verichains

SECURITY AUDIT OF

COM2US VOTING CONTRACTS

com2us

Public Report

Mar 06, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
NFT	A non-fungible token (NFT) is a unique digital identifier that cannot be copied, substituted, or subdivided, that is recorded in a blockchain, and that is used to certify authenticity and ownership.
XPLA	XPLA is an open-source blockchain leveraging Tendermint and the Cosmos SDK. It offers extensive experiences in De-Fi and P2O gaming, aiming to transition Web2 users into the Web3 space with EVM compatibility and developer-friendly SDKs.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Mar 06, 2024. We would like to thank the Com2uS for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Com2uS Voting Contracts. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team found some vulnerabilities in the application. Com2uS team has resolved and updated the issue following our recommendations.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Com2uS Voting Contracts

Com2uS is driving innovation in the domestic mobile game business and shaping the future of the global game and entertainment industry.

Com2uS Platform provides an optimized global network and blockchain game platform for global services in line with the paradigm of the future content industry.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the smart contracts of Com2uS Voting Contracts.

It was conducted on the following git repository *https://github.com/c2xnft/xpla_vote_contracts* on commit `a4261545db46466d47a3beed9e99f084725c9857`.

The latest version of the following files were made available in the course of the review:

SHA256 Sum	File
16038986745a66827963da4b85e1413b5ce1c4f9f801103d7e251e690fcc3c6f	./error.rs
8ec6263aaad8c8b928af15f77f36c4342dc2987d1a3419280722049e002350c6	./lib.rs
c8fb0f380926944d162888c693af2e6618365b3684a34f640c9e9f6c4cd6b979	./query.rs
045b1babfa0d8842fc91d61c0c106d98ec8dc305df14bce132ca32b5b1aee24e	./state.rs
921dd55c29f934696b66569a5b98c03dc22c2dde6e051301da55d827e5592e67	./contract.rs
823e1de6bbf30524ba6fc0f133700d041275f513b1edff6efcb06aa4baf21cf7	./msg.rs
11abe6a3efff4300294d71c7bbb86c048190d50f84454520c5ca4d15b8d0e236	./vote.rs

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Com2uS acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Com2uS understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Com2uS agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Com2uS will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Com2uS, the final report will be considered fully accepted by the Com2uS without the signature.

2. AUDIT RESULT

2.1. Overview

This is an XPLA blockchain-based smart contract designed to create a voting system. It includes a voting contract and test code developed using CosmWasm.

In general voting, each wallet can only vote once, with one vote per wallet.

For NFT voting, individuals receive voting rights corresponding to the number of NFTs they own, enabling multiple votes, but each NFT can only be used once for a specific vote.

Admins can vote without any condition checking in the smart contract. However, when sending a message to a smart contract, all conditions must have already been verified.

2.2. Findings

During the audit process, the audit team found some vulnerabilities in the given version of Com2uS Voting Contracts.

Com2uS fixed the code, according to Verichains's draft report.

#	Issue	Severity	Status
1	NFT token can be used multiples time when casting vote	CRITICAL	Fixed
2	Incorrect updated value of `admin_create`	LOW	Fixed
3	Missing admin duplication check	LOW	Fixed
4	Incorrect error message	INFORMATIVE	Fixed
5	`AlreadyUsedNftTokens` error should be used instead of `generic_err`	INFORMATIVE	Fixed

2.2.1. NFT token can be used multiples time when casting vote CRITICAL

Affected files:

- contracts/vote/src/vote.rs

The voting power is calculated based on the NFT tokens owned by the voter. However, the NFT tokens can be used multiple times when casting a vote. This is because the contract does not update the `USED_NFT_TOKENS` state while looping through the `unchecked_tokens` list. This can be exploited to increase the voting power by using the same NFT tokens multiple times.

```
fn get_voting_power_in_nft_condition(
    deps: Deps,
    vote_id: u64,
    addr: &Addr,
    nft_condition: &NftCondition,
    nft_tokens: &Vec<String>,
) -> StdResult<u64> {
    // ...
    while !unchecked_tokens.is_empty() {
        let token = unchecked_tokens.pop().unwrap();
        let owned = owned_tokens.iter().any(|x| x == &token);
        if !owned {
            return Err(StdError::generic_err("Not owned NFT tokens"));
        }
        let used = USED_NFT_TOKENS.may_load( // AUDIT: STATE NOT UPDATED
            deps.storage,
            (
                vote_id,
                &nft_addr,
                &token,
            ),
        )?;
        match used {
            Some(_) => {
                return Err(StdError::generic_err("Already used NFT tokens"));
            }
            None => {
                power += 1;
            }
        }
    }
}
```

UPDATES

- **Mar 06, 2024**: This issue has been acknowledged and fixed by Com2uS team by checking duplicated NFT tokens in the `nft_tokens` input.

2.2.2. Incorrect updated value of `admin_create` LOW

Affected files:

- contracts/vote/src/contract.rs

The `admin_create` value is not updated correctly in the `execute_update_config` function; it is always set to `false`, regardless of the input value.

```
pub fn execute_update_config(
    mut deps: DepsMut,
    env: Env,
    info: MessageInfo,
    admin_create: bool,
) -> Result<Response, ContractError> {
    only_admin(deps.as_ref(), info.clone())?;

    CONFIG.save(
        deps.storage,
        &Config {
            admin_create: false, // AUDIT: INCORRECT UPDATED VALUE
        },
    )?;

    increase_block(&mut deps, &env)?;

    Ok(
        Response::default()
            .add_attribute("action", "update_config")
            .add_attribute("admin_create", admin_create.to_string())
            .add_attribute("sender", info.sender)
    )
}
```

UPDATES

- **Mar 06, 2024**: This issue has been acknowledged and fixed by Com2uS team.

2.2.3. Missing admin duplication check LOW

Affected files:

- contracts/vote/src/contract.rs

The `execute_add_admins` function does not check for duplicate admin addresses before adding them to the `ADMINS` state. This can lead to multiple identical admin addresses being added to the state.

```
pub fn execute_add_admins(
    mut deps: DepsMut,
    env: Env,
    info: MessageInfo,
    admins: Vec<String>,
) -> Result<Response, ContractError> {
    only_admin(deps.as_ref(), info.clone())?;

    let new_admin_addrs = admins.iter()
        .map(|admin| deps.api.addr_validate(admin))
        .collect::<StdResult<Vec<_>>>()?;
    let mut admin_addrs = ADMINS.load(deps.storage)?;
    admin_addrs.extend_from_slice(&new_admin_addrs);
    ADMINS.save(deps.storage, &admin_addrs)?;
    increase_block(&mut deps, &env)?;

    Ok(Response::default()
        .add_attribute("action", "add_admins")
        .add_attribute("admins", admins.join(","))
        .add_attribute("sender", info.sender))
}
```

UPDATES

- **Mar 06, 2024**: This issue has been acknowledged and fixed by Com2uS team by changing the type of `ADMINS` state to `HashSet<Addr>` and checking for duplicate admin addresses before adding them to the state.

2.2.4. Incorrect error message INFORMATIVE

Affected files:

- contracts/vote/src/error.rs
- contracts/vote/src/contract.rs

The error message for the `TooLargeChoiceValue` error is incorrect. The error message states that the choice value must be 1 or 0, but the error is thrown when the choice value exceeds the voting power.

```
// contracts/vote/src/error.rs
#[error("Choice value ({value}) must be 1 or 0")]
TooLargeChoiceValue {
    value: u64,
}

// contracts/vote/src/contract.rs
pub fn validate_ballots(
    items: &Vec<ItemInfo>,
    vote_id: u64,
    ballots: &HashMap<u64, Vec<u64>>,
    vote_power: u64,
) -> Result<(), ContractError> {
    // ...
    let too_large_choice_value = choices.iter().find(|value| value > &&vote_power)
        .map(|value| Err(ContractError::TooLargeChoiceValue {
            value: *value,
        }))
        .unwrap_or(Ok(()))?;
    // ...
}
```

UPDATES

- **Mar 06, 2024**: This issue has been acknowledged and fixed by Com2uS team.

2.2.5. `AlreadyUsedNftTokens` error should be used instead of `generic_err` INFORMATIVE

Affected files:

- contracts/vote/src/error.rs
- contracts/vote/src/vote.rs

```
fn get_voting_power_in_nft_condition(
    deps: Deps,
    vote_id: u64,
    addr: &Addr,
    nft_condition: &NftCondition,
    nft_tokens: &Vec<String>,
) -> StdResult<u64> {
    // ...
    while !unchecked_tokens.is_empty() {
        // ...
        match used {
            // ...
            Some(_) => {
                return Err(StdError::generic_err("Already used NFT tokens")); // AUDIT: USE
AlreadyUsedNftTokens ERROR INSTEAD
            }
            // ...
        }
    }
}
```

UPDATES

- **Mar 06, 2024**: This issue has been acknowledged and fixed by Com2uS team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 06, 2024*	Public Report	Verichains Lab

Table 2. Report versions history