慢雾科技
slow mist

EOSIO.WPS

Smart Contract Security Audit

2020-04-15

1. Abstract

This report provides a comprehensive view of the security audit results for the smart contract code of the EOSIO.WPS project. The task of SlowMist is to review and point out security issues in the audited smart contract code.

2. Disclaimer

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility base on these. For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of the smart contract, and is not responsible for it. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of this report (referred to as "the provided information"). If the provided information is missing, tampered, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom.

3. Summary

In this report, SlowMist audited the smart contract code for the EOSIO.WPS project. The audit results showed that no issues with critical severity or high severity were found. However, some issues with medium severity and low severity were discovered. After communication and feedback from both parties, the issues have been fixed.

4. Project Overview

4.1 Description

The following are the details of the smart contract code of the audited project (EOSIO.WPS)

The project link: https://github.com/EOS-Nation/eos-wps

Initial commit: 51399034ca947c66cc6c4429bd439f41308f5340

Final commit: e115197eb78b6f6a851f253f68784324383fae14

4.2 Project Structure

The project includes the following smart contract files:

```
./src
├── activate.cpp
├── claims.cpp
├── comments.cpp
├── complete.cpp
├── deposits.cpp
├── drafts.cpp
├── eosio.wps.cpp
├── on_notify.cpp
├── proposers.cpp
├── refresh.cpp
├── settings.cpp
├── utils
│   └── get_tx_id.cpp
└── vote.cpp

./external
├── eosio.system
│   ├── eosio.system.abi
│   ├── eosio.system.wasm
```

```
|     └── include
|         └── eosio.system
└── eosio.token
    ├── eosio.token.abi
    ├── eosio.token.wasm
    ├── include
    |   └── eosio.token
    └── src
        └── eosio.token.cpp

./include
└── eosio.wps
    └── eosio.wps.hpp
```

4.3 Contracts Structure

EOSIO.WPS is mainly divided into three parts: drafts, proposals, votes, and complete, with the main logic located in drafts.cpp, active.cpp, vote.cpp, and complete.cpp, respectively. Users need to first submit a proposal through the `submitdraft()` function in drafts.cpp, then send 100 EOS to the contract and activate the proposal through the `activate()` function in active.cpp to enter the voting process. At the end of each voting cycle, any user can manually call the `complete()` function in complete.cpp to settle and receive the proposal budget.

5. Audit Methodology

The security audit process for smart contracts consists of the following two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house automated analysis tools.
- Manually audit the security of the code. Discover the potential security issues in the code by manually analyzing the contract code.

The following is a list of common vulnerabilities that will be highlighted during the contract code audit:

- Overflow Audit
- Authority Vulnerability Audit
- Authority Excessive Audit
- Shown coding Audit
- Abnormal check Audit
- Type safety Audit
- Denial of Service Audit
- Performance Optimization Audit
- Design Logic Audit
- False Notice Audit
- False Error Notification Audit
- Counterfeit Token Audit
- Random Number Security Audit

- Rollback Attack Audit
- Replay Attack Audit
- Micro Fork Audit
- Tx-Exclude Attack Audit

6. Audit Result

6.1 Critical Vulnerabilities

Critical severity issues can have a major impact on the security of smart contracts, and it is highly recommended to fix critical severity vulnerability.

The audit has shown no critical severity vulnerability.

6.2 High Risk Vulnerabilities

High severity issues can affect the normal operation of smart contracts, and it is highly recommended to fix high severity vulnerability.

The audit has shown no high severity vulnerability.

6.3 Medium Risk Vulnerabilities

Medium severity issues can affect the operation of a smart contract, and it is recommended to fix medium severity vulnerability.

6.3.1 Missing Check

(1) In setting.cpp the `init()` function does not check whether `voting_interval` is greater than a

month in seconds, if it less then a month, for example, 20 minutes, a 5-duration proposal will get all budget after 5 voting_intervals other than 5 months.

```
[[eosio::action]]
void wps::init( const wps_parameters params )
{
    require_auth( get_self() );
    const name ram_payer = get_self();

    check( !_state.exists(), "already initialized" );

    // define `settings`
    auto settings = params;
    //SlowMist// Not check the voting_internal
    _settings.set( settings, ram_payer );

    // set available EOS as `available_funding`
    auto state = _state.get_or_default();
    state.available_funding = token::get_balance( CORE_TOKEN_CONTRACT, get_self(), CORE_SYMBOL.code() );

    // start of voting period will start at the nearest 00:00UTC
    const uint64_t now = current_time_point().sec_since_epoch();
    const time_point_sec current_voting_period = time_point_sec(now - now % DAY);

    // define `state`
    state.current_voting_period = current_voting_period;
    state.next_voting_period = state.current_voting_period + settings.voting_interval;
    _state.set( state, ram_payer );

    // check if WPS account has enough funding to initialize the first voting period
    check_available_funding();
}
```

Fix status: Fixed in commit adc43133f2269b7f73e00b7fd26313a3c6a187d0.

6.3.2 Logic Defect

(1) In complete.cpp, line 26, the `copy_current_to_next_periods()` function copy both

`active` and `expired` proposals to the next period.

```
void wps::copy_current_to_next_periods()
{
    const name ram_payer = get_self();
    auto state = _state.get();
    auto settings = _settings.get();

    // must calculate next_voting_period in case the next period has been changed caused by delayed complete action
    // cannot use `state.next_voting_period`
    const time_point_sec next_voting_period = current_time_point() + time_point_sec( settings.voting_interval );

    // lookup iterators
    auto current_periods_itr = _periods.find( state.current_voting_period.sec_since_epoch() );
    auto next_periods_itr = _periods.find( next_voting_period.sec_since_epoch() );

    // create new set of proposals
    if ( next_periods_itr == _periods.end() ) {
        _periods.emplace( ram_payer, [&]( auto& row ) {
            row.voting_period    = next_voting_period;
            //Slowmist// copy all proposals, include expired proposal
            row.proposals        = current_periods_itr->proposals;
        });
    // insert proposal to old ones
    } else {
        _periods.modify( next_periods_itr, ram_payer, [&]( auto& row ) {
            row.proposals = current_periods_itr->proposals;
        });
    }
}
```

Fix status: Fixed in commit e115197eb78b6f6a851f253f687843243 83fae14

(2) In file complete.cpp, the `set_pending_to_active()` function only update one `pending` proposal each time, if there is more than one proposal, the remaining proposal will not be updated.

```
void wps::set_pending_to_active()
{
    auto index = _proposals.get_index<"bystatus"_n>();
    auto itr = index.find("pending"_n.value);

    if (itr == index.end()) return;

    index.modify( itr, same_payer, [&]( auto& row ) {
        row.status = "active"_n;
    });
}
```

Fix status: Fixed in commit e115197eb78b6f6a851f253f687843243 83fae14.

(3) In complete.cpp, line 29-32, because the current proposals are copied before updating the voting period, results in `pending` & the `active` proposals don't have the same `voting_period` when they get copied over.

```
[[eosio::action]]
void wps::complete( )
{
    // no authorization required (can be executed by any account)

    // is contract paused or not
    check_contract_active();

    // check if current voting period is completed
    check( is_voting_period_complete(), "[current_voting_period] is not completed");

    // check if account has enough funding
    check_available_funding();

    // update `votes` from eligible voters
    // any existing votes with voters with less than 100 EOS vpay will be removed
    refresh_proposals();

    // update `proposals::eligible` field for all active proposals
    update_eligible_proposals();

    // payouts of active proposals
    handle_payouts();

    // copy current proposals to next period
    copy_current_to_next_periods();

    // update current & next voting period
    update_to_next_voting_period();

    // set pending proposals to active status
    set_pending_to_active();

}
```

Fix status: fixed in commit e115197eb78b6f6a851f253f687843243 83fae14.

6.4 Low Risk Vulnerabilities

Low severity issues can affect smart contracts operation in future versions of code. We recommend the project party to evaluate and consider whether these problems need to be fixed.

6.4.1 Missing check

(1) The proposer does not require to submit his proposer information through proposer.cpp at first when he start to submit his proposal.

```
[[eosio::action]]
void wps::submitdraft(const name proposer,
                      const name proposal_name,
                      const string title,
                      const asset monthly_budget,
                      const uint8_t duration,
                      const map<name, string> proposal_json )
{
    require_auth( proposer );
    const name ram_payer = proposer;

    // get scoped draft
    drafts_table _drafts( get_self(), proposer.value );
    auto drafts_itr = _drafts.find( proposal_name.value );
    auto proposals_itr = _proposals.find( proposal_name.value );
    // check if proposer is eligible to activate proposal
    check_eligible_proposer( proposer );

    // validation
    check( proposals_itr == _proposals.end(), "[proposal_name] activated proposal already exists, try using a different
proposal name" );
    check( drafts_itr == _drafts.end(), "[proposal_name] draft already exists, try using `modifydraft` or `canceldraft` or
`modifybudget`" );
    check( proposal_name.length() > 2, "[proposal_name] should be at least 3 characters in length" );
    check( proposal_name.length() < 13, "[proposal_name] cannot exceed 12 characters in length" );
    check_title( title );
    check_monthly_budget( monthly_budget );
    check_duration( duration );
     //SlowMist// Not require to submit proposer information first
    // create draft proposal
    _drafts.emplace( ram_payer, [&]( auto& row ) {
        row.proposer            = proposer;
        row.proposal_name       = proposal_name;
        row.title               = title;
        row.monthly_budget      = monthly_budget;
        row.duration            = duration;
        row.total_budget        = asset{ monthly_budget.amount * duration, monthly_budget.symbol };
        row.proposal_json       = proposal_json;
    });

    create_deposit_account( proposer, ram_payer );
}
```

Fix status: After feeding back with the project side, it’s confirmed that the problem is ignored, enforcing the proposer to submit additional information is not required.

(2) In file claim.cpp, line 17, the proposer_itr is checked whether it is at the end() after getting the corresponding information of the pointer, it should be checked first after define.

```
[[eosio::action]]
void wps::claim( const eosio::name proposal_name )
{
    // no authorization required (can be executed by any account)
    // static actions
    token::transfer_action transfer( CORE_TOKEN_CONTRACT, { get_self(), "active"_n });

    auto proposals_itr = _proposals.find( proposal_name.value );
    const eosio::name proposer = proposals_itr->proposer;
    const eosio::asset payouts = proposals_itr->payouts;
    const eosio::asset claimed = proposals_itr->claimed;

    // calculate claimable amount
    const eosio::asset claimable = payouts + claimed;

    check( proposals_itr != _proposals.end(), "[proposal_name] does not exist" );
    check( claimable.amount > 0, "no claimable amount" );

    transfer.send( get_self(), proposer, claimable, "wps::" + proposal_name.to_string() );

    add_claim( proposer, proposal_name, claimable );

    _proposals.modify( proposals_itr, same_payer, [&]( auto& row ) {
        row.claimed -= claimable;
    });
}
```

Fix status: Fixed in commit f61a20fd9dc56be8d1cea5ff6d4d60bf421afdbb.

(3) In file active.cpp, there is a logic check in `activate()` function to check whether a proposal is in the min_time_voting_end when it wants to be activated, but there is not a same logic in file complete.cpp in `complete()` function which can also activate a proposal.

```
[[eosio::action]]
void wps::complete( )
{
    // no authorization required (can be executed by any account)

    // is contract paused or not
    check_contract_active();

    // check if current voting period is completed
    check( is_voting_period_complete(), "[current_voting_period] is not completed");

    // check if account has enough funding
    check_available_funding();

    // update `votes` from eligible voters
    // any existing votes with voters with less than 100 EOS vpay will be removed
    refresh_proposals();

    // update `proposals::eligible` field for all active proposals
    update_eligible_proposals();

    // payouts of active proposals
    handle_payouts();

    // set pending proposals to active status
    // SlowMist// not require proposal must be activated out of min_time_voting_end
    set_pending_to_active();

    // update current & next voting period
    update_to_next_voting_period();

    // re-update `proposals::eligible`
    update_eligible_proposals();
}
```

Fix status: After feeding back with the project side, it's confirmed that the problem is ignored, the logic to check `min_time_voting_end` is to prevent user active his proposal in the last few minutes of vote period and lose his EOS.

(4) In file setting.cpp, the `init()` function does not require the `min_time_voting_end` must be greater than the `voting_interval`, and not require symbol of `deposit_required` and `max_monthly_budget` must be EOS.

```
[[eosio::action]]
void wps::init( const wps_parameters params )
{
    require_auth( get_self() );
    const name ram_payer = get_self();

    check( !_state.exists(), "already initialized" );
    // define `settings`
    auto settings = params;
    //Slowmist// The relevant properties are not checked
    _settings.set( settings, ram_payer );

    // set available EOS as `available_funding`
    auto state = _state.get_or_default();
    state.available_funding = token::get_balance( CORE_TOKEN_CONTRACT, get_self(), CORE_SYMBOL.code() );

    // start of voting period will start at the nearest 00:00UTC
    const uint64_t now = current_time_point().sec_since_epoch();
    const time_point_sec current_voting_period = time_point_sec(now - now % DAY);

    // define `state`
    state.current_voting_period = current_voting_period;
    state.next_voting_period = state.current_voting_period + settings.voting_interval;
    _state.set( state, ram_payer );

    // check if WPS account has enough funding to initialize the first voting period
    check_available_funding();
}
```

Fix status: Fixed in commit

0b41e951469ebed52d10e836b1e89bc40742664a,

1601d05cd01640da3bd1fbe046fe520913a3474b.

6.4.2 Parameter not used

(1) In file complete.cpp, line 57, the `claim` variable in handle_payouts() function is not used and should be deleted.

```
void wps::handle_payouts()
{
    // settings
    auto settings = _settings.get();
    auto state = _state.get();
    // static actions
    wps::claim_action claim( get_self(), { get_self(), "active"_n });

    // iterate proposals by active status
    for ( auto proposal_name : group_proposals( "active"_n ) ) {
        auto proposals_itr = _proposals.find( proposal_name.value );
        const eosio::asset monthly_budget = proposals_itr->monthly_budget;
```

Fix status: Fixed in commit 031b72533a375c0aa22965debf2fcfa658567a7f.

(2) In vote.cpp, line 87, the `proposer` variable in update_eligible_proposals is not used.

```
void wps::update_eligible_proposals()
{
    // settings
    auto settings = _settings.get();
    auto state = _state.get();

    // containers
    eosio::asset total_payout = asset{ 0, symbol{ "EOS", 4 }};

    // filter out min voting threshold proposals
    std::map<int16_t, std::set<eosio::name>> proposals = sort_proposals_by_net_votes( "active"_n );

    // iterate proposals from highest to lowest net votes
    for ( auto itr = proposals.rbegin(); itr != proposals.rend(); ++itr ) {

        // iterate over proposals
        for ( auto proposal_name : itr->second ) {
            // proposal variables
            auto proposal_itr = _proposals.find( proposal_name.value );
            const int16_t total_net_votes = itr->first;
            const eosio::name proposer = proposal_itr->proposer;
            const eosio::asset monthly_budget = proposal_itr->monthly_budget;

            // min requirements for payouts
            const bool is_min_vote_margin = total_net_votes >= settings.vote_margin;
            const bool is_enough_budget = (total_payout + monthly_budget) <= settings.max_monthly_budget;

            // set eligible of proposal (true/false)
            _proposals.modify( proposal_itr, same_payer, [&]( auto& row ) {
                if ( is_enough_budget && is_min_vote_margin ) {
                    total_payout += monthly_budget;
                    row.eligible = true;
                } else {
                    row.eligible = false;
                }
            });
        }
    }
}
```

Fix status: Fixed in commit 031b72533a375c0aa22965debf2fcfa658567a7f.

(3) In vote.cpp, line 54, the `proposals_itr` variable is not used.

```
void wps::update_vote( const name voter, const name proposal_name, const name vote )
{
    // validate vote
    auto votes_itr = _votes.find( proposal_name.value );
    auto proposals_itr = _proposals.find( proposal_name.value );

    check( votes_itr != _votes.end(), "[proposal_name] votes does not exist");
    check( vote == "yes"_n || vote == "no"_n || vote == "abstain"_n, "[vote] invalid (ex: yes/no/abstain)");
```

Fix status: Fixed in commit 031b72533a375c0aa22965debf2fcfa658567a7f.

6.5 Enhancement Suggestions

The recommendation for improvement is a logical recommendation for the code to be optimized according to the actual situation.

6.5.1 Logic Defect

(1) In file complete.cpp, when nobody invokes the `complete()` function for a long time, a year for example, someone will need to invoke it for too many times to activate a proposal or make other operations.

```
[[eosio::action]]
void wps::complete( )
{
    // no authorization required (can be executed by any account)

    // is contract paused or not
    check_contract_active();

    // check if current voting period is completed
    check( is_voting_period_complete(), "[current_voting_period] is not completed");

    // check if account has enough funding
    check_available_funding();

    // update `votes` from eligible voters
    // any existing votes with voters with less than 100 EOS vpay will be removed
    refresh_proposals();

    // update `proposals::eligible` field for all active proposals
    update_eligible_proposals();

    // payouts of active proposals
    handle_payouts();

    // set pending proposals to active status
    set_pending_to_active();

    // update current & next voting period
    update_to_next_voting_period();

    // re-update `proposals::eligible`
    update_eligible_proposals();
}
```

Fix status: Fixed in commit c31333a3c2959c14e56d8ae1fc6a3f634930b8d3.

(2) When complete() is invoked, if a proposal is eligible but there is not enough fund in the system, the proposal will be set to `expired`.

```
void wps::handle_payouts()
{
    // settings
    auto settings = _settings.get();
    auto state = _state.get();

    // static actions
    wps::claim_action claim( get_self(), { get_self(), "active"_n });

    // iterate proposals by active status
    for ( auto proposal_name : group_proposals( "active"_n ) ) {
        auto proposals_itr = _proposals.find( proposal_name.value );
        const eosio::asset monthly_budget = proposals_itr->monthly_budget;

        // only payout eligible (true) proposals
        if ( proposals_itr->eligible ) {

            // check internal available funding
            check( state.available_funding >= monthly_budget, "insufficient `available_funding`");
            sub_funding( monthly_budget );

            // update proposal payouts
            _proposals.modify( proposals_itr, same_payer, [&]( auto& row ) {
                row.payouts += monthly_budget;
            });
        } else {
            // remove proposal that no longer met threshold, cancel all subsequent proposals
            _proposals.modify( proposals_itr, same_payer, [&]( auto& row ) {
                row.status = "expired"_n;
            });
        }

        // set proposals to `completed/partial/expired/active`
        update_proposal_status( proposal_name );
    }
}
```

Fix status: After feeding back with the project side, it's confirmed that that's the design.

(3) In complete.cpp, the `update_eligible_proposals()` function executed twice time in `complete()` function.

```
[[eosio::action]]
void wps::complete( )
{
    // no authorization required (can be executed by any account)

    // is contract paused or not
    check_contract_active();

    // check if current voting period is completed
    check( is_voting_period_complete(), "[current_voting_period] is not completed");
    // check if account has enough funding
    check_available_funding();

    // update `votes` from eligible voters
    // any existing votes with voters with less than 100 EOS vpay will be removed
    refresh_proposals();

    // update `proposals::eligible` field for all active proposals
    update_eligible_proposals();

    // payouts of active proposals
    handle_payouts();

    // set pending proposals to active status
    set_pending_to_active();

    // update current & next voting period
    update_to_next_voting_period();

    // re-update `proposals::eligible`
    update_eligible_proposals();
}
```

Fix status: Fixed in commit 073eff02c4706f94dee4df9c9ff17ef581ae67c2.

6.5.2 symbol hard-coded

"EOS" hard-coded in file should be replaced to CORE_SYMBOL to test on any chain for convenient test.

Fix status: Fixed in commit 8f77df700c3e4686d43bb06f95b452a108adbb5a.

慢雾科技
slow mist

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

WeChat Official Account