



360 核心安全技术博客

主页 Home

归档 Archive

分类 Category

关于 About

EOS Node Remote Code Execution Vulnerability --- EOS WASM Contract Function Table Array Out of Bounds

05月29, 2018

Vulnerability Credit

Yuki Chen of Qihoo 360 Vulcan Team Zhiniang Peng of Qihoo 360 Core Security

文章目录

- [Vulnerability Credit](#)
 - [Vulnerability Description](#)
 - [Vulnerability Reporting Timeline](#)
 - [Technical Detail of the Vulnerability](#)
 - [How to reproduce the vulnerability](#)
 - [Exploit the vulnerability to achieve Remote Code Execution](#)
 - [The Fixing of Vulnerability](#)

Vulnerability Description

We found and successfully exploit a buffer out-of-bounds write vulnerability in EOS when parsing a WASM file. To use this vulnerability, attacker could upload a malicious smart contract to the nodes server, after the contract get parsed by nodes server, the malicious payload could execute on the server and taken control of it. After taken control of the nodes server, attacker could then pack the malicious contract into new block and further control all nodes of the EOS network.

Vulnerability Reporting Timeline

2018-5-11 EOS Out-of-bound Write Vulnerability Found 2018-5-28 Full Exploit Demo of Compromise EOS

Super Node Completed 2018-5-28 Vulnerability Details Reported to Vendor 2018-5-29 Vendor Fixed the Vulnerability on Github and Closed the Issue 2018-5-29 Notices the Vendor the Fixing is not complete Some Telegram chats with Daniel Larimer: We trying to report the bug to him. He said they will not ship the EOS without fixing, and ask us send the report privately since some people are running public test nets

Chats Daniel Larimer last seen 3 hours ago May 28

Hey Daniel, I am the leader of Qihoo 360Vulcan Team, a famous security research team. We just found and exploited a critical bug on EOS producer node. The attacker can use it to remotely execute code on nodes.

We have successfully exploited bug on latest version.

How could we report details to EOS? Can we discuss through mail ? 23:43 ✓

May 29

We can also submit it to github, but that would make others see this issue, is this OK? 00:17 ✓

Hi 00:19

Yes, we won't ship with out it fixed 00:20

Or you can submit it to me here 00:21

Since people are running public test nets 00:21

He provided his mailbox and we send the report to him

Chats Daniel Larimer last seen 3 hours ago

OK can I submit to you by mail? 00:22 ✓

I think fixing would be easy 00:22 ✓

[REDACTED] mail.com 00:23

Cool 00:23 ✓

Will send you soon 00:23 ✓

Thanks. 00:23

| EOS WASM Contract Function Table Array Out of Bounds Write Vulnerabilityvvvvvvvvv|

qihoo 360

To: [REDACTED]@mail.com
Cc: [REDACTED]
Attachments: (4) Download all attachments
Description.txt (8 KB); poc [REDACTED] KB; poc [REDACTED] KB; EOS exploit proof of concept-1.mp4 (5 MB)

Hi Daniel,

I am security researcher at Qihoo 360. We have found a critical bug on EOS node which can result in remote code execution. We can remotely own the nodes who parsed our malicious contract.

EOS fixed the vulnerability and Daniel would give the acknowledgement.



Technical Detail of the Vulnerability

This is a buffer out-of-bounds write vulnerability at `libraries/chain/webassembly/binaryen.cpp` (Line 78). Function `binaryen_runtime::instantiate_module`:

```
for (auto& segment : module->table.segments) { Address offset = ConstantExpressionRunner<TrivialGlobalManager>(globals).visit(segment.offset).value.get32(); assert(offset + segment.data.size() <= module->table.initial); for (size_t i = 0; i != segment.data.size(); ++i) { table[offset + i] = segment.data[i]; <span style="color: #ff0000; /*<span> */> OOB write here !*</span> } }
```

 Here `table` is a `std::vector` containing the names in the function table. When storing elements into the table, the `[offset]` field is not correctly checked. Note there is an assert before setting the value, which checks the offset, however unfortunately, `[assert]` only works in Debug build and does not work in a Release build. The table is initialized earlier in the statement: `table.resize(module->table.initial)`. Here `[module->table.initial]` is read from the function table declaration section in the WASM file and the valid value for this field is 0 ~ 1024. The `[offset]` field is also read from the WASM file, in the data section, it is a signed 32-bit value. So basically with this vulnerability we can write to a fairly wide range after the table vector's memory.

How to reproduce the vulnerability

1. Build the release version of latest EOS code

```
./eosio-build.sh
```

2. Start EOS node, finish all the necessary settings described at:

<https://github.com/EOSIO/eos/wiki/Tutorial-Getting-Started-With-Contracts>

3. Set a vulnerable contract:

We have provided a proof of concept WASM to demonstrate a crash.

In our PoC, we simply set the `[offset]` field to `0xffffffff` so it can crash immediately when the out of bound write occurs. To test the PoC: `cd poc cleos set contract eosio ./poc -p eosio` If everything is OK, you will see `nodeos` process gets segment fault. The crash info:

```
(gdb) c Continuing. Program received signal SIGSEGV, Segmentation fault. 0x000000000a32f7c in
eosio::chain::webassembly::binaryen::binaryen_runtime::instantiate_module(char const, unsigned long, std::vector<unsigned char,
std::allocator >) () (gdb) x/ $pc => 0xa32f7c
<_ZN5eosio5chain11webassembly8binaryen16binaryen_runtime18instantiate_moduleEPKcmSt6vectorIhSaIhEE+2972>: mov %rcx,
(%rdx,%rax,1) (gdb) p $rdx $1 = 59699184 (gdb) p $rax $2 = 34359738360 Here |rdx| points to the start of the |table| vector, And |rax| is
0x7FFFFFFF8, which holds the value of |offset| 8.
```

Exploit the vulnerability to achieve Remote Code Execution

This vulnerability could be leveraged to achieve remote code execution in the `nodeos` process, by uploading malicious contracts to the victim node and letting the node parse the malicious contract. In a real attack, the attacker may publish a malicious contract to the EOS main network. The

malicious contract is first parsed by the EOS super node, then the vulnerability was triggered and the attacker controls the EOS super node which parsed the contract. The attacker can steal the private key of super nodes or control content of new blocks. What's more, attackers can pack the malicious contract into a new block and publish it. As a result, all the full nodes in the entire network will be controlled by the attacker. We have finished a proof-of-concept exploit, and tested on the nodeos build on 64-bits Ubuntu system. The exploit works like this:

1. The attacker uploads malicious contracts to the nodeos server.
2. The server nodeos process parses the malicious contracts, which triggers the vulnerability.
3. With the out of bound write primitive, we can overwrite the WASM memory buffer of a WASM module instance. And with the help of our malicious WASM code, we finally achieves arbitrary memory read/write in the nodeos process and bypass the common exploit mitigation techniques such as DEP/ASLR on 64-bits OS.
4. Once successfully exploited, the exploit starts a reverse shell and connects back to the attacker.

You can refer to the video we provided to get some idea about what the exploit looks like, We may provide the full exploit chain later.

http://v.youku.com/v_show/id_XMzYzMTg1NjYwMA==.html

The Fixing of Vulnerability

Bytemaster on EOS's github opened issue 3498 for the vulnerability that we reported:

Convert assert() into throwing exceptions in WAVM and BINARYEN #3498

Closed bytemaster opened this issue 13 hours ago · 0 comments

 bytemaster commented 13 hours ago · Contributor

If any of these asserts trigger in release it shouldn't pass, but should throw. Allowing the code to continue running in release is a potential security vulnerability and will likely result in crashes elsewhere.

And fixed the related code

EOSIO / eos

Branch: master → eos / libraries / chain / webassembly /

 bytemaster fix asserts that get compiled out in debug

..

 **binaryen.cpp** fix asserts that get compiled out in debug

But as the comment made by Yuki on the commit, the fixing is still have problem on 32-bits process and not so prefect.

2  libraries/chain/webassembly/binaryen.cpp

@@ -73,7 +73,7 @@ std::unique_ptr<wasm_instantiated_module_interface> binaryen_runtime::instantiat

73 73 table.resize(module->table.initial);
74 74 for (auto& segment : module->table.segments) {
75 75 Address offset = ConstantExpressionRunner<TrivialGlobalManager>(globals).visit(segment.offset).value.get32();
76 76 - assert(offset + segment.data.size() <= module->table.initial);
76 + FC_ASSERT(offset + segment.data.size() <= module->table.initial);
77 77 for (size_t i = 0; i != segment.data.size(); ++i) {
78 78 table[offset + i] = segment.data[i];
79 79 }
79
79 }

1 comment on commit ea89dce

 guhe120 commented on ea89dce 5 hours ago

Hi, there is still some problem with this patch. in 32-bits process, offset + segment.data.size() could overflow and bypass the FC_ASSERT check

本文链接: <https://blogs.360.cn/post/eos-node-remote-code-execution-vulnerability.html>

-- EOF --

作者 admin001 发表于 2018-05-29 07:18:45 , 添加在分类 Blockchain Threat Intelligence Vulnerability Analysis 下 , 最后修改于 2018-08-24 08:38:43

分享到: 新浪微博微信Twitter印象笔记QQ好友有道云笔记

© 2022 - 360 核心安全技术博客 - blogs.360.cn
Powered by [ThinkJS](#) & [FireKylin](#) 1.3.1