DeHacker

Code Security Assessment

EOS

March 31th, 2025

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	EOS
Platform	EOS (EOS)
Website	https://eos.io
Type	DeFi
Language	C++
Codebase	base

Vulnerability Summary

Vulnerability Level	**Total**	**Mitigated**	**Declined**	**Acknowledged**	**Partially Resolved**	**Resolved**
Critical	0	0	0	0	0	0
Major	1	1	0	0	0	0
Medium	2	0	0	0	0	2
Minor	2	0	0	0	0	2
Informational	0	0	0	0	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
SYS	contracts/system.entry.cpp	e8616606504173e96ba5076807f6c627d3f65109dcbcb1f7f984483154229b4f
OLD	contracts/include/system/oldsystem.hpp	a34f23f62710fbed262484e7714036bf179b12b25164baf6fb4a315617f2b94f
SYE	contracts/include/system/system.entry.hpp	cd43b5f834cf03bbc46a178feff7ebe6f88ec97917a1e05fe09a7f73033e7091
TOE	contracts/include/system/token.hpp	98996e50de23a6aeb11bfdb09639533be15ac6afa0d30545159e3e9c1504c3b
TON	contracts/include/token/token.hpp	f59ca79b3b276029401cc37ed9f3a1a83012e7e84593d95e21e58067068088da

Findings

ID	Title	Severity	Status
SYS-03	Centralization Related Risks	Major	Mitigated
SYS-04	System_contract :: retire() is Redundant	Medium	Resolved
SYS-05	Lack Of Authorization In system_contract :: swapexcess()	Medium	Resolved
SYS-06	System_contract :: add_balance() Overrides ram_payer Argument	Minor	Resolved
SYS-07	system_contract :: blockswapto() Auth Inconsistency	Minor	Resolved

MAJOR

SYS-03 | Centralization Related Risks

Issue	Severity	Location	Status
Centralization	Major	contracts/system.entry.cpp: 16	Mitigated

Description

In the contract system_contract, the self can execute functions:

- init
- retire
- blockswapto
- swaptrace

If the deployer controls the contract account's active permission, they can effecticely call any action that require_auth(get_self()).

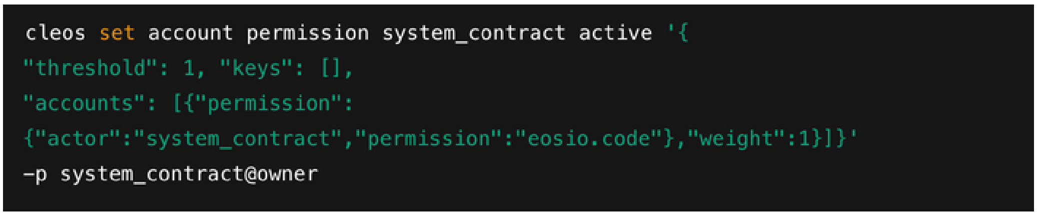
- require_auth(get_self()) ensure that only the contract itself can authorize the action.
- However, the deployer (or any other entity controlling the contract's permissions) can sign transactions as the contract account.
- This means the deployer can call any self-protected action because that can authorize it on behalf of the contract.

Any compromise to the deployer account may allow a hacker to take advantage of this authority and burn all the XYZ, block any account from calling swapto(), or emit swaptrace.

Recommendation

To make a contract truly autonomous, the deployer should:

- Remove their key from the active permission
- If no one controls active, the contract will only execute the logic it was programmed to do.



```
cleos set account permission system_contract active '{
"threshold": 1, "keys": [],
"accounts": [{"permission":
{"actor":"system_contract","permission":"eosio.code"},"weight":1}]}'
-p system_contract@owner
```

This ensure only the smart contract’s own code can trigger actions. No external entity can sign transactions as the contract.

- As long the deployer’s key has owner permission, they can modify the action permission at any time.
- Remove the deployer’s key from the owner permission using the cleos command.
- After renouncing your ownership and permissions, make sure the contract operates as expected without you having control over it. Attempting to execute a function that requires require_auth(get_self()) should now fail for you as the deployer.

MEDIUM

SYS-04 | System_contract :: retire() is Redundant

Issue	Severity	Location	Status
Centralization	Medium	contracts/system.entry.cpp: 100	Resolved

Description

The only currency in statstable is XYZ added by system_contract::init() with issuer = get_self() . The contractdoesn't have issue() and create() actions. system_contract::retire() allows the issuer (contract itself) to burnquantity from the contract balance. This makes no sense and can potentially break the contract invariants. The ownerargument is also unused.

Recommendation

We recommend removing of system_contract::retire().

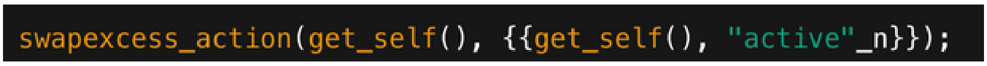
MEDIUM

SYS-05 | Lack Of Authorization In system_contract :: swapexcess()

Issue	Severity	Location	Status
Volatile Code	Medium	contracts/system.entry.cpp: 272	Resolved

Description

The system_contract::swap_after_forwarding() function utilizes the {account, "active"_n} permission to transferEOS tokens from the account to the contract itself. This is a private function that is called by thesystem_contract::swapexcess() action.The system_contract::swapexcess() action only uses require_auth(get_self()) , which does not ensure thatauthorization from account is provided.The swapexcess_action() is invoked by multiple other actions: bidrefund , sellram , powerup , refund , andclaimrewards . In every instance, the action is called without the {account, "active"_n} permission, as shown below:



```
swapexcess_action(get_self(), {{get_self(), "active"_n}});
```

Consequently, the excesses will not be swapped.

Recommendation

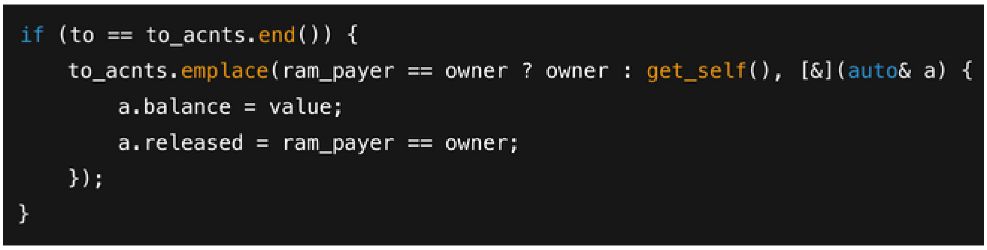
We recommend calling swapexcess_action with {{account, "active"_n}, {get_self(), "active"_n}} permissionsattached.

MINOR

SYS-06 | System_contract :: add_balance() Overrides ram_payer Argument

Issue	Severity	Location	Status
Volatile Code	Minor	contracts/system.entry.cpp: 125	Resolved

Description



```
if (to == to_acnts.end()) {
    to_acnts.emplace(ram_payer == owner ? owner : get_self(), [&](auto& a) {
        a.balance = value;
        a.released = ram_payer == owner;
    });
}
```

The function system_contract::add_balance() employs self as the ram_payer when the balance is not updated bythe owner . For instance, in the system_contract::transfer() function, the from account is designated as theram_payer when a new account is created. However, this behavior is not mirrored in the add_balance() function, leadingto a lack of clarity in its execution.

Recommendation

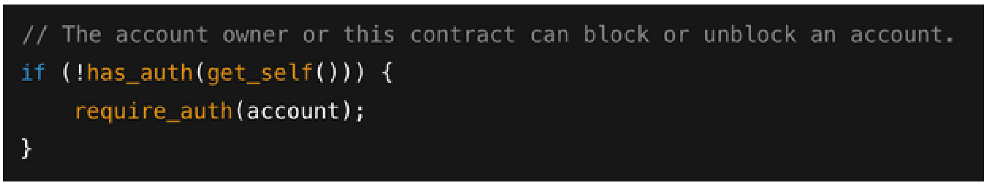
We recommend clarifying the intended behavior and nor overriding the function argument.

MINOR

SYS-07 | System_contract :: blockswapto() AUTH INCONSISTENCY

Issue	Severity	Location	Status
Volatile Code	Minor	contracts/system.entry.cpp: 189	Resolved

Description



```
// The account owner or this contract can block or unblock an account.
if (!has_auth(get_self())) {
    require_auth(account);
}
```

When system_contract::blockswapto() is called by self, it still uses account as a ram_payer to insert the element into _blocked . However, since there's no authorization for an account , the action will fail.

It is unclear when the action is executed by the contract itself.

Recommendation

We recommend removing of if (!has_auth(get_self())) or using a correct name as a ram_payer.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum

Eos

Cosmos

Substrate

TECH STACK

Python

Rust

Solidity

c++

CONTACTS

https://dehacker.io

https://twitter.com/dehackerio

https://github.com/dehacker/audits_public

https://t.me/dehackerio

https://blog.dehacker.io/

DeHacker

March 31th 2025