal=0xffffffffc00, DerefOffset=0xffffffffc0c, DerefType=0xffffffffc378, DerefIsValue=0xffffffffc00) at variable.c:319
319      DerefType  PointerValue  Typ  FromType
LEGEND: STACK | HEAP | CODE | DATA | ROM | XMM | REGISTER
RAX 0x0
RDX 0x0
RDI 0xffffffffc378 ← 0x0
RDX 0xffffffffc36c ← 0x020200000000
RDI 0xffffffffc36c ← 0x0
R12 0xffffffffc300 ← 0x0
R8 0xffffffffc300 ← 0x1
R9 0x0
R10 0xffffffffffff00
R11 0x20
R12 0x0000000000000001 ← aux  rbp, rbp
R13 0x0
R14 0x0
R15 0x0
RBP 0xffffffffc300 → 0xffffffffc300 → 0xffffffffc20 → 0xffffffffc00 → 0xffffffffc00 ← ...
RSP 0xffffffffc300 → 0xffffffffc300 → 0xffffffffc20 → 0xffffffffc00 → 0xffffffffc00 ← ...
RIP 0x0000000000000000 ← mov  rax, qword ptr [rax + 0x10]
~ 0x13f08 <VariableReferencePointer+00> mov  rax, qword ptr [rax + 0x10]
0x13f0A <VariableReferencePointer+00> mov  rax, qword ptr [rax + 0x10]
0x13f0B <VariableReferencePointer+05> mov  qword ptr [rax], rax
0x13f0B <VariableReferencePointer+05> rbp  qword ptr [rbp + 0x10], 0
0x13f0B <VariableReferencePointer+22> je  VariableReferencePointer+04 <VariableReferencePointer+0>
+
0x13f0c <VariableReferencePointer+0A> rbp  qword ptr [rbp + 0x10], 0
0x13f0c <VariableReferencePointer+0A> je  VariableReferencePointer+101 <VariableReferencePointer+10>
+
0x13f0d <VariableReferencePointer+101> mov  rax, qword ptr [rbp + 0]
0x13f0E <VariableReferencePointer+105> mov  rax, qword ptr [rax + 0]
0x13f0F <VariableReferencePointer+109> mov  rax, qword ptr [rax]
0x13f0F <VariableReferencePointer+112> rbp  rbp

```

Please [register](#) or [sign in](#) to reply