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

05月29, 2018

Vulnerability Credit

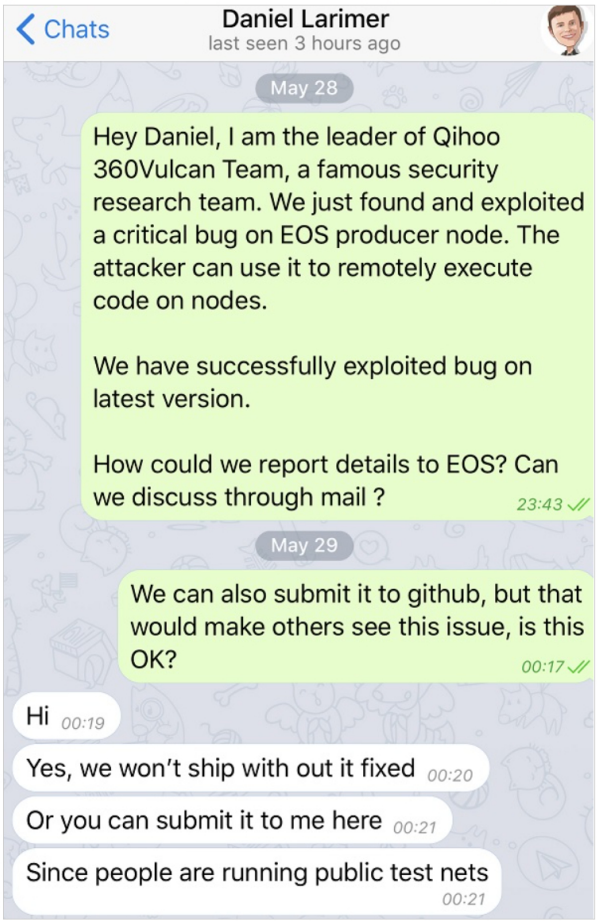
Yuki Chen of Qihoo 360 Vulcan Team Zhiniang Peng of Qihoo 360 Core Security

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



He provided his mailbox and we send the report to him

EOS WASM Contract Function Table Array Out of Bounds Write Vulnerabilityvvvvvvvvvvv

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.geti32(); assert(offset + segment.data.size() <= module->table.initial); for (size_t i = 0; i != segment.data.size(); ++i) { table[offset + i] = segment.data[i]; **<= OOB write here !** } }` Here table is a std::vector contains the Names in the function table. When storing elements into the table, the |offset| filed is not correctly checked. Note there is a 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| filed is also read from the WASM file, in the data section, it is a signed 32-bits 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. 0x0000000000a32f7c in eosio::chain::webassembly::binaryen::binaryen_runtime::instantiate_module(char const, *unsigned long, std::vector<unsigned char, std::allocator >) () (gdb) x/i $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 publishes 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:

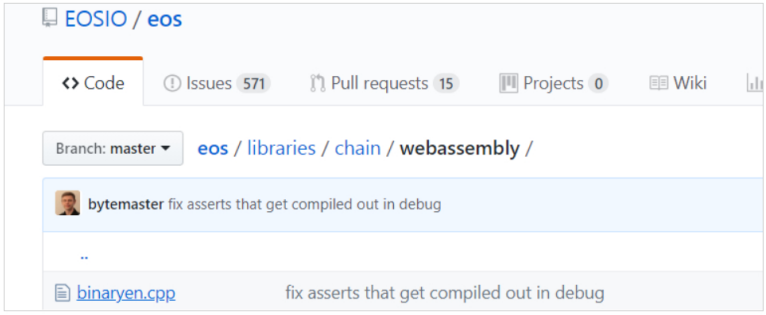
1. The attacker uploads malicious contracts to the nodeos server.
2. The server nodeos process parses the malicious contracts, which triggers the vulnerability.
3. With the out of bound write primitive, we can overwrite the WASM memory buffer of a WASM module instance. And with the help of our malicious 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

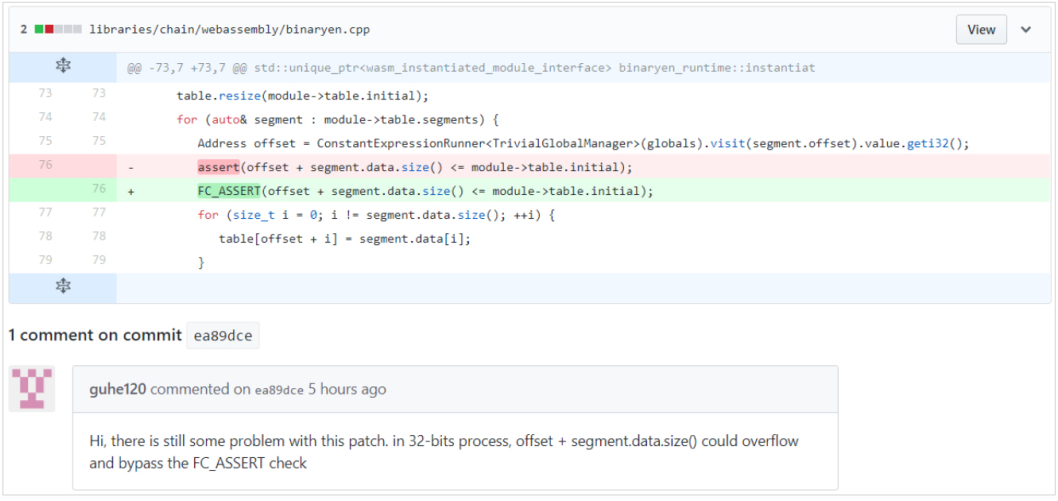
The Fixing of Vulnerability

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

And fixed the related code



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



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

« APT-C-06组织在全球范围内首例使用"双杀"0day漏洞(CVE-2018-8174)发起的APT攻击分析及溯源
EOS节点远程代码执行漏洞 --- EOS智能合约WASM函数表数组越界 »

Comments