CodeHawks

First Flight
Competition

2024-08 PRESIDENT ELECTOR

@JJS_OnChain

JORDAN J.
SOLOMON

SUMMARY

Prepared By: Jordan J. Solomon

Competition Type: CodeHawks First Flight

Competition Dates: Sep 12th, 2024 → Sep 19th, 2024

Audit Summary

First Flight audits are designed to help beginner security researchers improve their skills. These audits test fundamental concepts of blockchain development and the potential attacks associated with them.
PresidentElector follows the same principle. It is a simple protocol that changes the way traditional politics is made. Instead of having a one person on vote mechanism, the protocol suggests a ranking election.

@JJS_OnChain

TABLE OF CONTENTS

Project Information

Risk Assessment

PROJECT OVERVIEW

Project Summary

Project Name	**PresidentElector First Flight Competition**
Language	Solidity
Code Base	https://github.com/Cyfrin/2024-09-president-elector
Commits	**N/A**

Audit Summary

End Date	**19th of Sep 2024**
Methodology	Manual Review, Unit Tests.

Vulnerabilities Summary

High	**0**
Medium	1
Low	0
Info/Gas	**1**

AUDIT SCOPE & METHODOLOGY

Vulnerabilities Classification

High	Impact is extremely high on the protocol, with a combination of highly likely to happen
Medium	Risk of exploits that may occur and impact could be important.
Low	Minor issues that are less likely to occur and if they would the risk is not high.
Info/Gas	Simple improvements that have no risk for the protocol. Miss use of best practices, gas efficiency etc.

Methodology

- **Manual Review - Walking through the code. First getting an understanding of the code base. Then to try and find vulnerabilities.**
- **Unit Tests - Using Solidity to test the code and make sure a specific feature does what it needs to do.**

FINDINGS

MEDIUM

M-1	Wrong definition of the **TYPEHASH** leads to the vote with signatures functionality to be unusable

INFO

I-1	Wrong definition of the **TYPEHASH** leads to the vote with signatures functionality to be unusable

[M-1] Wrong Definition of the `TYPEHASH` Leads to the Vote With Signatures Functionality to be Unusable

Type	Severity	Location
Logic Error	Medium	RankedChoice: 23

Description

In the protocol's docs, it states that a big part of the protocol's functionality is that voters can let others spend gas for they're votes by using signatures. The following function is in charge of doing so:

```
function rankCandidatesBySig(address[] memory orderedCandidates, bytes memory signature) external {
    bytes32 structHash = keccak256(abi.encode(TYPEHASH, orderedCandidates));
    bytes32 hash = _hashTypedDataV4(structHash);
    address signer = ECDSA.recover(hash, signature);
    _rankCandidates(orderedCandidates, signer);
}
```

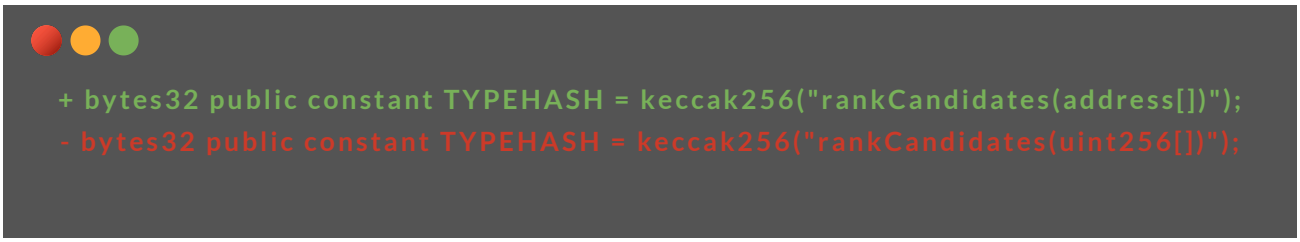
The function uses a variable called `TYPEHASH` that contains the function's signature and is encoded to create the hash eventually. However, in the `TYPEHASH` declaration the function's signature is wrong. It uses a uint256 and not string!

Impact

In the **rankCandidatesBySig** function the hash will be encoded with the wrong input parameters for the function, which will revert every transaction made with the signatures. Which means that an important capability in the protocol will not work

Recommended Mitigation

Fix the TYPEHASH to the correct input parameters:



```
+ bytes32 public constant TYPEHASH = keccak256("rankCandidates(address[])");
- bytes32 public constant TYPEHASH = keccak256("rankCandidates(uint256[])");
```

[I-1] No checks for address(0) in `rankCandidates` could potentially lead to no one winning the election

Type	Severity	Location
Checks	Low	Pot.sol: 159

Description

Voters can call the `rankCandidates` `external` function, or the `rankCandidatesBySig` function, that calls an internal function to rank the candidates (`_rankCandidates`). The voter passes through an ordered candidate list, with their ranking.

However, there is no checks for an `address(0)` in `rankCandidates` or `_rankCandidates`. In the case of `rankCandidatesBySig`, the `ECDSA.recover` function calls an internal function in the `ECDSA` contract that does check for an `address(0)`, so this does not apply.

Impact

LOW. This is because the likelihood of voters actually voting for an `address(0)` is very low. However, This could mean that, potentially, an empty address could win the election, and therefore no one is the actual president!

Recommended Mitigation

Have a for loop that checks for address(0) in the `_rankCandidates`