CONSENSYS
Diligence

Pegasys Permissioning Audit

Date	August 2019
Lead Auditor	Steve Marx
Co-auditors	John Mardlin, Dean Pierce

1 Summary

ConsenSys Diligence conducted a security audit on the PegaSys Permissioning smart contracts. These contracts are used to provide on-chain permission rules for a Pantheon node.

2 Audit Scope

This audit covered the following files:

File Name	SHA-1 Hash
AccountIngress.sol	57207a6878535bc2f3d40216d96f07eef9bbdfd9
AccountRulesList.sol	73ffd92be5b6c3b1e18d1b860344dac578c9aa31
Admin.sol	e13931323093f1555f4dfcc74fad6a2c457c1082
AdminProxy.sol	eecd073b4e05a4445fb00888074b48c443c5bbf4
Ingress.sol	b0fcff06fa7d55136cfe483331280e4e9bb9def4
NodeIngress.sol	3f46f78e4c1b9a546287135a13ffa303f62a826b
NodeRulesList.sol	fa9382c4cf3f4d800aa3d0e89bb9a712d5aa5f0c
AccountRules.sol	c730212300e070ed22b1490f6e67347d1f36c051
AccountRulesProxy.sol	1024d00149ee0258f5ee4c0671a09ada723c3645
AdminList.sol	0304e06bfc4c87abc4d2f4c0361633590c5ef830
NodeRules.sol	8f0dc9efd5bc09a8c6346495e23a398c907baf21
NodeRulesProxy.sol	01967d8481a3f1497ecdfcfcd5e7dd2ea9f9c17e

The audit team evaluated that the system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three broad categories:

1. **Security:** Identifying security related issues within the contract.
2. **Architecture:** Evaluating the system architecture through the lens of established smart contract best practices.
3. **Code quality:** A full review of the contract source code. The primary areas of focus include:
    - Correctness
    - Readability
    - Scalability
    - Code complexity
    - Quality of test coverage

3 System Overview

Pantheon, an enterprise Ethereum client, can be used to create permissioned networks. Such a network has rules dictating what nodes are allowed to connect and what transactions are allowed to be submitted. The Permissioning smart contracts are an on-chain mechanism for managing these rules and synchronizing them among clients.

3.1 Detailed Design

Two contracts are included as part of the network's genesis block:

- `AccountIngress` is used to check whether incoming transactions should be allowed. It calls `transactionAllowed` on the `AccountRules` contract.
- `NodeIngress` is used to check whether incoming node connections should be allowed. It calls `connectionAllowed` on the `NodeRules` contract.

These contracts are a layer of indirection that can be used for upgradeability. Rather than directly implement permissioning rules, they delegate to other

contracts which can be swapped at runtime by administrators.

The following contracts provide the actual rules implementations:

- `AccountRules` keeps a whitelist of accounts that are allowed to make transactions.
- `NodeRules` keeps a whitelist of nodes that are allowed to connect to the network.

Finally, the `Admin` contract is used to keep a whitelist of administrator accounts. These accounts are allowed to add or remove things from the whitelists. They can also swap out the rules contracts altogether.

4 Key Observations/Recommendations

- Overall the system is low in complexity but general enough to allow more sophisticated contracts in the future.
- There is a lot of code duplication, especially between `AccountRulesList` and `NodeRulesList`. Some of it could perhaps be factored out into a common base contract.
- All administrators have equal rights in the system. This means that any administrator can remove all the other administrators or change any of the contracts. This limits the contracts' usefulness to situations where all administrators trust each other.

5 Style Recommendations

We recommend implementing the following changes to improve the maintainability and readability of the code.

5.1 Use `0` to represent a `bytes32` zero value

In `contracts/Ingress.sol`, comparisons can be done with `0` rather than `0x0000000000000000000000000000000000000000000000000000000000000000`.

5.2 Use the `external` visibility rather than `public` when possible

If a function does not need to be called by the contract itself, it can be labelled external. This clarifies the intention, and may allow the compiler to improve gas efficiency.

efficiency.

The following functions can be labelled `external`:

- `AccountRules.getContractVersion()`
- `AccountRules.isReadOnly()`
- `AccountRules.enterReadOnly()`
- `AccountRules.exitReadOnly()`
- `AccountRules.transactionAllowed(address,address,uint256,uint256,uint256`
- `AccountRulesProxy.transactionAllowed(address,address,uint256,uint256,ui`
- `AccountRules.addAccount(address)`
- `AccountRules.removeAccount(address)`
- `AccountRules.getSize()`
- `AccountRules.getByIndex(uint256)`
- `AccountRules.getAccounts()`
- `AccountRules.addAccounts(address[])`
- `Ingress.setContractAddress(bytes32,address)`
- `Ingress.removeContract(bytes32)`
- `Ingress.getAllContractKeys()`
- `NodeRules.getContractVersion()`
- `NodeRules.isReadOnly()`
- `NodeRules.enterReadOnly()`
- `NodeRules.exitReadOnly()`
- `NodeRulesProxy.connectionAllowed(bytes32,bytes32,bytes16,uint16,bytes32`
- `NodeRules.connectionAllowed(bytes32,bytes32,bytes16,uint16,bytes32,byte`
- `NodeRules.addEnode(bytes32,bytes32,bytes16,uint16)`
- `NodeRules.removeEnode(bytes32,bytes32,bytes16,uint16)`
- `NodeRules.getSize()`
- `NodeRules.getByIndex(uint256)`
- `Admin.addAdmin(address)`
- `Admin.removeAdmin(address)`
- `Admin.getAdmins()`
- `Admin.addAdmins(address[])`
- `NodeIngress.getContractVersion()`

- `NodeIngress.emitRulesChangeEvent(bool)`
- `NodeIngress.connectionAllowed(bytes32,bytes32,bytes16,uint16,bytes32,by`
- `AccountIngress.getContractVersion()`
- `AccountIngress.emitRulesChangeEvent(bool)`
- `AccountIngress.transactionAllowed(address,address,uint256,uint256,uint2`

5.3 Declare variables as `constant` where possible

If a variable is not meant to change, it should be labelled as `constant`. This will save on gas costs, and increase safety.

The following storage variables can be made `constant`:

- `Ingress.ADMIN_CONTRACT`
- `Ingress.RULES_CONTRACT`
- `NodeIngress.version`
- `NodeRules.version`

5.4 Declare important literals as constants

`0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF` appears in two places in the code, and `0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF` appears once.

These literal values should be declared as `constant`s and given a name that describes their semantics.

5.5 Return explicitly

Solidity allows functions to end without a `return` statement, as happens in `NodeRules.getByIndex()` and `AccountRules.getByIndex()` if the specified index does not exist.

Functions are easier to use, analyze, and maintain when return statements are explicit in all code branches (or where all returns use a named return variable).

5.6 Give contracts more descriptive and consistent names

More descriptive contract naming would aid maintainability.

- The contracts that have names ending with `Proxy` are `Interface`s and not proxies.
- `AccountRulesList` could be named `AccountList`.
- `Admin`'s similarities to `NodeRules` and `AccountRules` may be more obvious if either *all* or *none* of these contract names used the `Rules` suffix.
- The purpose of `Exposed` contracts may be more immediately understood if the prefix was changed to `Test` and/or if they were moved into a folder containing only test contracts.

5.7 Give event parameters more descriptive names

The meaning of the `adminAdded` parameter in the `AdminAdded` event is not obvious. The log might be better named `AdminAddRequested`, and/or the parameter might be better named `successful`.

The same feedback applies to the first parameters in the following events:

- `AdminRemoved`
- `NodeAdded`
- `NodeRemoved`
- `AccountAdded`
- `AccountRemoved`

6 Security Specification

This section describes, **from a security perspective**, the expected behavior of the system under audit. It is not a substitute for documentation. The purpose of this section is to identify specific security properties that were validated by the audit team.

6.1 Actors

The relevant actors are as follows:

- **Nodes** – Only authorized nodes may connect to the network. Once connected, though, a node's operator can replace the Pantheon software with software of their choosing, which enables them to bypass any on-chain connection restrictions.
- **Administrators** – These are on-chain accounts that have the ability to change the rules contracts and to grant and revoke administrative priveleges to others.
- **Authorized accounts** – These accounts have been permitted, based on the on-chain permissioning system, to send transactions.
- **Other accounts** – Other accounts should be unable to send transactions.

6.2 Trust Model

In any smart contract system, it's important to identify what trust is expected/required between various actors. For this audit, we established the following trust model:

- Everyone must trust the administrators fully. They each have the ability to single-handedly take over the permissioning system, so all participants in the network, including the administrators themselves, must fully trust all the administrators.
- Node operators are free to make their nodes do whatever they want. The rest of the network should not have to trust that a given node is acting properly. The network should be robust to a malicious or buggy node.

6.3 Important Security Properties

The following is a non-exhaustive list of security properties that were verified in this audit:

- It's possible for each node to validate all incoming blocks and reject those that violate permissioning rules. This means there's no need to trust the other nodes in the network.

- Non-administrators cannot change the rules contracts or change any of the rules themselves.

7 Issues

Each issue has an assigned severity:

- **Minor** issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- **Medium** issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- **Major** issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- **Critical** issues are directly exploitable security vulnerabilities that need to be fixed.

7.1 `readOnlyMode` is ineffective and may result in a false sense of security Medium ✓ Addressed

Resolution
This was addressed in PegaSysEng/permissioning-smart-contracts@ `ed2d4a2` by adding comments to clarify that `readOnlyMode` is meant simply to prevent accidental changes during upgrades.

Description

`AccountRules` and `NodeRules` can both enter and exit a mode of operation called `readOnlyMode`.

The only effect of `readOnlyMode` is to prevent admins (who are the only users able to change rules) from changing rules.

Those same admins can disable `readOnlyMode`, so this mode will not prevent a determined actor from doing something they want to do.

Recommendation

Either `readOnlyMode` should be removed to prevent it from providing a false

sense of security, or the authorization required to toggle `readOnlyMode` should be separated from the authorization required to change rules.

7.2 `Ingress.setContractAddress()` can cause duplicate entries in `contractKeys` Medium ✓ Fixed

Resolution
This is fixed in PegaSysEng/permissioning-smart-contracts@ `faff726`.

Description

`setContractAddress()` checks `ContractDetails` existence by inspecting `contractAddress`. A `contractAddress` of `0` means that the contract does not already exist, and its name must be added to `contractKeys`:

code/contracts/Ingress.sol:L39-L62

```
function setContractAddress(bytes32 name, address addr) public returns
    require(name > 0x0000000000000000000000000000000000000000000000000
    require(isAuthorized(msg.sender), "Not authorized to update contra

    ContractDetails memory info = registry[name];
    // create info if it doesn't exist in the registry
    if (info.contractAddress == address(0)) {
        info = ContractDetails({
            owner: msg.sender,
            contractAddress: addr
        });

        // Update registry indexing
        contractKeys.push(name);
    } else {
        info.contractAddress = addr;
    }
     // update record in the registry
     registry[name] = info;
```

```
    emit RegistryUpdated(addr,name);

    return true;
}
```

If, however, a contract is actually added with the address `0`, which is currently allowed in the code, then the contract does already exists, and adding the name to `contractKeys` again will result in a duplicate.

Mitigation

An admin can call `removeContract` repeatedly with the same name to remove multiple duplicate entries.

Recommendation

Either disallow a contract address of `0` or check for existence via the `owner` field instead (which can never be `0`).

7.3 Use specific contract types instead of `address` where possible Minor ✓ Fixed

Resolution
This is fixed in PegaSysEng/permissioning-smart-contracts@ `05d33ae` and PegaSysEng/permissioning-smart-contracts@ `2728bac`.

Description

For clarity and to get more out of the Solidity type checker, it's generally preferred to use a specific contract type for variables rather than the generic `address`.

Examples

`AccountRules.ingressContractAddress` could instead be `AccountRules.ingressContract` and use the type `IngressContract`:

code/contracts/AccountRules.sol:L16

```
address private ingressContractAddress;
```

code/contracts/AccountRules.sol:L24

```
AccountIngress ingressContract = AccountIngress(ingressContractAddres
```

code/contracts/AccountRules.sol:L32

```
constructor (address ingressAddress) public {
```

This same pattern is found in `NodeRules`:

code/contracts/NodeRules.sol:L32

```
address private nodeIngressContractAddress;
```

Recommendation

Where possible, use a specific contract type rather than `address`.

7.4 `Ingress` should use a set Minor ✓ Fixed

Resolution
This is fixed in PegaSysEng/permissioning-smart-contracts@ `2978bd0` and PegaSysEng/permissioning-smart-contracts@ `f973035`.

Description

The `AdminList`, `AccountRulesList`, and `NodeRulesList` contracts have been recently rewritten to use a set. `Ingress` has the semantics of a set but has not been written the same way.

been written the same way.

This leads to some inefficiencies. In particular, `Ingress.removeContract` is an O(n) operation:

code/contracts/Ingress.sol:L68-L74

```
for (uint i = 0; i < contractKeys.length; i++) {
    // Delete the key from the array + mapping if it is present
    if (contractKeys[i] == name) {
        delete registry[contractKeys[i]];
        contractKeys[i] = contractKeys[contractKeys.length - 1];
        delete contractKeys[contractKeys.length - 1];
        contractKeys.length--;
```

Recommendation

Use the same set implementation for `Ingress`: an array of `ContractDetails` and a mapping of names to indexes in that array.

7.5 Use a specific Solidity compiler version Minor ✓ Fixed

Resolution

This is fixed in PegaSysEng/permissioning-smart-contracts@ `acf5a22` by pinning to Solidity 0.5.9 everywhere except the `Ingress` contract. Because the `Ingress` contract is hardcoded into the genesis block, it can't be easily changed. Non-critical issues like this one won't be addressed in that contract.

Description

A number of files use a "floating" pragma as follows:

```
pragma solidity >=0.4.22 <0.6.0;
```

It's better to use a specific Solidity compiler version (preferably a current version). This removes any confusion about which compiler was used when the contract is

This removes any confusion about which compiler was used when the contract is deployed, and it makes sure the code is never subjected to older compiler bugs.

It's still a good idea to upgrade the compiler version in the future as compiler bugs are fixed, but this way you must explicitly choose the new compiler version in your code when you do so.

Recommendation

Based on the Truffle configuration, the code is currently compiled with Solidity 0.5.9. Consider changing the existing `pragma`s to the following:

```
pragma solidity 0.5.9;
```

7.6 `ContractDetails.owner` is never read Minor ✓ Fixed

Resolution
This is fixed in PegaSysEng/permissioning-smart-contracts@ `d3f505e`.

Description

The `ContractDetails` struct used by `Ingress` contracts has an `owner` field that is written to, but it is never read.

code/contracts/Ingress.sol:L14-L19

```
struct ContractDetails {
    address owner;
    address contractAddress;
}

mapping(bytes32 => ContractDetails) registry;
```

Recommendation

If `owner` is not (yet) needed, the `ContractDetails` struct should be removed altogether and the type of `Ingress.registry` should change to `mapping(bytes32 => address)`.

```
mapping(bytes32 => address)
```

8 Tool-Based Analysis

Several tools were used to perform automated analysis of the reviewed contracts. These issues were reviewed by the audit team, and relevant issues are listed in the Issue Details section.

8.1 MythX

MythX is a security analysis API for Ethereum smart contracts. It performs multiple types of analysis, including fuzzing and symbolic execution, to detect many common vulnerability types. The tool was used for automated vulnerability discovery for all audited contracts and libraries. More details on MythX can be found at mythx.io.

Below is the raw output of the MythX vulnerability scan:

Summary

40 problems (0 errors, 40 warnings)

Warnings

SWC	count	visual
SWC-108	5	XXXXX
SWC-131	27	XXXXXXXXXXXXXXXXXXXX
SWC-110	3	XXX
SWC-128	3	XXX
SWC-123	2	XX

Details

AccountRules.sol - 7 problems (0 errors, 7 warnings)

Type	Line	Description	SWC
		The state variable visibility is not set. It is best practice	

Type	Line	Description	SWC
`Warning`	12:9	to set the visibility of state variables explicitly. The default visibility for "readOnlyMode" is internal. Other possible visibility values are public and private.	SWC-108
`Warning`	14:9	The state variable visibility is not set. It is best practice to set the visibility of state variables explicitly. The default visibility for "version" is internal. Other possible visibility values are public and private.	SWC-108
`Warning`	61:8	Unused local variable "" The local variable "" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	62:8	Unused local variable "" The local variable "" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	63:8	Unused local variable "" The local variable "" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	64:8	Unused local variable "" The local variable "" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	65:8	Unused local variable "" The local variable "" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131

AccountRulesList.sol - 2 problems (0 errors, 2 warnings)

Type	Line	Description	SWC
`Warning`	15:4	A reachable exception has been detected. It is possible to trigger an exception (opcode 0xfe). Exceptions can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. Note that explicit `assert()` should only be used to check invariants. Use `require()` for regular input checking.	SWC-110
		Potential denial-of-service if block gas limit is reached. A	

Type	Line	Description	SWC
`Warning`	36:8	storage modification is executed in a loop. Be aware that the transaction may fail to execute if the loop is unbounded and the necessary gas exceeds the block gas limit.	SWC-128

AccountRulesProxy.sol - 12 problems (0 errors, 12 warnings)

Type	Line	Description	SWC
`Warning`	5:8	Unused local variable "sender" The local variable "sender" is created within the contract "AccountRulesProxy" but does not seem to be used anywhere.	SWC-131
`Warning`	5:8	Unused local variable "sender" The local variable "sender" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	6:8	Unused local variable "target" The local variable "target" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	6:8	Unused local variable "target" The local variable "target" is created within the contract "AccountRulesProxy" but does not seem to be used anywhere.	SWC-131
`Warning`	7:8	Unused local variable "value" The local variable "value" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	7:8	Unused local variable "value" The local variable "value" is created within the contract "AccountRulesProxy" but does not seem to be used anywhere.	SWC-131
`Warning`	8:8	Unused local variable "gasPrice" The local variable "gasPrice" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	8:8	Unused local variable "gasPrice" The local variable "gasPrice" is created within the contract "AccountRulesProxy" but does not seem to be used anywhere.	SWC-131

		Unused local variable "gasLimit" The local variable	SWC-

Type	Line	Description	SWC
`Warning`	9:8	"gasLimit" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	9:8	Unused local variable "gasLimit" The local variable "gasLimit" is created within the contract "AccountRulesProxy" but does not seem to be used anywhere.	SWC-131
`Warning`	10:8	Unused local variable "payload" The local variable "payload" is created within the contract "AccountRules" but does not seem to be used anywhere.	SWC-131
`Warning`	10:8	Unused local variable "payload" The local variable "payload" is created within the contract "AccountRulesProxy" but does not seem to be used anywhere.	SWC-131

AdminList.sol - 3 problems (0 errors, 3 warnings)

Type	Line	Description	SWC
`Warning`	17:4	A reachable exception has been detected. It is possible to trigger an exception (opcode 0xfe). Exceptions can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. Note that explicit `assert()` should only be used to check invariants. Use `require()` for regular input checking.	SWC-110
`Warning`	38:8	Potential denial-of-service if block gas limit is reached. A storage modification is executed in a loop. Be aware that the transaction may fail to execute if the loop is unbounded and the necessary gas exceeds the block gas limit.	SWC-128
`Warning`	42:23	Potential denial-of-service if block gas limit is reached. A storage modification is executed in a loop. Be aware that the transaction may fail to execute if the loop is unbounded and the necessary gas exceeds the block gas limit.	SWC-128

AdminProxy.sol - 3 problems (0 errors, 3 warnings)

Type	Line	Description	SWC
		precondition violation A precondition was violated.	

Type	Line	Description	SWC
`Warning`	4:16	Make sure valid inputs are provided to both callees (e.g, via passed arguments) and callers (e.g., via return values).	SWC-123
`Warning`	4:26	Unused local variable "source" The local variable "source" is created within the contract "Admin" but does not seem to be used anywhere.	SWC-131
`Warning`	4:26	Unused local variable "source" The local variable "source" is created within the contract "AdminProxy" but does not seem to be used anywhere.	SWC-131

Ingress.sol - 3 problems (0 errors, 3 warnings)

Type	Line	Description	SWC
`Warning`	12:14	The state variable visibility is not set. It is best practice to set the visibility of state variables explicitly. The default visibility for "contractKeys" is internal. Other possible visibility values are public and private.	SWC-108
`Warning`	19:40	The state variable visibility is not set. It is best practice to set the visibility of state variables explicitly. The default visibility for "registry" is internal. Other possible visibility values are public and private.	SWC-108
`Warning`	35:19	precondition violation A precondition was violated. Make sure valid inputs are provided to both callees (e.g, via passed arguments) and callers (e.g., via return values).	SWC-123

NodeRulesList.sol - 1 problem (0 errors, 1 warning)

Type	Line	Description	SWC
`Warning`	15:4	assertion violation An assertion was violated. Make sure your program logic is correct (e.g., no division by zero) and that you add appropriate validation for inputs from both callers (e.g, passed arguments) and callees (e.g., return values).	SWC-110

NodeIngress.sol - 1 problem (0 errors, 1 warning)

Type	Line	Description	SWC
		The state variable visibility is not set. It is best practice	

Type	Line	Description	SWC
Warning	5:9	to set the visibility of state variables explicitly. The default visibility for "version" is internal. Other possible visibility values are public and private.	SWC-108

NodeRulesProxy.sol - 8 problems (0 errors, 8 warnings)

Type	Line	Description	SWC
Warning	5:8	Unused local variable "sourceEnodeHigh" The local variable "sourceEnodeHigh" is created within the contract "NodeRulesProxy" but does not seem to be used anywhere.	SWC-131
Warning	6:8	Unused local variable "sourceEnodeLow" The local variable "sourceEnodeLow" is created within the contract "NodeRulesProxy" but does not seem to be used anywhere.	SWC-131
Warning	7:8	Unused local variable "sourceEnodeIp" The local variable "sourceEnodeIp" is created within the contract "NodeRulesProxy" but does not seem to be used anywhere.	SWC-131
Warning	8:8	Unused local variable "sourceEnodePort" The local variable "sourceEnodePort" is created within the contract "NodeRulesProxy" but does not seem to be used anywhere.	SWC-131
Warning	9:8	Unused local variable "destinationEnodeHigh" The local variable "destinationEnodeHigh" is created within the contract "NodeRulesProxy" but does not seem to be used anywhere.	SWC-131
Warning	10:8	Unused local variable "destinationEnodeLow" The local variable "destinationEnodeLow" is created within the contract "NodeRulesProxy" but does not seem to be used anywhere.	SWC-131
Warning	11:8	Unused local variable "destinationEnodeIp" The local variable "destinationEnodeIp" is created within the contract "NodeRulesProxy" but does not seem to be used anywhere.	SWC-131
		Unused local variable "destinationEnodePort" The local	

Type	Line	Description	SWC
Warning	12:8	variable "destinationEnodePort" is created within the contract "NodeRulesProxy" but does not seem to be used anywhere.	SWC-131

AccountIngress.sol - 0 problems

Admin.sol - 0 problems

ExposedAccountRulesList.sol - 0 problems

ExposedAdminList.sol - 0 problems

ExposedNodeRulesList.sol - 0 problems

```
Generated on Thu Aug 29 2019 15:16:37 GMT-0700 (Pacific Daylight Time)

MythX Logs:

AccountRules.sol
UUID: 6db36465-5d19-43b8-8318-20d038616ffb
info: skipped automated fuzz testing due to incompatible bytecode inpu

AccountRulesList.sol
UUID: 17faa2da-60ed-4e9c-8f76-c9d87ebfa025

AccountRulesProxy.sol
UUID: 0579eb33-82ef-4ac7-99e1-948ba46955df

Admin.sol
UUID: da4012ea-98e3-4116-9ee9-896da7904e7c

AdminList.sol
UUID: 6a5da947-d87d-4f3a-b3e6-94d76712aa73

AdminProxy.sol
UUID: ef18baac-d986-4bcd-aefc-1d0801e214d2

ExposedAccountRulesList.sol
UUID: 706738e2-35a4-4def-b7c4-f680920db1a1
```

```
ExposedAdminList.sol
UUID: ea78f05f-c0f2-46e0-84eb-0168d35fccc4

ExposedNodeRulesList.sol
UUID: 33d4d1a4-2f85-4d6d-99c7-dd76c738d305

Ingress.sol
UUID: 162745bf-308e-4cc8-a07b-b5e1564f7764

NodeIngress.sol
UUID: a40eebe4-d51e-40fd-8b0b-913491e63411

NodeRulesList.sol
UUID: c5d7bd11-591d-4bd6-8364-883d8db35bb9

NodeRulesProxy.sol
UUID: 851c3349-b9f2-459c-a3ec-5bbd7cb6d616
```

8.2 Ethlint

Ethlint is an open source project for linting Solidity code. Only security-related issues were reviewed by the audit team.

Ethlint didn't find any issues.

8.3 Surya

Surya is an utility tool for smart contract systems. It provides a number of visual outputs and information about structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

Below is a complete list of functions with their visibility and modifiers:

Files Description Table

File Name	SHA-1 Hash
AccountIngress.sol	57207a6878535bc2f3d40216d96f07eef9bbdfd9

File Name	SHA-1 Hash
AccountRulesList.sol	73ffd92be5b6c3b1e18d1b8e0344dac578c9aa31
Admin.sol	e13931323093f1555f4dfcc74fad6a2c457c1082
AdminProxy.sol	eecd073b4e05a4445fb00888074b48c443c5bbf4
Ingress.sol	b0fcff06fa7d55136cfe483331280e4e9bb9def4
NodeIngress.sol	3f46f78e4c1b9a546287135a13ffa303f62a826b
NodeRulesList.sol	fa9382c4cf3f4d800aa3d0e89bb9a712d5aa5f0c
AccountRules.sol	c730212300e070ed22b1490f6e67347d1f36c051
AccountRulesProxy.sol	1024d00149ee0258f5ee4c0671a09ada723c3645
AdminList.sol	0304e06bfc4c87abc4d2f4c0361633590c5ef830
NodeRules.sol	8f0dc9efd5bc09a8c6346495e23a398c907baf21
NodeRulesProxy.sol	01967d8481a3f1497ecdfcfcd5e7dd2ea9f9c17e

Contracts Description Table

Contract	Type	Bases	
└	**Function Name**	**Visibility**	**Mutability**
AccountIngress	Implementation	Ingress	
└	getContractVersion	Public ❗	
└	emitRulesChangeEvent	Public ❗	🛑
└	transactionAllowed	Public ❗	
AccountRulesList	Implementation		
└	size	Internal 🔒	
└	exists	Internal 🔒	
└	add	Internal 🔒	🛑
└	addAll	Internal 🔒	🛑
└	remove	Internal 🔒	🛑
Admin	Implementation	AdminProxy,	

Contract	Type	Bases	
└	<Constructor>	Public ❗	🛑
└	isAuthorized	Public ❗	
└	addAdmin	Public ❗	🛑
└	removeAdmin	Public ❗	🛑
└	getAdmins	Public ❗	
└	addAdmins	Public ❗	🛑
AdminProxy	Interface		
└	isAuthorized	External ❗	
Ingress	Implementation		
└	getContractAddress	Public ❗	
└	isAuthorized	Public ❗	
└	setContractAddress	Public ❗	🛑
└	removeContract	Public ❗	🛑
└	getAllContractKeys	Public ❗	
NodeIngress	Implementation	Ingress	
└	getContractVersion	Public ❗	
└	emitRulesChangeEvent	Public ❗	🛑
└	connectionAllowed	Public ❗	
NodeRulesList	Implementation		
└	calculateKey	Internal 🔒	
└	size	Internal 🔒	
└	exists	Internal 🔒	
└	add	Internal 🔒	🛑
└	remove	Internal 🔒	🛑

Contract	Type	Bases	
AccountRules	Implementation	AccountRulesProxy, AccountRulesList	
└	<Constructor>	Public ❗	🛑
└	getContractVersion	Public ❗	
└	isReadOnly	Public ❗	
└	enterReadOnly	Public ❗	🛑
└	exitReadOnly	Public ❗	🛑
└	transactionAllowed	Public ❗	
└	accountInWhitelist	Public ❗	
└	addAccount	Public ❗	🛑
└	removeAccount	Public ❗	🛑
└	getSize	Public ❗	
└	getByIndex	Public ❗	
└	getAccounts	Public ❗	
└	addAccounts	Public ❗	🛑
AccountRulesProxy	Interface		
└	transactionAllowed	External ❗	
AdminList	Implementation		
└	size	Internal 🔒	
└	exists	Internal 🔒	
└	add	Internal 🔒	🛑
└	addAll	Internal 🔒	🛑
└	remove	Internal 🔒	🛑

NodeRules	Implementation	NodeRulesProxy,	

Contract	Type	Bases	
└	<Constructor>	Public ❗	🛑
└	getContractVersion	Public ❗	
└	isReadOnly	Public ❗	
└	enterReadOnly	Public ❗	🛑
└	exitReadOnly	Public ❗	🛑
└	connectionAllowed	Public ❗	
└	enodeInWhitelist	Public ❗	
└	addEnode	Public ❗	🛑
└	removeEnode	Public ❗	🛑
└	getSize	Public ❗	
└	getByIndex	Public ❗	
└	triggerRulesChangeEvent	Public ❗	🛑
NodeRulesProxy	Interface		
└	connectionAllowed	External ❗	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

8.4 Slither

Slither is a Solidity static analysis framework written in Python 3. It runs a suite of vulnerability detectors.

Below is the raw output of the Slither scan:

SLITHER

```
INFO:Detectors:
Pragma version ">=0.4.22<0.6.0" allows old versions (ExposedAdminList
Pragma version ">=0.4.22<0.6.0" allows old versions (AccountRules.sol#
Pragma version ">=0.4.22<0.6.0" allows old versions (AccountRulesList
Pragma version ">=0.4.22<0.6.0" allows old versions (Ingress.sol#1)
Pragma version ">=0.4.22<0.6.0" allows old versions (NodeRules.sol#1)
Pragma version ">=0.4.22<0.6.0" allows old versions (AdminProxy.sol#1
Pragma version ">=0.4.22<0.6.0" allows old versions (NodeRulesProxy.s
Pragma version ">=0.4.22<0.6.0" allows old versions (ExposedNodeRules
Pragma version ">=0.4.22<0.6.0" allows old versions (AdminList.sol#1)
Pragma version ">=0.4.22<0.6.0" allows old versions (ExposedAccountRu
Pragma version ">=0.4.22<0.6.0" allows old versions (Admin.sol#1)
Pragma version ">=0.4.22<0.6.0" allows old versions (Migrations.sol#1
Pragma version ">=0.4.22<0.6.0" allows old versions (NodeIngress.sol#
Pragma version ">=0.4.22<0.6.0" allows old versions (AccountRulesProx
Pragma version ">=0.4.22<0.6.0" allows old versions (AccountIngress.s
Pragma version ">=0.4.22<0.6.0" allows old versions (NodeRulesList.so
Reference: https://github.com/crytic/slither/wiki/Detector-Documentat
INFO:Detectors:
Function 'ExposedAdminList._size()' (ExposedAdminList.sol#9-11) is no
Function 'ExposedAdminList._exists(address)' (ExposedAdminList.sol#13
Parameter '_address' of _address (ExposedAdminList.sol#13) is not in
Function 'ExposedAdminList._add(address)' (ExposedAdminList.sol#17-19
Parameter '_address' of _address (ExposedAdminList.sol#17) is not in
Function 'ExposedAdminList._remove(address)' (ExposedAdminList.sol#21
Parameter '_address' of _address (ExposedAdminList.sol#21) is not in
Function 'ExposedAdminList._addBatch(address[])' (ExposedAdminList.so
Parameter '_addresses' of _addresses (ExposedAdminList.sol#25) is not
Parameter '_account' of _account (AccountRules.sol#77) is not in mixe
Parameter '_account' of _account (AccountRulesList.sol#22) is not in
Parameter '_account' of _account (AccountRulesList.sol#26) is not in
Parameter '_account' of _account (AccountRulesList.sol#45) is not in
Variable 'Ingress.RULES_CONTRACT' (Ingress.sol#8) is not in mixedCase
Variable 'Ingress.ADMIN_CONTRACT' (Ingress.sol#9) is not in mixedCase
Function 'ExposedNodeRulesList._calculateKey(bytes32,bytes32,bytes16,
Parameter '_enodeHigh' of _enodeHigh (ExposedNodeRulesList.sol#8) is
Parameter '_enodeLow' of _enodeLow (ExposedNodeRulesList.sol#8) is no
Parameter '_ip' of _ip (ExposedNodeRulesList.sol#8) is not in mixedCa
```

```
Parameter '_port' of _port (ExposedNodeRulesList.sol#8) is not in mixe
Function 'ExposedNodeRulesList._size()' (ExposedNodeRulesList.sol#12-1
Function 'ExposedNodeRulesList._exists(bytes32,bytes32,bytes16,uint16)
Parameter '_enodeHigh' of _enodeHigh (ExposedNodeRulesList.sol#16) is
Parameter '_enodeLow' of _enodeLow (ExposedNodeRulesList.sol#16) is no
Parameter '_ip' of _ip (ExposedNodeRulesList.sol#16) is not in mixedCa
Parameter '_port' of _port (ExposedNodeRulesList.sol#16) is not in mix
Function 'ExposedNodeRulesList._add(bytes32,bytes32,bytes16,uint16)' (
Parameter '_enodeHigh' of _enodeHigh (ExposedNodeRulesList.sol#20) is
Parameter '_enodeLow' of _enodeLow (ExposedNodeRulesList.sol#20) is no
Parameter '_ip' of _ip (ExposedNodeRulesList.sol#20) is not in mixedCa
Parameter '_port' of _port (ExposedNodeRulesList.sol#20) is not in mix
Function 'ExposedNodeRulesList._remove(bytes32,bytes32,bytes16,uint16)
Parameter '_enodeHigh' of _enodeHigh (ExposedNodeRulesList.sol#24) is
Parameter '_enodeLow' of _enodeLow (ExposedNodeRulesList.sol#24) is no
Parameter '_ip' of _ip (ExposedNodeRulesList.sol#24) is not in mixedCa
Parameter '_port' of _port (ExposedNodeRulesList.sol#24) is not in mix
Parameter '_account' of _account (AdminList.sol#24) is not in mixedCas
Parameter '_account' of _account (AdminList.sol#28) is not in mixedCas
Parameter '_account' of _account (AdminList.sol#56) is not in mixedCas
Function 'ExposedAccountRulesList._size()' (ExposedAccountRulesList.so
Function 'ExposedAccountRulesList._exists(address)' (ExposedAccountRul
Parameter '_account' of _account (ExposedAccountRulesList.sol#12) is n
Function 'ExposedAccountRulesList._add(address)' (ExposedAccountRulesL
Parameter '_account' of _account (ExposedAccountRulesList.sol#16) is n
Function 'ExposedAccountRulesList._addAll(address[])' (ExposedAccountR
Function 'ExposedAccountRulesList._remove(address)' (ExposedAccountRul
Parameter '_account' of _account (ExposedAccountRulesList.sol#24) is n
Parameter '_address' of _address (Admin.sol#22) is not in mixedCase
Parameter '_address' of _address (Admin.sol#26) is not in mixedCase
Parameter '_address' of _address (Admin.sol#38) is not in mixedCase
Parameter 'new_address' of new_address (Migrations.sol#20) is not in m
Variable 'Migrations.last_completed_migration' (Migrations.sol#6) is n
Struct 'NodeRulesList.enode' (NodeRulesList.sol#8-13) is not in CapWor
Parameter '_enodeHigh' of _enodeHigh (NodeRulesList.sol#18) is not in
Parameter '_enodeLow' of _enodeLow (NodeRulesList.sol#18) is not in mi
Parameter '_ip' of _ip (NodeRulesList.sol#18) is not in mixedCase

Parameter '_port' of _port (NodeRulesList.sol#18) is not in mixedCase
Parameter '_enodeHigh' of _enodeHigh (NodeRulesList.sol#26) is not in
Parameter '_enodeLow' of _enodeLow (NodeRulesList.sol#26) is not in mi
```

```
Parameter '_ip' of _ip (NodeRulesList.sol#26) is not in mixedCase
Parameter '_port' of _port (NodeRulesList.sol#26) is not in mixedCase
Parameter '_enodeHigh' of _enodeHigh (NodeRulesList.sol#30) is not in
Parameter '_enodeLow' of _enodeLow (NodeRulesList.sol#30) is not in m
Parameter '_ip' of _ip (NodeRulesList.sol#30) is not in mixedCase
Parameter '_port' of _port (NodeRulesList.sol#30) is not in mixedCase
Parameter '_enodeHigh' of _enodeHigh (NodeRulesList.sol#39) is not in
Parameter '_enodeLow' of _enodeLow (NodeRulesList.sol#39) is not in m
Parameter '_ip' of _ip (NodeRulesList.sol#39) is not in mixedCase
Parameter '_port' of _port (NodeRulesList.sol#39) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentat
INFO:Detectors:
AccountRules.slitherConstructorVariables (AccountRules.sol#9-113) uses
        - version = 1000000
NodeRules.slitherConstructorVariables (NodeRules.sol#9-170) uses liter
        - version = 1000000
NodeIngress.getContractAddress (Ingress.sol#26-29) uses literals with
        - require(bool,string)(name > 0x0000000000000000000000000000000
NodeIngress.setContractAddress (Ingress.sol#39-62) uses literals with
        - require(bool,string)(name > 0x0000000000000000000000000000000
NodeIngress.removeContract (Ingress.sol#64-81) uses literals with too
        - require(bool,string)(name > 0x0000000000000000000000000000000
NodeIngress.slitherConstructorVariables (NodeIngress.sol#7-49) uses l
        - RULES_CONTRACT = 0x72756c6573000000000000000000000000000000
NodeIngress.slitherConstructorVariables (NodeIngress.sol#7-49) uses l
        - ADMIN_CONTRACT = 0x61646d696e697374726174696f6e000000000000
NodeIngress.slitherConstructorVariables (NodeIngress.sol#7-49) uses l
        - version = 1000000
AccountIngress.getContractAddress (Ingress.sol#26-29) uses literals w
        - require(bool,string)(name > 0x0000000000000000000000000000000
AccountIngress.setContractAddress (Ingress.sol#39-62) uses literals w
        - require(bool,string)(name > 0x0000000000000000000000000000000
AccountIngress.removeContract (Ingress.sol#64-81) uses literals with
        - require(bool,string)(name > 0x0000000000000000000000000000000
AccountIngress.slitherConstructorVariables (AccountIngress.sol#7-40) u
        - RULES_CONTRACT = 0x72756c6573000000000000000000000000000000
AccountIngress.slitherConstructorVariables (AccountIngress.sol#7-40) u
        - ADMIN_CONTRACT = 0x61646d696e697374726174696f6e000000000000
AccountIngress.slitherConstructorVariables (AccountIngress.sol#7-40) u
        - version = 1000000
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentat
INFO:Detectors:
AccountIngress.version should be constant (AccountIngress.sol#9)
AccountRules.version should be constant (AccountRules.sol#14)
Ingress.ADMIN_CONTRACT should be constant (Ingress.sol#9)
Ingress.RULES_CONTRACT should be constant (Ingress.sol#8)
NodeIngress.version should be constant (NodeIngress.sol#9)
NodeRules.version should be constant (NodeRules.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentat
INFO:Detectors:
ExposedAdminList._size() (ExposedAdminList.sol#9-11) should be declare
ExposedAdminList._exists(address) (ExposedAdminList.sol#13-15) should
ExposedAdminList._add(address) (ExposedAdminList.sol#17-19) should be
ExposedAdminList._remove(address) (ExposedAdminList.sol#21-23) should
ExposedAdminList._addBatch(address[]) (ExposedAdminList.sol#25-27) sho
AccountRules.getContractVersion() (AccountRules.sol#38-40) should be d
AccountRules.isReadOnly() (AccountRules.sol#43-45) should be declared
AccountRules.enterReadOnly() (AccountRules.sol#47-51) should be decla
AccountRules.exitReadOnly() (AccountRules.sol#53-57) should be declare
AccountRules.transactionAllowed(address,address,uint256,uint256,uint25
AccountRulesProxy.transactionAllowed(address,address,uint256,uint256,u
AccountRules.addAccount(address) (AccountRules.sol#82-88) should be de
AccountRules.removeAccount(address) (AccountRules.sol#90-96) should be
AccountRules.getSize() (AccountRules.sol#98-100) should be declared ex
AccountRules.getByIndex(uint256) (AccountRules.sol#102-104) should be
AccountRules.getAccounts() (AccountRules.sol#106-108) should be decla
AccountRules.addAccounts(address[]) (AccountRules.sol#110-112) should
Ingress.setContractAddress(bytes32,address) (Ingress.sol#39-62) should
Ingress.removeContract(bytes32) (Ingress.sol#64-81) should be declared
Ingress.getAllContractKeys() (Ingress.sol#83-85) should be declared ex
NodeRules.getContractVersion() (NodeRules.sol#53-55) should be declare
NodeRules.isReadOnly() (NodeRules.sol#58-60) should be declared extern
NodeRules.enterReadOnly() (NodeRules.sol#62-66) should be declared ext
NodeRules.exitReadOnly() (NodeRules.sol#68-72) should be declared exte
NodeRulesProxy.connectionAllowed(bytes32,bytes32,bytes16,uint16,bytes3
NodeRules.connectionAllowed(bytes32,bytes32,bytes16,uint16,bytes32,by
NodeRules.addEnode(bytes32,bytes32,bytes16,uint16) (NodeRules.sol#112-

NodeRules.removeEnode(bytes32,bytes32,bytes16,uint16) (NodeRules.sol#
NodeRules.getSize() (NodeRules.sol#156-158) should be declared externa
NodeRules.getByIndex(uint256) (NodeRules.sol#160-165) should be decla
```

```
ExposedNodeRulesList._calculateKey(bytes32,bytes32,bytes16,uint16) (E
ExposedNodeRulesList._size() (ExposedNodeRulesList.sol#12-14) should b
ExposedNodeRulesList._exists(bytes32,bytes32,bytes16,uint16) (Exposed
ExposedNodeRulesList._add(bytes32,bytes32,bytes16,uint16) (ExposedNode
ExposedNodeRulesList._remove(bytes32,bytes32,bytes16,uint16) (Exposed
ExposedAccountRulesList._size() (ExposedAccountRulesList.sol#8-10) sho
ExposedAccountRulesList._exists(address) (ExposedAccountRulesList.sol
ExposedAccountRulesList._add(address) (ExposedAccountRulesList.sol#16
ExposedAccountRulesList._addAll(address[]) (ExposedAccountRulesList.s
ExposedAccountRulesList._remove(address) (ExposedAccountRulesList.sol
Admin.addAdmin(address) (Admin.sol#26-36) should be declared external
Admin.removeAdmin(address) (Admin.sol#38-42) should be declared exter
Admin.getAdmins() (Admin.sol#44-46) should be declared external
Admin.addAdmins(address[]) (Admin.sol#48-50) should be declared exter
Migrations.setCompleted(uint256) (Migrations.sol#16-18) should be dec
Migrations.upgrade(address) (Migrations.sol#20-23) should be declared
NodeIngress.getContractVersion() (NodeIngress.sol#15-17) should be de
NodeIngress.emitRulesChangeEvent(bool) (NodeIngress.sol#19-22) should
NodeIngress.connectionAllowed(bytes32,bytes32,bytes16,uint16,bytes32,
AccountIngress.getContractVersion() (AccountIngress.sol#15-17) should
AccountIngress.emitRulesChangeEvent(bool) (AccountIngress.sol#19-22) s
AccountIngress.transactionAllowed(address,address,uint256,uint256,uin
Reference: https://github.com/crytic/slither/wiki/Detector-Documentat
INFO:Slither:. analyzed (16 contracts), 157 result(s) found
```

Appendix 1 - Disclosure

ConsenSys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect

provides any warranty or representation to any Third Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself 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 Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites’ owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use

of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software

by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.

2019 © ConsenSys Privacy Policy