TestMachine

SECURITY ANALYSIS REPORT (SAR)

Project: 0xc011a73ee8576fb46f5e1c5751ca3b9fe0af2a6f
Based on: Snapshot 46
Tools used: Snap Scan
2023-12-13_06-16

⬡ Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product" created by any team that contracts TestMachine. This report is based on the scope of materials and documentation provided to TestMachine. Results may not be complete nor inclusive of all vulnerabilities. This report does not provide any warranty or guarantee regarding the absolute product, software or services or that such will be bug-free, without risk or subject to vulnerabilities nature of the technology analyzed.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides financial or investment advice, nor should be leveraged as financial or investment advice of any sort. TestMachine's position is that each company and individual are responsible for their own due diligence and continuous security. TestMachine's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies.

⬡ Results Summary

Totals by Severity

Severity	Count
⬢High	1
⬢Medium	1
⬢Low	0
⬢Informational	0
⬢Optimization	0
⬢Unknown	3

Totals by Tool

Tool	Count
Snap Scan	5

⬡ Findings from tool: Snap Scan

⬢ high

Unprotected Ether Withdrawal SWC-105

/0xc011a73ee8576fb46f5e1c5751ca3b9fe0af2a6f.sol:332:333

Found error: transfer. Found 1 unprotected Ether withdrawals:. Unprotected Ether withdrawal can lead to unauthorized transactions.

```

```

Recommendation

Use the checks-effects-interactions pattern and ensure that the caller is authorized.

⬢ unknown

State Variable Default Visibility SWC-108

/0xc011a73ee8576fb46f5e1c5751ca3b9fe0af2a6f.sol:263:264

Found error: messageSender. Found 1 state variables with default visibility:. Default visibility can lead to exposure of sensitive data or contract internals.

```
265.
```

Recommendation

Explicitly declare the visibility of all state variables.

⬢ unknown

Variable Shadowing SWC-119

/0xc011a73ee8576fb46f5e1c5751ca3b9fe0af2a6f.sol:328:329

Found error: owner in balanceOf. Found 2 potential variable shadowing instances:. Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues. Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

```
330.
```

Recommendation

Review storage variable layouts for your contract systems carefully and remove any ambiguities. Always check for compiler warnings as they can flag the issue within a single contract. Avoid using the same variable name in different scopes to prevent any potential confusions and misbehavior.

⬢ unknown

Variable Shadowing SWC-119

/0xc011a73ee8576fb46f5e1c5751ca3b9fe0af2a6f.sol:330:331

Found error: owner in allowance. Found 2 potential variable shadowing instances:. Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues. Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

```
330.
```

Recommendation

Review storage variable layouts for your contract systems carefully and remove any ambiguities. Always check for compiler warnings as they can flag the issue within a single contract. Avoid using the same variable name in different scopes to prevent any potential confusions and misbehavior.

⬢ medium

Message call with hardcoded gas amountSWC-134

/0xc011a73ee8576fb46f5e1c5751ca3b9fe0af2a6f.sol:332:333

Found error: transfer. Found 1 functions with hardcoded gas limits:. The transfer() and send() functions forward a fixed amount of 2300 gas. The gas cost of EVM instructions may change significantly during hard forks which may break already deployed contract systems that make fixed assumptions about gas costs.

Recommendation

Avoid the use of transfer() and send() and do not otherwise specify a fixed amount of gas when performing calls. Use .call.value(...)('') instead. Use the checks-effects-interactions pattern and/or reentrancy locks to prevent reentrancy attacks.

Security Analysis Report (SAR) provided by

TestMachine

https://testmachine.ai