Polygon PoS Portal

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **June 13th - July 7th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	07/07/2021	Ferran Celades
0.9	Document Creation	07/12/2021	Ferran Celades
1.0	Final Review	07/12/2021	Gabi Urrutia
1.1	Remediation Plan	07/21/2021	Ferran Celades
1.1	Remediation Plan	07/26/2021	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Ferran Celades	Halborn	Ferran.Celades@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Polygon engaged Halborn to conduct a security assessment on their Smart contracts beginning on June 13th, 2021 and ending July 7th, 2021. The security assessment was scoped to the smart contract provided in the Github repository PoS Portal for Polygon and an audit of the security risk and implications regarding the changes introduced by the development team at Polygon prior to its production release shortly following the assessments deadline.

1.2 AUDIT SUMMARY

The team at Halborn was provided one month for the engagement and assigned two full time security engineers to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit to achieve the following:

- Ensure that smart contract functions are intended.
- Identify potential security issues with the smart contracts.

Though this security audit's outcome is satisfactory, only the most essential aspects were tested and verified to achieve objectives and deliverables set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure smart-contract development.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to

the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process,and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart Contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions(solgraph)
- Manual Assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes (brownie console and manual deployments on Ganache)
- Manual testing by custom Python scripts.
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions.(Slither)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident, and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. It's quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.

2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the smart contracts:

/child/
- ChildChainManager/ChildChainManager.sol
- ChildChainManager/ChildChainManagerProxy.sol
- ChildChainManager/IChildChainManager.sol
- ChildToken/ChildERC1155.sol
- ChildToken/ChildERC20.sol
- ChildToken/ChildERC721.sol
- ChildToken/ChildMintableERC1155.sol
- ChildToken/ChildMintableERC20.sol
- ChildToken/ChildMintableERC721.sol
- ChildToken/DappTokens/UChildDAI.sol
- ChildToken/IChildToken.sol
- ChildToken/MaticWETH.sol
- ChildToken/UpgradeableChildERC20/ERC20.sol
- ChildToken/UpgradeableChildERC20/UChildERC20.sol
- ChildToken/UpgradeableChildERC20/UChildERC20Proxy.sol
- IStateReceiver.sol

/common/
- AccessControlMixin.sol
- ContextMixin.sol
- EIP712Base.sol
- Initializable.sol
- NativeMetaTransaction.sol
- Proxy/IERCProxy.sol
- Proxy/Proxy.sol
- Proxy/UpgradableProxy.sol

/lib/
lib/Merkle.sol
lib/MerklePatriciaProof.sol
lib/RLPReader.sol

/root/
- ICheckpointManager.sol
- MockCheckpointManager.sol
- RootChainManager/IRootChainManager.sol
- RootChainManager/RootChainManager.sol
- RootChainManager/RootChainManagerProxy.sol
- RootChainManager/RootChainManagerStorage.sol
- RootToken/DummyERC1155.sol
- RootToken/DummyERC20.sol
- RootToken/DummyERC721.sol
- RootToken/DummyMintableERC1155.sol
- RootToken/DummyMintableERC20.sol
- RootToken/DummyMintableERC721.sol
- RootToken/IMintableERC1155.sol
- RootToken/IMintableERC20.sol
- RootToken/IMintableERC721.sol
- RootToken/IRootERC721.sol
- StateSender/DummyStateSender.sol
- StateSender/IStateSender.sol
- TokenPredicates/ERC1155Predicate.sol
- TokenPredicates/ERC1155PredicateProxy.sol
- TokenPredicates/ERC20Predicate.sol
- TokenPredicates/ERC20PredicateProxy.sol
- TokenPredicates/ERC721Predicate.sol
- TokenPredicates/ERC721PredicateProxy.sol
- TokenPredicates/EtherPredicate.sol
- TokenPredicates/EtherPredicateProxy.sol
- TokenPredicates/ITokenPredicate.sol
- TokenPredicates/MintableERC1155Predicate.sol
- TokenPredicates/MintableERC1155PredicateProxy.sol
- TokenPredicates/MintableERC20Predicate.sol
- TokenPredicates/MintableERC20PredicateProxy.sol
- TokenPredicates/MintableERC721Predicate.sol
- TokenPredicates/MintableERC721PredicateProxy.sol

/tunnel/
- BaseChildTunnel.sol
- BaseRootTunnel.sol

- ChildTunnel.sol
- RootTunnel.sol

Commit ID: d06271188412a91ab9e4bdea4bbbfeb6cb9d7669

Fixed Commit ID: b62515b73084900434c24d4b743bf3aa5fe81af4

OUT-OF-SCOPE:

Contracts not listed above, external libraries and economic attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	4	6

LIKELIHOOD

IMPACT

			(HAL-01)	
(HAL-04)				
(HAL-07)	(HAL-02) (HAL-03) (HAL-05)			
(HAL-08) (HAL-09) (HAL-10) (HAL-11)	(HAL-06)			

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
UNCHECKED TRANSFERS AND AMOUNTS	High	NOT SOLVED - 07/21/2021
INVALID ROLE-BASED ACCESS CONTROL MODIFIER	Low	SOLVED - 07/21/2021
MISSING ADDRESS VALIDATION	Low	SOLVED - 07/21/2021
FLOATING PRAGMA AND VERSION MISMATCH	Low	SOLVED - 07/21/2021
POSSIBLE MISUSE OF ROLE-BASED ACCESS CONTROL POLICY	Low	RISK ACCEPTED - 07/21/2021
POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	RISK ACCEPTED - 07/21/2021
DOS WITH BLOCK GAS LIMIT	Informational	SOLVED - 07/21/2021
GAS SAVING WHEN COPYING MEMORY	Informational	SOLVED - 07/21/2021
UNEXPECTED BEHAVIOUR ON CONTEXT MIXIN	Informational	ACKNOWLEDGED - 07/21/2021
CODE REDUNDANCY	Informational	RISK ACCEPTED - 07/21/2021
MINTING IS ALLOWED TO ANY CALLER	Informational	SOLVED - 07/21/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) UNCHECKED TRANSFERS AND AMOUNTS - HIGH

Description:

Several tokens do not revert in case of failure and return false. If one of these tokens is used in PoS portal transfers, deposit will not revert if the transfer fails, and an invalid state can be synced to the Bor chain. Furthermore, deflationary or inflationary tokens that transfer a different amount than the requested can have invalid synced states. As an example, transferred deflationary or inflationary tokens would cause the original amount to be synced to the Bor network causing a mismatched amount sync between the chains.

As seen in Listing 1 and Listing 2, the synced state does only take into consideration the requested amount and not the real transferred amount.

Listing 1: RootChainManager.sol depositFor function (Lines 312)

```
306 ITokenPredicate(predicateAddress).lockTokens(
307     _msgSender(),
308     user,
309     rootToken,
310     depositData
311 );
312 bytes memory syncData = abi.encode(user, rootToken, depositData);
313 _stateSender.syncState(
314     childChainManagerAddress,
315     abi.encode(DEPOSIT, syncData)
316 );
```

Listing 2: ERC20Predicate.sol (Lines 53)

```
41 function lockTokens(
42     address depositor,
43     address depositReceiver,
44     address rootToken,
45     bytes calldata depositData
46 )
```

```
47      external
48      override
49      only(MANAGER_ROLE)
50  {
51      uint256 amount = abi.decode(depositData, (uint256));
52      emit LockedERC20(depositor, depositReceiver, rootToken, amount
            );
53      IERC20(rootToken).safeTransferFrom(depositor, address(this),
            amount);
54  }
```

Code Location:

Listing 3: root/TokenPredicates/MintableERC20Predicate.sol (Lines 54)

```
45 function lockTokens(
46     address depositor,
47     address depositReceiver,
48     address rootToken,
49     bytes calldata depositData
50 ) external override only(MANAGER_ROLE) {
51     uint256 amount = abi.decode(depositData, (uint256));
52
53     emit LockedMintableERC20(depositor, depositReceiver, rootToken
           , amount);
54     IMintableERC20(rootToken).transferFrom(
55         depositor,
56         address(this),
57         amount
58     );
59 }
```

Listing 4: child/ChildToken/DappTokens/UChildDAI.sol (Lines 11,14,17)

```
10 function push(address usr, uint wad) external {
11     transferFrom(msg.sender, usr, wad);
12 }
13 function pull(address usr, uint wad) external {
14     transferFrom(usr, msg.sender, wad);
15 }
16 function move(address src, address dst, uint wad) external {
```

```
17     transferFrom(src, dst, wad);
18 }
```

Risk Level:

Likelihood - 4
Impact - 5

Recommendations:

It is recommended to use SafeERC20 on the both networks when possible and ensure that the transfer/transferFrom return value is checked. A custom function named safeTransferFrom can be implemented that checks the return value of the transferFrom function and makes sure that the funds were transferred. Furthermore, the before and after balance should be checked once the transfer is performed to validate differences between the requested amount and the transferred amount, possible in inflationary and deflationary tokens, before syncing to the Bor chain. This check has to be performed on all the Predicates contracts.

Remediation Plan:

NOT SOLVED: Recommendations for the transferFrom function were not applied. Polygon states that an open pull request Pull 82 does implement the fixes needed to address the inflationary and deflationary tokens issue. However, those changes are experimental and were not pushed yet.

3.2 (HAL-02) INVALID ROLE-BASED ACCESS CONTROL MODIFIER - LOW

Description:

On the RootChainManager contract on the registerPredicate function the comment states that the function should only be called by mappers, but the role is checked against the DEFAULT_ADMIN_ROLER by using only(DEFAULT_ADMIN_ROLE). .

Code Location:

Listing 5: RootChainManager.sol (Lines 151)

```
148 function registerPredicate(bytes32 tokenType, address
        predicateAddress)
149     external
150     override
151     only(DEFAULT_ADMIN_ROLE)
152 {
153     typeToPredicate[tokenType] = predicateAddress;
154     emit PredicateRegistered(tokenType, predicateAddress);
155 }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to change the modifier to only(MAPPER_ROLE) , or update the comment to reflect the current source code modifier.

Remediation Plan:

SOLVED: The comment was modified to reflect the current source code, stating that ADMIN permissions are required.

3.3 (HAL-03) MISSING ADDRESS VALIDATION - LOW

Description:

The RootChainManager.sol and BaseRootTunnel.sol contracts have lack of safety check inside their functions. Setters of address type parameters should include a zero-address check. Otherwise, contract functionality may become inaccessible, or tokens could be burnt forever. Furthermore, validating that the set address does conform to the interface ABI would prevent unexpected logic errors.

Code Location:

Listing 6: RootChainManager.sol

```
94 function setStateSender(address newStateSender)
95     external
96     only(DEFAULT_ADMIN_ROLE)
97 {
98     _stateSender = IStateSender(newStateSender);
99 }
```

Listing 7: RootChainManager.sol

```
114 function setCheckpointManager(address newCheckpointManager)
115     external
116     only(DEFAULT_ADMIN_ROLE)
117 {
118     _checkpointManager = ICheckpointManager(newCheckpointManager);
119 }
```

Listing 8: BaseRootTunnel.sol

```
43 function setStateSender(address newStateSender)
44     external
45     only(DEFAULT_ADMIN_ROLE)
46 {
```

```
47      stateSender = IStateSender(newStateSender);
48  }
```

Listing 9: BaseRootTunnel.sol

```
55  function setCheckpointManager(address newCheckpointManager)
56      external
57      only(DEFAULT_ADMIN_ROLE)
58  {
59      checkpointManager = ICheckpointManager(newCheckpointManager);
60  }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It recommended to add proper address validation when assigning a value to a variable from user-supplied data. Better yet, address white-listing/black-listing should be implemented in relevant functions if possible.

Remediation Plan:

SOLVED: Checks for non-zero addresses were added to the functions.

3.4 (HAL-04) FLOATING PRAGMA AND VERSION MISMATCH - LOW

Description:

Some smart contracts are using floating pragma versions ^0.6.6 and ^0.6.0 . Locking the **pragma** helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively or recently released pragma versions may have unknown security vulnerabilities.

Listing 10

```
1 - ^0.6.0 (contracts/contracts/child/ChildToken/
    UpgradeableChildERC20/ERC20.sol#3)
2 - ^0.6.6 (contracts/tunnel/BaseRootTunnel.sol#1)
3 - ^0.6.6 (contracts/tunnel/ChildTunnel.sol#1)
4 - ^0.6.6 (contracts/tunnel/RootTunnel.sol#1)
```

Code Location:

Listing 11: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
3 pragma solidity ^0.6.0;
```

Listing 12: contracts/tunnel/BaseRootTunnel.sol

```
1 pragma solidity ^0.6.6;
```

Listing 13: contracts/tunnel/ChildTunnel.sol

```
1 pragma solidity ^0.6.6;
```

Listing 14: contracts/tunnel/RootTunnel.sol

```
1 pragma solidity ^0.6.6;
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

It is recommended to lock the pragma version and not use floating pragma in production. Apart from just locking the pragma version in the code, the sign (^) needs to be removed. It is possible to lock the pragma by fixing the version both in truffle-config.js for Truffle framework or in hardhat.config.js for HardHat framework.

Remediation Plan:

SOLVED: Pragma version was locked to 0.6.6 and all the stated code locations.

3.5 (HAL-05) POSSIBLE MISUSE OF ROLE-BASED ACCESS CONTROL POLICY - LOW

Description:

The `_setupRole(DEFAULT_ADMIN_ROLE, *owner);` does register `_owner` as the admin for all roles created:

> By default, the admin role for all roles is `DEFAULT_ADMIN_ROLE`, which means that only accounts with this role will be able to grant or revoke other roles. More complex role relationships can be created by using `_setRoleAdmin`.

This means that the modifier `only(DEFAULT_ADMIN_ROLE)` does check for a role that is admin of all roles.

Code Location:

Listing 15: contracts/child/ChildChainManager/ChildChainManager.sol (Lines 27)

```
24
25 function initialize(address _owner) external initializer {
26     _setupContractId("ChildChainManager");
27     _setupRole(DEFAULT_ADMIN_ROLE, _owner);
28     _setupRole(MAPPER_ROLE, _owner);
29     _setupRole(STATE_SYNCER_ROLE, _owner);
30 }
```

Listing 16: contracts/child/ChildToken/ChildERC20.sol (Lines 27)

```
24 ) public ERC20(name_, symbol_) {
25     _setupContractId("ChildERC20");
26     _setupDecimals(decimals_);
27     _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
```

```
28      _setupRole(DEPOSITOR_ROLE, childChainManager);
29      _initializeEIP712(name_);
30  }
```

Listing 17: contracts/child/ChildToken/ChildMintableERC721.sol (Lines 32)

```
29  address childChainManager
30  ) public ERC721(name_, symbol_) {
31      _setupContractId("ChildMintableERC721");
32      _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
33      _setupRole(DEPOSITOR_ROLE, childChainManager);
34      _initializeEIP712(name_);
35  }
```

Listing 18: contracts/child/ChildToken/ChildMintableERC20.sol (Lines 27)

```
24  ) public ERC20(name_, symbol_) {
25      _setupContractId("ChildMintableERC20");
26      _setupDecimals(decimals_);
27      _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
28      _setupRole(DEPOSITOR_ROLE, childChainManager);
29      _initializeEIP712(name_);
30  }
```

Listing 19: contracts/child/ChildToken/ChildMintableERC1155.sol (Lines 23)

```
20  ERC1155(uri_)
21  {
22      _setupContractId("ChildMintableERC1155");
23      _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
24      _setupRole(DEPOSITOR_ROLE, childChainManager);
25      _initializeEIP712(uri_);
26  }
```

Listing 20: contracts/child/ChildToken/ChildERC721.sol (Lines 30)

```
27 address childChainManager
28 ) public ERC721(name_, symbol_) {
29     _setupContractId("ChildERC721");
30     _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
31     _setupRole(DEPOSITOR_ROLE, childChainManager);
32     _initializeEIP712(name_);
33 }
```

Listing 21: contracts/child/ChildToken/ChildERC1155.sol (Lines 23)

```
20 ERC1155(uri_)
21 {
22     _setupContractId("ChildERC1155");
23     _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
24     _setupRole(DEPOSITOR_ROLE, childChainManager);
25     _initializeEIP712(uri_);
26 }
```

Listing 22: contracts/child/ChildToken/UpgradeableChildERC20/UChildERC20.sol (Lines 38)

```
35 setSymbol(symbol_);
36 setDecimals(decimals_);
37     _setupContractId(string(abi.encodePacked("Child", symbol_)));
38     _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
39     _setupRole(DEPOSITOR_ROLE, childChainManager);
40     _initializeEIP712(name_);
41 }
```

Listing 23: contracts/tunnel/BaseRootTunnel.sol (Lines 33)

```
30 mapping(bytes32 => bool) public processedExits;
31
32 constructor() internal {
33     _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
34     _setupContractId("RootTunnel");
35 }
```

Listing 24: contracts/tunnel/BaseChildTunnel.sol (Lines 16)

```
13 event MessageSent(bytes message);
14
15 constructor() internal {
16     _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
17     _setupRole(STATE_SYNCER_ROLE, 0
           x0000000000000000000000000000000000001001);
18     _setupContractId("ChildTunnel");
19 }
```

Listing 25: contracts/root/RootToken/DummyMintableERC721.sol (Lines 22)

```
19 ERC721(name_, symbol_)
20 {
21     _setupContractId("DummyMintableERC721");
22     _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
23     _setupRole(PREDICATE_ROLE, _msgSender());
24     _initializeEIP712(name_);
25 }
```

Listing 26: contracts/root/RootToken/DummyMintableERC1155.sol (Lines 20)

```
17
18 constructor(string memory uri_) public ERC1155(uri_) {
19     _setupContractId("DummyMintableERC1155");
20     _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
21     _setupRole(PREDICATE_ROLE, _msgSender());
```

Listing 27: contracts/root/RootToken/DummyERC721.sol (Lines 22)

```
19 ERC721(name_, symbol_)
20 {
21     _setupContractId("DummyERC721");
22     _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
23     _setupRole(PREDICATE_ROLE, _msgSender());
24     _initializeEIP712(name_);
25 }
```

Listing 28: contracts/root/RootToken/DummyMintableERC20.sol (Lines 23)

```
20 ERC20(name_, symbol_)
21 {
22     _setupContractId("DummyMintableERC20");
23     _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
24     _setupRole(PREDICATE_ROLE, _msgSender());
```

Listing 29: contracts/root/TokenPredicates/MintableERC721Predicate.sol (Lines 46)

```
43
44 function initialize(address _owner) external initializer {
45     _setupContractId("MintableERC721Predicate");
46     _setupRole(DEFAULT_ADMIN_ROLE, _owner);
47     _setupRole(MANAGER_ROLE, _owner);
48 }
```

Listing 30: contracts/root/TokenPredicates/ERC1155Predicate.sol (Lines 34)

```
31
32 function initialize(address _owner) external initializer {
33     _setupContractId("ERC1155Predicate");
34     _setupRole(DEFAULT_ADMIN_ROLE, _owner);
35     _setupRole(MANAGER_ROLE, _owner);
36 }
```

Listing 31: contracts/root/TokenPredicates/MintableERC1155Predicate.sol (Lines 43)

```
40
41 function initialize(address _owner) external initializer {
42     _setupContractId("MintableERC1155Predicate");
43     _setupRole(DEFAULT_ADMIN_ROLE, _owner);
44     _setupRole(MANAGER_ROLE, _owner);
45 }
```

Listing 32: contracts/root/TokenPredicates/EtherPredicate.sol (Lines 31)

```
28
29 function initialize(address _owner) external initializer {
30     _setupContractId("EtherPredicate");
31     _setupRole(DEFAULT_ADMIN_ROLE, _owner);
32     _setupRole(MANAGER_ROLE, _owner);
33 }
```

Listing 33: contracts/root/TokenPredicates/MintableERC20Predicate.sol (Lines 34)

```
31
32 function initialize(address _owner) external initializer {
33     _setupContractId("MintableERC20Predicate");
34     _setupRole(DEFAULT_ADMIN_ROLE, _owner);
35     _setupRole(MANAGER_ROLE, _owner);
36 }
```

Listing 34: contracts/root/TokenPredicates/ERC721Predicate.sol (Lines 43)

```
40
41 function initialize(address _owner) external initializer {
42     _setupContractId("ERC721Predicate");
43     _setupRole(DEFAULT_ADMIN_ROLE, _owner);
44     _setupRole(MANAGER_ROLE, _owner);
45 }
```

Listing 35: contracts/root/TokenPredicates/ERC20Predicate.sol (Lines 30)

```
27
28 function initialize(address _owner) external initializer {
29     _setupContractId("ERC20Predicate");
30     _setupRole(DEFAULT_ADMIN_ROLE, _owner);
31     _setupRole(MANAGER_ROLE, _owner);
32 }
```

Listing 36: contracts/root/RootChainManager/RootChainManager.sol (Lines 68)

```
65 {
66     _initializeEIP712("RootChainManager");
67     _setupContractId("RootChainManager");
68     _setupRole(DEFAULT_ADMIN_ROLE, _owner);
69     _setupRole(MAPPER_ROLE, _owner);
70 }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to use a separated role like ADMIN_ROLE (Whose admin will be set to the owner of the DEFAULT_ADMIN_ROLE role by default) in order to segregate privileges.

Remediation Plan:

RISK ACCEPTED: Polygon claims that multi-signature wallet is used for the role-based access control policy composed of several organisations. Changing the code would break the current multi-signature state.

3.6 (HAL-06) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In public functions, array arguments are immediately copied to memory, while external functions can read directly from calldata. Reading calldata is cheaper than memory allocation. Public functions need to write the arguments to memory because public functions may be called internally. Internal calls are passed internally by pointers to memory. Thus, the function expects its arguments being located in memory when the compiler generates the code for an internal function.
Also, methods do not necessarily have to be public if they are only called within the contract-in such case they should be marked internal.

Code Location:

Listing 37: common/Proxy/UpgradableProxy.sol (Lines 56)

```
56 function transferProxyOwnership(address newOwner) public
       onlyProxyOwner {
57         require(newOwner != address(0), "ZERO_ADDRESS");
58         emit ProxyOwnerUpdate(newOwner, loadProxyOwner());
59         setProxyOwner(newOwner);
60     }
```

Listing 38: common/Proxy/UpgradableProxy.sol (Lines 78)

```
78 function updateAndCall(address _newProxyTo, bytes memory data)
       payable public onlyProxyOwner {
79         updateImplementation(_newProxyTo);
80
81         (bool success, bytes memory returnData) = address(this).
               call{value: msg.value}(data);
82         require(success, string(returnData));
83     }
```

Listing 39: contracts/common/NativeMetaTransaction.sol (Lines 37)

```
31 function executeMetaTransaction(
32         address userAddress,
33         bytes memory functionSignature,
34         bytes32 sigR,
35         bytes32 sigS,
36         uint8 sigV
37     ) public payable returns (bytes memory) {
```

Listing 40: contracts/common/NativeMetaTransaction.sol (Lines 83)

```
83 function getNonce(address user) public view returns (uint256 nonce
       ) {
84     nonce = nonces[user];
85 }
```

Listing 41: contracts/common/NativeMetaTransaction.sol (Lines 83)

```
83 function getNonce(address user) public view returns (uint256 nonce
       ) {
84     nonce = nonces[user];
85 }
```

Listing 42: contracts/child/ChildToken/ChildMintableERC20.sol (Lines 76)

```
76 function mint(address user, uint256 amount) public only(
       DEFAULT_ADMIN_ROLE) {
77     _mint(user, amount);
78 }
```

Listing 43: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
75 function name() public view returns (string memory) {
```

Listing 44: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
87 function symbol() public view returns (string memory) {
```

Listing 45: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
108 function decimals() public view returns (uint8) {
```

Listing 46: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
119 function totalSupply() public view override returns (uint256) {
```

Listing 47: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
126 function balanceOf(address account) public view override returns (
        uint256) {
```

Listing 48: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
138 function transfer(address recipient, uint256 amount) public
        virtual override returns (bool) {
```

Listing 49: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
146 function allowance(address owner, address spender) public view
        virtual override returns (uint256) {
```

Listing 50: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
157 function approve(address spender, uint256 amount) public virtual
        override returns (bool) {
```

Listing 51: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
174 function transferFrom(address sender, address recipient, uint256
        amount) public virtual override returns (bool) {
```

Listing 52: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
192 function increaseAllowance(address spender, uint256 addedValue)
        public virtual returns (bool) {
```

Listing 53: contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

```
211 function decreaseAllowance(address spender, uint256
      subtractedValue) public virtual returns (bool) {
```

Listing 54: contracts/root/MockCheckpointManager.sol

```
14 function setCheckpoint(bytes32 rootHash, uint256 start, uint256
      end) public {
15     HeaderBlock memory headerBlock = HeaderBlock({
16         root: rootHash,
17         start: start,
18         end: end,
19         createdAt: now,
20         proposer: msg.sender
21     });
22
23     currentCheckpointNumber = currentCheckpointNumber.add(1);
24     headerBlocks[currentCheckpointNumber] = heakderBlock;
25 }
```

Listing 55: contracts/root/TokenPredicates/ERC20Predicate.sol

```
63 function exitTokens(
64     address,
65     address rootToken,
66     bytes memory log
67 )
68     public
69     override
70     only(MANAGER_ROLE)
71 {
```

Listing 56: contracts/root/TokenPredicates/ERC721Predicate.sol

```
105 function exitTokens(
106     address,
107     address rootToken,
108     bytes memory log
109 )
110     public
111     override
```

```
112     only(MANAGER_ROLE)
113 {
```

Listing 57: contracts/root/TokenPredicates/ERC1155Predicate.sol

```
110 function exitTokens(
111     address,
112     address rootToken,
113     bytes memory log
114 )
115     public
116     override
117     only(MANAGER_ROLE)
118 {
```

Listing 58: contracts/root/TokenPredicates/EtherPredicate.sol

```
 66 function exitTokens(
 67     address,
 68     address rootToken,
 69     bytes memory log
 70 )
 71     public
 72     override
 73     only(MANAGER_ROLE)
 74 {
```

Listing 59: contracts/root/TokenPredicates/MintableERC20Predicate.sol

```
 68 function exitTokens(
 69     address,
 70     address rootToken,
 71     bytes memory log
 72 ) public override only(MANAGER_ROLE) {
```

Listing 60: contracts/root/TokenPredicates/MintableERC721Predicate.sol

```
137 function exitTokens(
138     address,
139     address rootToken,
140     bytes memory log
```

```
141 )
142     public
143     override
144     only(MANAGER_ROLE)
145 {
```

Listing 61: contracts/root/TokenPredicates/MintableERC1155Predicate.sol

```
195 function exitTokens(
196     address,
197     address rootToken,
198     bytes memory log
199 ) public override only(MANAGER_ROLE) {
```

Listing 62: contracts/Tunnel/BaseChildTunnel.sol

```
26 function onStateReceive(uint256, bytes memory message) public only
       (STATE_SYNCER_ROLE) {
27     _processMessageFromRoot(message);
28 }
```

Listing 63: contracts/Tunnel/BaseRootTunnel.sol

```
204 function receiveMessage(bytes memory inputData) public virtual {
205     bytes memory message = _validateAndExtractMessage(inputData);
206     _processMessageFromChild(message);
207 }
```

Remediation Plan:

RISK ACCEPTED: Polygon claims that this changes are not very important and would be performed during deployment.

3.7 (HAL-07) DOS WITH BLOCK GAS LIMIT - INFORMATIONAL

Description:

Iterating over a large array in a loop might lead to a denial-of-service attack. In on of the functions discovered there is a for loop that iterates up to the RLPItem memory length. If this integer is evaluated at extremely large numbers this can cause a DoS.

Code Location:

Listing 64: contracts/lib/RLPRreader.sol (Lines 50)

```
36 function toList(RLPItem memory item)
37     internal
38     pure
39     returns (RLPItem[] memory)
40 {
41     require(isList(item), "RLPReader: ITEM_NOT_LIST");
42
43     uint256 items = numItems(item);
44     RLPItem[] memory result = new RLPItem[](items);
45     uint256 listLength = _itemLength(item.memPtr);
46     require(listLength == item.len, "RLPReader:
           LIST_DECODED_LENGTH_MISMATCH");
47
48     uint256 memPtr = item.memPtr + _payloadOffset(item.memPtr);
49     uint256 dataLen;
50     for (uint256 i = 0; i < items; i++) {
51         dataLen = _itemLength(memPtr);
52         result[i] = RLPItem(dataLen, memPtr);
53         memPtr = memPtr + dataLen;
54     }
55
56     return result;
57 }
```

Listing 65: contracts/lib/RLPReader.sol (Lines 166)

```
159 function numItems(RLPItem memory item) private pure returns (
        uint256) {
160     // add `isList` check if `item` is expected to be passsed
            without a check from calling function
161     // require(isList(item), "RLPReader: NUM_ITEMS_NOT_LIST");
162
163     uint256 count = 0;
164     uint256 currPtr = item.memPtr + _payloadOffset(item.memPtr);
165     uint256 endPtr = item.memPtr + item.len;
166     while (currPtr < endPtr) {
167         currPtr = currPtr + _itemLength(currPtr); // skip over an
                item
168         require(currPtr <= endPtr, "RLPReader:
                NUM_ITEMS_DECODED_LENGTH_MISMATCH");
169         count++;
170     }
171
172     return count;
173 }
```

Listing 66: contracts/lib/RLPReader.sol (Lines 241)

```
233 function copy(
234     uint256 src,
235     uint256 dest,
236     uint256 len
237 ) private pure {
238     if (len == 0) return;
239
240     // copy as many word sizes as possible
241     for (; len >= WORD_SIZE; len -= WORD_SIZE) {
242         assembly {
243             mstore(dest, mload(src))
244         }
245
246         src += WORD_SIZE;
247         dest += WORD_SIZE;
248     }
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

Caution is advised when you expect to have large arrays. Actions that require looping across the entire data structure should be avoided.

If you absolutely must loop over an array of unknown size, then you should plan for it to potentially take multiple blocks, and therefore require multiple transactions.

In this case the decoded data using the RLP must be properly generated during syncing. No one should be able to manually craft a RLP array with the hole purpose of breaking the decoding or performing a DOS on both chains. Make sure that the lengths are properly checked during the decoding phase and that no issue could arise by using big encoded lengths.

Remediation Plan:

SOLVED: Polygon team implemented an iterator system that allows to recover any specific item without iterating all of the data to prevent unwanted loop DOS. Furthermore, internal length checks are done as well.

3.8 (HAL-08) GAS SAVING WHEN COPYING MEMORY - INFORMATIONAL

Description:

In the function for copy memory when len % WORD_SIZE == 0 it is possible to save some gas by adding simple check:

Listing 67: contracts/lib/RLPReader.sol (Lines 254,261)

```
254 if (len > 0) {
255     // left over bytes. Mask is used to remove unwanted bytes from
            the word
256     uint mask = 256 ** (WORD_SIZE - len) - 1;
257     assembly {
258         let srcpart := and(mload(src), not(mask)) // zero out src
259         let destpart := and(mload(dest), mask) // retrieve the
              bytes
260         mstore(dest, or(destpart, srcpart))
261     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

It is recommended to use the latest version of the RLPReader which already implements this gas saving fix in the a2837797e4da79070701339947f32f5725e08b5 commit.

Remediation Plan:

SOLVED: The code was pushed to the latest release that includes the previous recommendation.

3.9 (HAL-09) UNEXPECTED BEHAVIOUR ON CONTEXT MIXIN - INFORMATIONAL

Description:

The msgSender function does not return the caller address when called from the same msg.sender as the contract implementing it. Instead it will return the last calling function that can be found in the internal memory. As seen in Figure 1 a testcase was written in order to call the msg.sender using the same contract address. The results can be seen in Figure 2.

As explained, instead of returning a valid eth address the Keccak-256 for the msgSender function will be returned instead:

Listing 68

```
1 d737d0c7c0024135268d6fb220d4eff805df4f1e65d2e057ffb5dd61d31af866
```

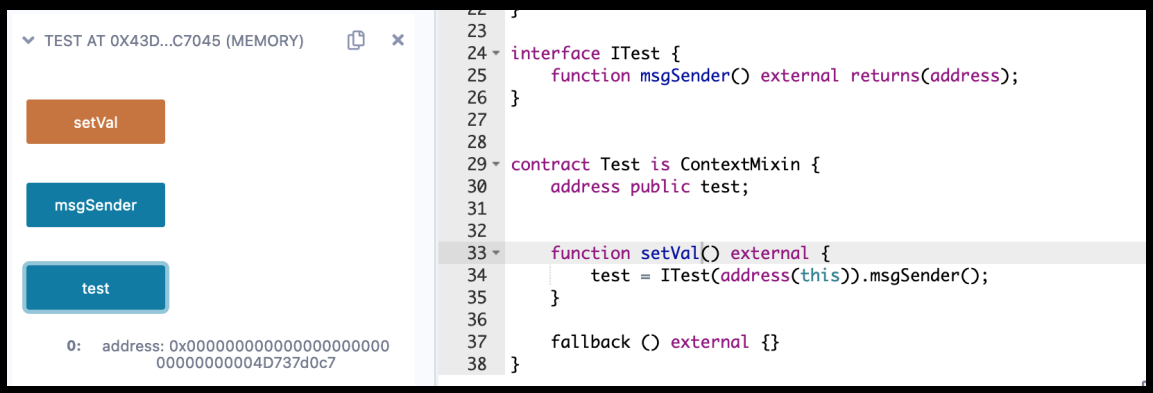

Figure 1: Testing code used to showcase the msgSender funtionally when called internally from the same contract address

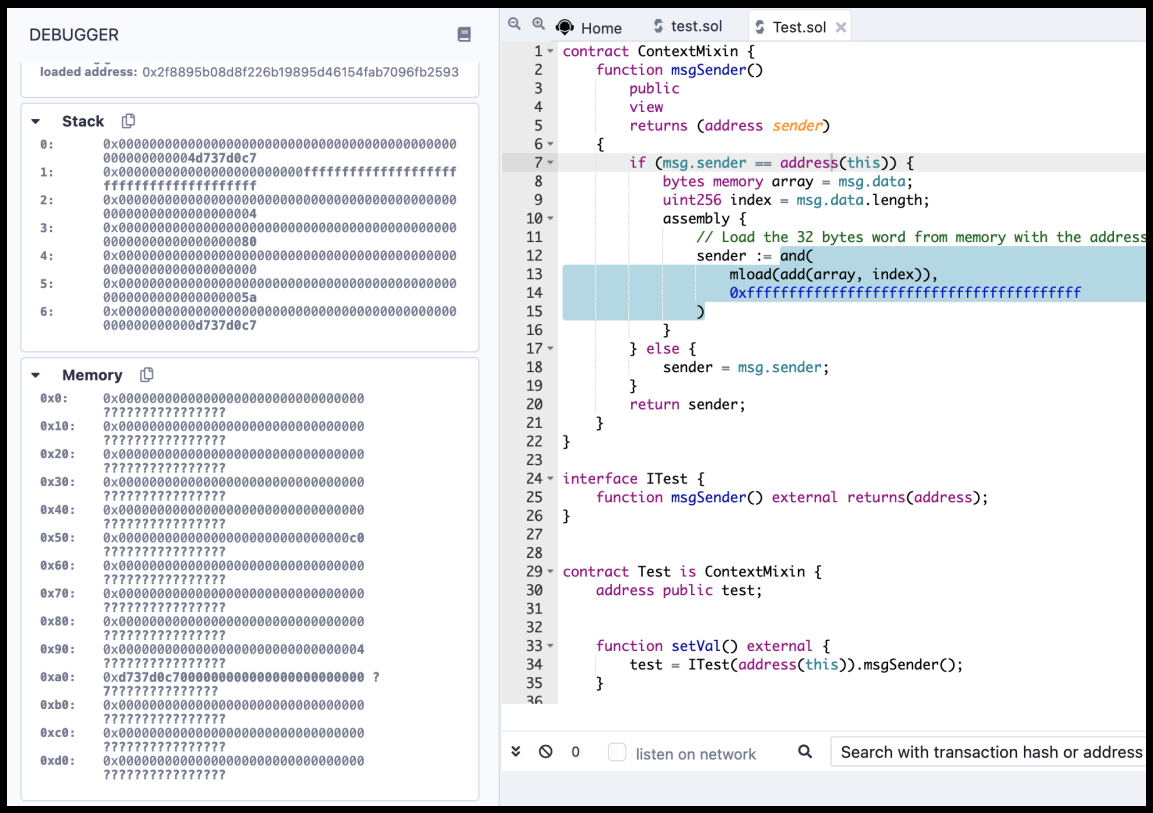

Figure 2: Results of the test case previously mentioned

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

It is recommended to provide information on how the msgSender is expected to return a valid msg.sender when called from the same contract itself. From our understanding, this code could be used when performing meta-transfers. However, the memory manipulation required to achieve a valid return state was not found during the audit. For this reason it is recommended tho provide information on the usage or remove the functionality if not used.

Remediation Plan:

ACKNOWLEDGED: The code will be only for meta-transactions but right now it is not used and ignored.

3.10 (HAL-10) CODE REDUNDANCY - INFORMATIONAL

Description:

The function initializeEIP712 and setupContractId should be removed since the same functionality is already performed on the initialize call of the contract (_initializeEIP712 and _setupContractId respective calls). Calling them again will have no implication on the state of the contract and will lead to an unnecessary gas usage.

Code Location:

Listing 69: root/RootChainManager/RootChainManager.sol (Lines 73,81)

```
72 // adding seperate function setupContractId since initialize is
       already called with old implementation
73 function setupContractId()
74     external
75     only(DEFAULT_ADMIN_ROLE)
76 {
77     _setupContractId("RootChainManager");
78 }
79
80 // adding seperate function initializeEIP712 since initialize is
       already called with old implementation
81 function initializeEIP712()
82     external
83     only(DEFAULT_ADMIN_ROLE)
84 {
85     _setDomainSeperator("RootChainManager");
86 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Consider removing the stated functions or switching those to an internal scope and calling them on the initialize function instead.

Remediation Plan:

RISK ACCEPTED: Polygon Team claims that the code is not harmful since admin permissions are required.

3.11 (HAL-11) MINTING IS ALLOWED TO ANY CALLER - INFORMATIONAL

Description:

All the Dummy tokens which are in-scope as stated on the "EXECUTIVE OVERVIEW - SCOPE" section do allow any caller to mint any amount of tokens.

Code Location:

Listing 70: contracts/root/RootToken/DummyERC1155.sol

```
19 function mint(address account, uint256 id, uint256 amount) public
       {
20     _mint(account, id, amount, bytes(""));
21 }
```

Listing 71: contracts/root/RootToken/DummyMintableERC1155.sol

```
26 function mint(
27     address account,
28     uint256 id,
29     uint256 amount,
30     bytes calldata data
31 ) external override only(PREDICATE_ROLE) {
```

```
32      _mint(account, id, amount, data);
33  }
```

Listing 72: contracts/root/RootToken/DummyERC721.sol

```
27  function mint(uint256 tokenId) public {
28      _mint(_msgSender(), tokenId);
29  }
```

Listing 73: contracts/root/RootToken/DummyERC20.sol

```
21  function mint(uint256 amount) public {
22      _mint(_msgSender(), amount);
23  }
```

Listing 74: contracts/root/RootToken/DummyERC1155.sol (Lines 19)

```
19  function mint(address account, uint256 id, uint256 amount) public
        {
20      _mint(account, id, amount, bytes(""));
21  }
```

Listing 75: contracts/root/RootToken/DummyMintableERC1155.sol (Lines 26)

```
26  function mint(
27      address account,
28      uint256 id,
29      uint256 amount,
30      bytes calldata data
31  ) external override only(PREDICATE_ROLE) {
32      _mint(account, id, amount, data);
33  }
```

Listing 76: contracts/root/RootToken/DummyERC721.sol (Lines 27)

```
27  function mint(uint256 tokenId) public {
28      _mint(_msgSender(), tokenId);
29  }
```

Listing 77: contracts/root/RootToken/DummyMintableERC20.sol (Lines 33)

```
33 function mint(address user, uint256 amount) external override only
       (PREDICATE_ROLE) {
34     _mint(user, amount);
35 }
```

Listing 78: contracts/root/RootToken/DummyERC20.sol (Lines 21)

```
21 function mint(uint256 amount) public {
22     _mint(_msgSender(), amount);
23 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

From the contract names, Halborn deduced that those tokens are used only for testing. In that case, they should be moved to a different folder or state that those tokens would not get deployed during release.

Remediation Plan:

SOLVED: As it is mentioned in the description, Polygon says that the names are examples, and they will never used them in production. Since the directory structure is being maintained for sometime now, they will add a comment on top of each of these Dummy* files, stating they are just examples. The security risk was decreased from CRITICAL to INFORMATIONAL.

MANUAL TESTING

Description:

During the manual testing multiple questions where considered while evaluation each of the defined functions:

- Can it be re-called changing admin/roles and permissions?
- Can somehow an external controlled contract call again the function during the execution of it? (Re-entrancy)
- Can it be called twice in the same block and cause issues?
- Do we control sensitive or vulnerable parameters?
- Does the function check for boundaries on the parameters and internal values? Bigger than zero or equal? Argument count, array sizes, integer truncation . . .
- Are the function parameters and variables controlled by external contracts?
- Can extended contracts cause issues on the extender contract?
- Can I fake the sync state on either side of the bridge?
- Can I get a different state than the one reported by the sender on the Bor chain?

During the evaluation of the source code it was noticed that Proxy were involved on all the deployed contracts. Two possible issues can happen when coding using standard non-proxy contracts:

- Storage collision
- Invalid initialization state

In Solidity, code that is inside a constructor or part of a global variable declaration is not part of a deployed contract's runtime bytecode. This code is executed only once, when the contract instance is deployed. As a consequence of this, the code within a logic contract's constructor will never be executed in the context of the proxy's state. To rephrase, proxies are completely oblivious to the existence of constructors. It's simply as if they weren't there for the proxy.

The problem is easily solved though. Logic contracts should move the code within the constructor to a regular initializer function, and have

this function be called whenever the proxy links to this logic contract. Special care needs to be taken with this initializer function so that it can only be called once, which is one of the properties of constructors in general programming. In this case, contracts make sure that they are only initialized once by extending from Initializable which provides the initializer modifier used in all the initialize declared functions.

The second key points is the storage collision. During contract upgrades it is possible that new variables can be added. Misplacing those variables could lead to storage collision, where that new variable does access a previous stored variable causing logical error a critical issues. This is solved by using unstructured storage, as does some of the contracts of PoS Portal. However, sometimes variables are needed since external libraries or compatible code were not written with unstructured storage in mind. As an example, the following is a dissection of the structured variables declared in RootChainManager:

- From Initializable contract : inited bool variable
- From AccessControl contract : _roles variable (mapping)
- From RootChainManagerStorage:

Listing 79

```
mapping(bytes32 => address) public typeToPredicate;
mapping(address => address) public rootToChildToken;
mapping(address => address) public childToRootToken;
mapping(address => bytes32) public tokenToType;
mapping(bytes32 => bool) public processedExits;
IStateSender internal _stateSender;
ICheckpointManager internal _checkpointManager;
address public childChainManagerAddress;
```

- From AccessControlMixin contract : _revertMsg
- From NativeMetaTransaction contract : nonces.
    - It extends the EIP712Base contract:
        - ERC712_VERSION variable
        - domainSeperator variable

When upgrading the code, new structured variables should always be added at the end of the storage. This means that no new variables should be added in-between the previous stated ones.

Results:

During Predicates testing it was noticed that a re-entracy could happen on the exitTokens function from the EtherPredicate contract if the only (MANAGER_ROLE) was not present. The withdrawer could potentially get control of the execution when performing the transfer. Furthermore, the receive function receive()external payable only(MANAGER_ROLE){} is protecting from sending funds that are not from the Manager. However, it is still possible to bypass this restriction by using selfdestruct.

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Results:

```
INFO:Detectors:
EtherPredicate.exitTokens(address,address,bytes) (contracts/root/TokenPredicates/EtherPredicate.sol#66-93) sends eth to arbitrary user
        Dangerous calls:
        - address(withdrawer).transfer(logRLPList[2].toUint()) (contracts/root/TokenPredicates/EtherPredicate.sol#92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
EtherPredicate.exitTokens(address,address,bytes).withdrawer (contracts/root/TokenPredicates/EtherPredicate.sol#83) lacks a zero-check on :
                - address(withdrawer).transfer(logRLPList[2].toUint()) (contracts/root/TokenPredicates/EtherPredicate.sol#92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
RLPReader.toRlpItem(bytes) (contracts/lib/RLPReader.sol#23-35) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#30-32)
RLPReader.isList(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#64-73) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#67-69)
RLPReader.toRlpBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#78-92) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#86-88)
RLPReader.toUint(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#102-123) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#113-120)
RLPReader.toUintStrict(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#126-139) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#134-136)
RLPReader.toBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#141-156) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#150-152)
RLPReader._itemLength(uint256) (contracts/lib/RLPReader.sol#180-212) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#183-185)
        - INLINE ASM (contracts/lib/RLPReader.sol#191-198)
        - INLINE ASM (contracts/lib/RLPReader.sol#202-208)
RLPReader._payloadOffset(uint256) (contracts/lib/RLPReader.sol#215-230) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#217-219)
RLPReader.copy(uint256,uint256,uint256) (contracts/lib/RLPReader.sol#237-261) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#246-248)
        - INLINE ASM (contracts/lib/RLPReader.sol#256-260)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#119-140) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#132-135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.6.6', '^0.6.0', '^0.6.2']
        - 0.6.6 (contracts/common/AccessControlMixin.sol#1)
```

- The found high vulnerability is a false positive since the withdrawn address is checked and validated via burn proof before calling the `exitTokens` function, that means that the address is controlled

```
INFO:Detectors:
MintableERC20Predicate.lockTokens(address,address,address,bytes) (contracts/root/TokenPredicates/MintableERC20Predicate.sol#45-59) ignores return value by IMintableERC20(rootToken).transferF
rom(depositor,address(this),amount) (contracts/root/TokenPredicates/MintableERC20Predicate.sol#54-58)
MintableERC20Predicate.exitTokens(address,address,bytes) (contracts/root/TokenPredicates/MintableERC20Predicate.sol#68-103) ignores return value by token.transfer(withdrawer,amount) (contrac
ts/root/TokenPredicates/MintableERC20Predicate.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
RLPReader.toRlpItem(bytes) (contracts/lib/RLPReader.sol#23-35) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#30-32)
RLPReader.isList(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#64-73) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#67-69)
RLPReader.toRlpBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#78-92) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#86-88)
RLPReader.toUint(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#102-123) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#113-120)
RLPReader.toUintStrict(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#126-139) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#134-136)
RLPReader.toBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#141-156) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#150-152)
RLPReader._itemLength(uint256) (contracts/lib/RLPReader.sol#180-212) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#183-185)
        - INLINE ASM (contracts/lib/RLPReader.sol#191-198)
        - INLINE ASM (contracts/lib/RLPReader.sol#202-208)
RLPReader._payloadOffset(uint256) (contracts/lib/RLPReader.sol#215-230) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#217-219)
RLPReader.copy(uint256,uint256,uint256) (contracts/lib/RLPReader.sol#237-261) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#246-248)
        - INLINE ASM (contracts/lib/RLPReader.sol#256-260)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#119-140) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#132-135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.6.6', '^0.6.0', '^0.6.2']
        - 0.6.6 (contracts/common/AccessControlMixin.sol#1)
        - 0.6.6 (contracts/common/Initializable.sol#1)
        - 0.6.6 (contracts/lib/RLPReader.sol#6)
        - 0.6.6 (contracts/root/RootToken/IMintableERC20.sol#3)
        - 0.6.6 (contracts/root/TokenPredicates/ITokenPredicate.sol#1)
        - 0.6.6 (contracts/root/TokenPredicates/MintableERC20Predicate.sol#1)
```

- The reported major issue is treated on the “UNCHECKED TRANSFERS AND AMOUNTS” section. It is recommended to use SafeERC20, or ensure that the transfer/transferFrom return value is checked.

```
Compilation warnings/errors on contracts/root/RootChainManager/RootChainManager.sol:
Warning: Contract code size exceeds 24576 bytes (a limit introduced in Spurious Dragon). This contract may not be deployable on mainnet. Consider enabling the optimizer (with a low "runs" va
lue!), turning off revert strings, or using libraries.
  --> contracts/root/RootChainManager/RootChainManager.sol:18:1:
   |
18 | contract RootChainManager is
   | ^ (Relevant source part starts here and spans across multiple lines).


INFO:Detectors:
ContextMixin.msgSender() (contracts/common/ContextMixin.sol#4-23) uses assembly
        - INLINE ASM (contracts/common/ContextMixin.sol#12-18)
EIP712Base.getChainId() (contracts/common/EIP712Base.sol#50-56) uses assembly
        - INLINE ASM (contracts/common/EIP712Base.sol#52-54)
Merkle.checkMembership(bytes32,uint256,bytes32,bytes) (contracts/lib/Merkle.sol#4-36) uses assembly
        - INLINE ASM (contracts/lib/Merkle.sol#19-21)
RLPReader.toRlpItem(bytes) (contracts/lib/RLPReader.sol#23-35) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#30-32)
RLPReader.isList(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#64-73) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#67-69)
RLPReader.toRlpBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#78-92) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#86-88)
RLPReader.toUint(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#102-123) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#113-120)
RLPReader.toUintStrict(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#126-139) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#134-136)
RLPReader.toBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#141-156) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#150-152)
RLPReader._itemLength(uint256) (contracts/lib/RLPReader.sol#180-212) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#183-185)
        - INLINE ASM (contracts/lib/RLPReader.sol#191-198)
        - INLINE ASM (contracts/lib/RLPReader.sol#202-208)
RLPReader._payloadOffset(uint256) (contracts/lib/RLPReader.sol#215-230) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#217-219)
RLPReader.copy(uint256,uint256,uint256) (contracts/lib/RLPReader.sol#237-261) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#246-248)
        - INLINE ASM (contracts/lib/RLPReader.sol#256-260)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#119-140) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#132-135)
```

- The stated warning does report that the contract cannot be deployed since it exceeds the maximum permited length of 24576 bytes. Consider reducing internal used code an variables on the declared functions, refer to “CODE REDUNDANCY” vulnerability.

5.2 AUTOMATED SECURITY SCAN

MYTHX:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruit on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the testers machine and sent the compiled results to the analyzers to locate any vulnerabilities. Only security-related findings are shown below.

Results:

contracts/child/ChildChainManager/ChildChainManagerProxy.sol

```
Report for common/Proxy/Proxy.sol
https://dashboard.mythx.io/#/console/analyses/0b47b3f2-c4e1-4b91-8db3-c99256ebbbe1
```

Line	SWC Title	Severity	Short Description
8	(SWC-123) Requirement Violation	Low	Requirement violation.
8	(SWC-112) Delegatecall to Untrusted Callee	High	The contract delegates execution to another contract with a user-supplied address.

```
Report for common/Proxy/UpgradableProxy.sol
https://dashboard.mythx.io/#/console/analyses/0b47b3f2-c4e1-4b91-8db3-c99256ebbbe1
```

Line	SWC Title	Severity	Short Description
56	(SWC-000) Unknown	Medium	Function could be marked as external.
78	(SWC-000) Unknown	Medium	Function could be marked as external.

contracts/child/ChildToken/ChildMintableERC20.sol

```
Report for contracts/child/ChildToken/ChildMintableERC20.sol
https://dashboard.mythx.io/#/console/analyses/1319a2c8-a568-489b-9bbb-01ad7959516a
```

Line	SWC Title	Severity	Short Description
76	(SWC-000) Unknown	Medium	Function could be marked as external.

```
Report for contracts/common/EIP712Base.sol
https://dashboard.mythx.io/#/console/analyses/1319a2c8-a568-489b-9bbb-01ad7959516a
```

Line	SWC Title	Severity	Short Description
15	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
38	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
39	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.

```
Report for contracts/common/NativeMetaTransaction.sol
https://dashboard.mythx.io/#/console/analyses/1319a2c8-a568-489b-9bbb-01ad7959516a
```

Line	SWC Title	Severity	Short Description

contracts/lib/RLPReader.sol

```
Report for contracts/lib/RLPReader.sol
https://dashboard.mythx.io/#/console/analyses/2eb1fd9f-d08b-488c-a070-6fb25462d064
```

Line	SWC Title	Severity	Short Description
54	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.
170	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.
245	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.

contracts/lib/MerklePatriciaProof.sol

```
Report for MerklePatriciaProof.sol
https://dashboard.mythx.io/#/console/analyses/920cd6fd-314d-41a1-bdbf-71036b2caa4f
```

Line	SWC Title	Severity	Short Description
40	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.
46	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
54	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
80	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
115	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.
120	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.

contracts/root/TokenPredicates/EtherPredicate.sol

```
Report for contracts/root/TokenPredicates/EtherPredicate.sol
https://dashboard.mythx.io/#/console/analyses/4d199722-bc9a-4ca2-b84d-a8ec39011477
```

Line	SWC Title	Severity	Short Description
66	(SWC-000) Unknown	Medium	Function could be marked as external.
92	(SWC-134) Message call with hardcoded gas amount	Low	Call with hardcoded gas amount.

```
Report for node_modules/@openzeppelin/contracts/access/AccessControl.sol
https://dashboard.mythx.io/#/console/analyses/4d199722-bc9a-4ca2-b84d-a8ec39011477
```

Line	SWC Title	Severity	Short Description
95	(SWC-000) Unknown	Medium	Function could be marked as external.
111	(SWC-000) Unknown	Medium	Function could be marked as external.
121	(SWC-000) Unknown	Medium	Function could be marked as external.
135	(SWC-000) Unknown	Medium	Function could be marked as external.
150	(SWC-000) Unknown	Medium	Function could be marked as external.

contracts/root/TokenPredicates/EtherPredicate.sol

```
Report for contracts/root/TokenPredicates/EtherPredicate.sol
https://dashboard.mythx.io/#/console/analyses/4d199722-bc9a-4ca2-b84d-a8ec39011477
```

Line	SWC Title	Severity	Short Description
66	(SWC-000) Unknown	Medium	Function could be marked as external.
92	(SWC-134) Message call with hardcoded gas amount	Low	Call with hardcoded gas amount.

```
Report for node_modules/@openzeppelin/contracts/access/AccessControl.sol
https://dashboard.mythx.io/#/console/analyses/4d199722-bc9a-4ca2-b84d-a8ec39011477
```

Line	SWC Title	Severity	Short Description
95	(SWC-000) Unknown	Medium	Function could be marked as external.
111	(SWC-000) Unknown	Medium	Function could be marked as external.
121	(SWC-000) Unknown	Medium	Function could be marked as external.
135	(SWC-000) Unknown	Medium	Function could be marked as external.
150	(SWC-000) Unknown	Medium	Function could be marked as external.

contracts/common/NativeMetaTransaction.sol

```
Report for contracts/common/EIP712Base.sol
https://dashboard.mythx.io/#/console/analyses/4b116b4a-4204-4584-8164-27ce5d8c29d8
```

Line	SWC Title	Severity	Short Description
15	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
38	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
39	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.

```
Report for contracts/common/NativeMetaTransaction.sol
https://dashboard.mythx.io/#/console/analyses/4b116b4a-4204-4584-8164-27ce5d8c29d8
```

Line	SWC Title	Severity	Short Description
8	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
18	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
31	(SWC-000) Unknown	Medium	Function could be marked as external.
78	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
83	(SWC-000) Unknown	Medium	Function could be marked as external.
87	(SWC-127) Arbitrary Jump with Function Type Variable	High	The caller can redirect execution to arbitrary bytecode locations.

contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol

Report for contracts/child/ChildToken/UpgradeableChildERC20/ERC20.sol
https://dashboard.mythx.io/#/console/analyses/a9952756-497c-4a88-8ddc-5e66fa93f777

Line	SWC Title	Severity	Short Description
75	(SWC-000) Unknown	Medium	Function could be marked as external.
87	(SWC-000) Unknown	Medium	Function could be marked as external.
108	(SWC-000) Unknown	Medium	Function could be marked as external.
119	(SWC-000) Unknown	Medium	Function could be marked as external.
126	(SWC-000) Unknown	Medium	Function could be marked as external.
138	(SWC-000) Unknown	Medium	Function could be marked as external.
146	(SWC-000) Unknown	Medium	Function could be marked as external.
157	(SWC-000) Unknown	Medium	Function could be marked as external.
174	(SWC-000) Unknown	Medium	Function could be marked as external.
192	(SWC-000) Unknown	Medium	Function could be marked as external.
211	(SWC-000) Unknown	Medium	Function could be marked as external.
327	(SWC-131) Presence of unused variables	Low	Unused function parameter "to".
327	(SWC-131) Presence of unused variables	Low	Unused function parameter "amount".
327	(SWC-131) Presence of unused variables	Low	Unused function parameter "from".

The issues stated as “Function could be marked as external” are explained in “POSSIBLE MISUSE OF PUBLIC FUNCTIONS”. The reported issues on the RLPReader are explained in “DOS WITH BLOCK GAS LIMIT”.

THANK YOU FOR CHOOSING

// HALBORN