Avalanche Bridge

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **June 21st, 2021 - June 28th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	06/21/2021	Gabi Urrutia
0.2	Document Edits	06/23/2021	Ataberk Yavuzer
0.5	Document Edits	06/25/2021	Gokberk Gulgun
1.0	Final Version	06/28/2021	Gabi Urrutia
1.1	Remediation Plan	07/21/2021	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
Ataberk Yavuzer	Halborn	Ataberk.Yavuzer@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Ava Labs engaged Halborn to conduct a security assessment on their Smart contract beginning on June 21st, 2021 and ending June 28th, 2021. The security assessment was scoped to the smart contract repository. An audit of the security risk and implications regarding the changes introduced by the development team at Ava Labs prior to its production release shortly following the assessments deadline.

The result of the audit is that some essential issues must be fixed.

1.2 AUDIT SUMMARY

The team at Halborn was provided one week 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.

In summary, Halborn identified few security risks, and recommends performing further testing to validate extended safety and correctness in context to the whole of contract. External threats, such as economic attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

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 read 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.
- Scanning of solidity files for vulnerabilities, security hotspots, or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (REMIX)

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 following smart contracts:

https://github.com/ava-labs/evm-sgx-bridge/tree/main/SmartContracts

Commit ID: 9eb4360782dc6cd3c6096f16aada3c34b68a37ca

OUT-OF-SCOPE:
Other smart contracts in the repository, external libraries and economics attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	5	5

LIKELIHOOD

IMPACT

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

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - USE OF TX.ORIGIN	Low	ACCEPTED RISK: 07/16/2021
HAL02 - FLOATING PRAGMA	Low	SOLVED: 07/16/2021
HAL03 - MISSING FEE LIMIT DEFINITION	Low	NOT APPLICABLE: 07/16/2021
HAL04 - MISSING ACCESS CONTROL CHECK	Low	NOT APPLICABLE: 07/16/2021
HAL05 - MISSING RE-ENTRANCY PROTECTION	Low	NOT APPLICABLE: 07/16/2021
HAL06 - USE OF INLINE ASSEMBLY	Informational	RISK ACCEPTED: 07/16/2021
HAL07 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	ACKNOWLEDGED: 07/16/2021
HAL08 - IGNORED RETURN VALUES	Informational	ACKNOWLEDGED: 07/16/2021
HAL09 - PRAGMA VERSION	Informational	RISK ACCEPTED: 07/16/2021
HAL10 - MISUSE OF BRIDGE ROLE ARRAY	Informational	NOT APPLICABLE: 07/16/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) USE OF TX.ORIGIN - LOW

Description:

WrappedERC20.sol, WrappedLINK.sol contracts use tx.origin so that unwrap () function can be called by anybody. It is recommended that you use msg.sender instead of tx.origin because if a transaction is made to a malicious wallet, when you check it you will have the origin address and you will not be able to know the address of the malicious wallet. Nevertheless, the use of tx.origin is semi-legitimized for recording who calls the contract most. Furthermore, tx.origin could be used to prevent an address from interacting with your contract because the owner of the address cannot use the contract as an intermediary to circumvent your blocking. Finally, it is important to remark that the use of tx.origin will be deprecated.

Code Location:

WrappedERC20.sol Line #54

Listing 1: WrappedERC20.sol (Lines 55)

```
54    function unwrap(uint256 amount, uint256 chain_id) public {
55        require(tx.origin == msg.sender, "
              CONTRACT_CALLS_NOT_SUPPORTED");
56        require(chain_ids[chain_id] == true, "
              CHAIN_ID_NOT_SUPPORTED");
57        _burn(msg.sender, amount);
58    }
```

WrappedLINK.sol Line #56

Listing 2: WrappedERC20.sol (Lines 57)

```
56    function unwrap(uint256 amount, uint256 chain_id) public {
57        require(tx.origin == msg.sender, "
              CONTRACT_CALLS_NOT_SUPPORTED");
```

```
        require(chain_ids[chain_id] == true, "
            CHAIN_ID_NOT_SUPPORTED");
        _burn(msg.sender, amount);
    }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended not to use tx.origin because a malicious wallet could receive funds and cannot be tracked. However, its use is semi-legitimate in some cases with caution.

Remediation Plan:

ACCEPTED RISK: AVA Labs Team accepts the risk considering that it will not pose a high severity security issue.

3.2 (HAL-02) FLOATING PRAGMA - LOW

Description:

In the WrappedERC20.sol, WrappedLINK.sol, Roles.sol, ERC677.sol and ERC677Receiver.sol the contracts use the floating pragma ^0.8.0. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the **pragma** helps to ensure that contracts do not accidentally get deployed using another pragma, for example, either an outdated pragma version that might introduce bugs that affect the contract system negatively or a recently released pragma version which has not been extensively tested.

Reference: ConsenSys Diligence - Lock pragmas

Code Location:

WrappedERC20.sol Line #1
WrappedLINK.sol Line #1
Roles.sol Line #1
ERC677.sol Line #1
ERC677Receiver.sol Line #1

Listing 3: WrappedERC20.sol (Lines 1)

```
pragma solidity ^0.8.0;
```

- This is an example where the floating pragma is used. ^0.8.0.

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Consider lock the pragma version known bugs for the compiler version. Therefore, it is recommended not to use floating pragma in the production. Apart from just locking the pragma version in the code, the sign (>=) need to be removed. it is possible locked the pragma fixing the version both in truffle-config.js if you use the Truffle framework and in hardhat.config.js if you use HardHat framework for the deployment.

Remediation Plan:

SOLVED: AVA Labs Team locked the pragma version of Solidity (0.8.0).

3.3 (HAL-03) MISSING FEE LIMIT DEFINITION - LOW

Description:

During the tests, Halborn Team noticed that on the mint function, the fee limits are not defined. To achieve a better implementation in both economic and security aspects, the amount in the mint function should be limited.

Code Location:

WrappedERC20.sol Line #35

Listing 4: WrappedERC20.sol (Lines 35)

```
35     function mint(
36         address to,
37         uint256 amount,
38         address fee_address,
39         uint256 fee_amount,
40         bytes32
41     ) public {
42         require(_bridge_roles.has(msg.sender), "
               DOES_NOT_HAVE_BRIDGE_ROLE");
43         _mint(to, amount);
44         if (fee_amount > 0) {
45             _mint(fee_address, fee_amount);
46         }
47     }
```

WrappedLINK.sol Line #35

Listing 5: WrappedLINK.sol (Lines 37)

```
37     function mint(
38         address to,
39         uint256 amount,
40         address fee_address,
41         uint256 fee_amount,
42         bytes32
43     ) public {
44         require(_bridge_roles.has(msg.sender), "
               DOES_NOT_HAVE_BRIDGE_ROLE");
45         _mint(to, amount);
46         if (fee_amount > 0) {
47             _mint(fee_address, fee_amount);
48         }
49     }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended define maximum and minimum fee range on the related functions.

Remediation Plan:

NOT APPLICABLE: AVA Labs Team considers that it is unlikely to know the maximum fee amount and just the bridge_role is able to call this function.

3.4 (HAL-04) MISSING ACCESS CONTROL CHECK - LOW

Description:

During the tests, It has been observed that, bridge role check is missing on the unwrap function. Although, chain id is checked in the function, chain_id's first element is always set to true. Therefore, It is possible to bypass modifier on the function.

Code Location:

WrappedERC20.sol Line #54

Listing 6: WrappedERC20.sol (Lines 54)

```
54     function unwrap(uint256 amount, uint256 chain_id) public {
55         require(chain_ids[chain_id] == true, "
               CHAIN_ID_NOT_SUPPORTED");
56         _burn(msg.sender, amount);
57     }
```

WrappedLINK.sol Line #56

Listing 7: WrappedERC20.sol (Lines 56)

```
56     function unwrap(uint256 amount, uint256 chain_id) public {
57         require(chain_ids[chain_id] == true, "
               CHAIN_ID_NOT_SUPPORTED");
58         _burn(msg.sender, amount);
59     }
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to implementing access control on the function.

Remediation Plan:

NOT APPLICABLE: AVA Labs Team claims that the unwrap function is supposed to be public and callable.

3.5 (HAL-05) MISSING RE-ENTRANCY PROTECTION - LOW

Description:

To protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the withdraw function with a recursive call. OpenZeppelin has it's own mutex implementation called ReentrancyGuard which provides a modifier to any function called nonReentrant that guards the function with a mutex against Reentrancy attacks.

Code Location:

WrappedERC20.sol Line #115

Listing 8: WrappedERC20.sol (Lines 115)

```
115 function swap(address token, uint256 amount) public {
116         require(isContract(token), "TOKEN_IS_NOT_CONTRACT");
117         require(
118             _swap_tokens[token].token_contract != address(0),
119             "SWAP_TOKEN_IS_NOT_SUPPORTED"
120         );
121         require(
122             amount <= _swap_tokens[token].supply,
123             "SWAP_AMOUNT_MORE_THAN_ALLOWED_SUPPLY"
124         );
125
126         . . .
127
128         _swap_tokens[token].supply = _swap_tokens[token].supply.
            sub(amount);
129
130        . . .
131
132         _mint(msg.sender, amount);
133     }
```

WrappedLINK.sol Line #113

Listing 9: WrappedLINK.sol (Lines 113)

```
113 function swap(address token, uint256 amount) public {
114         require(isContract(token), "TOKEN_IS_NOT_CONTRACT");
115         require(
116             _swap_tokens[token].token_contract != address(0),
117             "SWAP_TOKEN_IS_NOT_SUPPORTED"
118         );
119         require(
120             amount <= _swap_tokens[token].supply,
121             "SWAP_AMOUNT_MORE_THAN_ALLOWED_SUPPLY"
122         );
123
124         . . .
125
126         _swap_tokens[token].supply = _swap_tokens[token].supply.
            sub(amount);
127
128         . . .
129
130         _mint(msg.sender, amount);
131     }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

In WrappedERC20.sol and WrappedLINK.sol contracts, the swap() functions are missing nonReentrant guard. Use the nonReentrant modifier to avoid introducing future vulnerabilities.

Remediation Plan:

NOT APPLICABLE: AVA Labs Team claims that the amount to be swapped comes from the supply. In addition, the users balance is checked before executing mint and swaps functions. Furthermore, no external calls are executed after checking the balance in the function. Finally, note that just the contracts called by this method are set, deployed and checked directly by the bridge.

3.6 (HAL-06) USE OF INLINE ASSEMBLY - INFORMATIONAL

Description:

Inline assembly is a way to access the Virtual Machine at a low level. This discards several important safety features in Solidity.

Code Location:

WrappedERC20.sol Line #136

Listing 10: WrappedERC20.sol (Lines 136)

```
136     function isContract(address _addr) private view returns (bool
            hasCode) {
137         uint256 length;
138         assembly {
139             length := extcodesize(_addr)
140         }
141         return length > 0;
142     }
```

WrappedLINK.sol Line #165

Listing 11: WrappedLINK.sol (Lines 165)

```
165     function isContract(address _addr) private view returns (bool
            hasCode) {
166         uint256 length;
167         assembly {
168             length := extcodesize(_addr)
169         }
170         return length > 0;
171     }
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

The contracts should avoid using inline assembly because it interacts with the EVM (Ethereum Virtual Machine) at a low level. An attacker could bypass many essential safety features of Solidity.

Remediation Plan:

RISK ACCEPTED: AVA Labs Team assumes inline assembly is mandatory to implement isContract() function.

3.7 (HAL-07) 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.

Code Location:

WrappedERC20.sol
WrappedLINK.sol

Listing 12: Functions (Lines)

```
decimals()
mint(address,uint256,address,uint256,bytes32)
add_supported_chain_id(uint256)
unwrap(uint256,uint256)
migrate_bridge_role(address)
add_swap_token(address,uint256)
remove_swap_token(address,uint256)
swap(address,uint256)
transferAndCall(address,uint256,bytes)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider declaring external variables instead of public variables. A best practice is to use external if expecting a function to only be called externally and public if called internally. Public functions are always accessible, but external functions are only available to external callers.

Remediation:

ACKNOWLEDGED: AVA Labs Team claims that the use of public functions is intended.

3.8 (HAL-08) IGNORED RETURN VALUES - INFORMATIONAL

Description:

The return value of an external call is not stored in a local or state variable. In the Ava Labs contracts, there are a few instances where the multiple methods are called and the return value (bool) is ignored.

WrappedERC20.sol Line #115

Listing 13: WrappedERC20.sol (Lines 133)

```
115     function swap(address token, uint256 amount) public {
116         require(isContract(token), "TOKEN_IS_NOT_CONTRACT");
117         require(
118             _swap_tokens[token].token_contract != address(0),
119             "SWAP_TOKEN_IS_NOT_SUPPORTED"
120         );
121         require(
122             amount <= _swap_tokens[token].supply,
123             "SWAP_AMOUNT_MORE_THAN_ALLOWED_SUPPLY"
124         );
125 
126         // Update the allowed swap amount.
127         _swap_tokens[token].supply = _swap_tokens[token].supply.
                sub(amount);
128 
129         // Burn the old token.
130         ERC20Burnable swap_token = ERC20Burnable(
131             _swap_tokens[token].token_contract
132         );
133         swap_token.burnFrom(msg.sender, amount);
134 
135         // Mint the new token.
136         _mint(msg.sender, amount);
137     }
```

WrappedLINK.sol Line #113

Listing 14: WrappedERC20.sol (Lines 130)

```
113     function swap(address token, uint256 amount) public {
114         require(isContract(token), "TOKEN_IS_NOT_CONTRACT");
115         require(
116             _swap_tokens[token].token_contract != address(0),
117             "SWAP_TOKEN_IS_NOT_SUPPORTED"
118         );
119         require(
120             amount <= _swap_tokens[token].supply,
121             "SWAP_AMOUNT_MORE_THAN_ALLOWED_SUPPLY"
122         );
123
124         // Update the allowed swap amount.
125         _swap_tokens[token].supply = _swap_tokens[token].supply.
            sub(amount);
126
127         // Burn the old token.
128         ERC20Burnable swap_token =
129             ERC20Burnable(_swap_tokens[token].token_contract);
130         swap_token.burnFrom(msg.sender, amount);
131
132         // Mint the new token.
133         _mint(msg.sender, amount);
134     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Add a return value check to avoid an unexpected crash of the contract. Return value checks provide better exception handling.

Remediation Plan:

ACKNOWLEDGED: AVA Labs Team claims that the non-use of return values is intended.

3.9 (HAL-09) PRAGMA VERSION - INFORMATIONAL

Description:

In-scope contracts use one of the latest pragma version (0.8.0) which was released back in December 16, 2020. Many pragma versions have been released, going from version 0.6.x to the recently released version 0.8.x in close time gap. However, new pragma versions may still contain multiple undiscovered bugs.

Code Location:

WrappedERC20.sol Line #1

Listing 15: WrappedERC20.sol (Lines 2)

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 import "@openzeppelin/contracts/token/ERC20/extensions/
      ERC20Burnable.sol";
5 import "@openzeppelin/contracts/utils/math/SafeMath.sol";
6 import "./Roles.sol";
7
8 contract WrappedERC20 is ERC20Burnable {
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

In the Solitidy Github repository, there is a JSON file listing the bugs reported for each compiler version. There are multiple bugs have been found in version 0.8.0. According to the same document, there are several bugs discovered in version 0.6.12. However, pragma version 0.6.12 is widely used by Solidity developers and has been extensively tested in many security audits. Hence, version 0.6.12 is more reliable than version 0.8.0. We recommend using the latest stable version which is 0.6.12.

Reference: https://github.com/ethereum/solidity/blob/develop/docs/bugs_by_version.json

Remediation Plan:

RISK ACCEPTED: AVA Labs Team considers appropriate the use of pragma version 0.8.0 although it is too recent and not a very tested version.

3.10 (HAL-10) MISUSE OF BRIDGE ROLE ARRAY - INFORMATIONAL

Description:

In the WrappedERC20.sol and WrappedLINK.sol smart contracts, the migrate_bridge_role function is used to renounce bridge role. However, when the new bridge role address is chosen, the previous bridge role is removed from the array.

Code Location:

WrappedERC20.sol Line #59

Listing 16: WrappedERC20.sol (Lines 59)

```
59     function migrate_bridge_role(address new_bridge_role_address)
           public {
60         require(_bridge_roles.has(msg.sender), "
               DOES_NOT_HAVE_BRIDGE_ROLE");
61         _bridge_roles.remove(msg.sender);
62         _bridge_roles.add(new_bridge_role_address);
63     }
```

WrappedLINK.sol Line #61

Listing 17: WrappedLINK.sol (Lines 61)

```
61     function migrate_bridge_role(address new_bridge_role_address)
           public {
62         require(_bridge_roles.has(msg.sender), "
               DOES_NOT_HAVE_BRIDGE_ROLE");
63         _bridge_roles.remove(msg.sender);
64         _bridge_roles.add(new_bridge_role_address);
65     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended that to revise the migrate_bridge_role function to properly add multiple address on the bridge roles array.

Remediation Plan:

NOT APPLICABLE: AVA Labs Team claims that this behaviour is intentional and there should only be one bridge role.

3.11 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, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire codebase.

Results:

```
INFO:Detectors:
Reentrancy in WrappedERC20.swap(address,uint256) (contracts/WrappedERC20.sol#111-132):
        External calls:
        - swap_token.burnFrom(msg.sender,amount) (contracts/WrappedERC20.sol#128)
        State variables written after the call(s):
        - _mint(msg.sender,amount) (contracts/WrappedERC20.sol#131)
                - _balances[account] += amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#240)
        - _mint(msg.sender,amount) (contracts/WrappedERC20.sol#131)
                - _totalSupply += amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#239)
Reentrancy in WrappedLINK.swap(address,uint256) (contracts/WrappedLINK.sol#113-134):
        External calls:
        - swap_token.burnFrom(msg.sender,amount) (contracts/WrappedLINK.sol#130)
        State variables written after the call(s):
        - _mint(msg.sender,amount) (contracts/WrappedLINK.sol#133)
                - _balances[account] += amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#240)
        - _mint(msg.sender,amount) (contracts/WrappedLINK.sol#133)
                - _totalSupply += amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#239)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in WrappedERC20.swap(address,uint256) (contracts/WrappedERC20.sol#111-132):
        External calls:
        - swap_token.burnFrom(msg.sender,amount) (contracts/WrappedERC20.sol#128)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#241)
                - _mint(msg.sender,amount) (contracts/WrappedERC20.sol#131)
Reentrancy in WrappedLINK.swap(address,uint256) (contracts/WrappedLINK.sol#113-134):
        External calls:
        - swap_token.burnFrom(msg.sender,amount) (contracts/WrappedLINK.sol#130)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#241)
                - _mint(msg.sender,amount) (contracts/WrappedLINK.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
WrappedERC20.isContract(address) (contracts/WrappedERC20.sol#134-140) uses assembly
        - INLINE ASM (contracts/WrappedERC20.sol#136-138)
WrappedLINK.isContract(address) (contracts/WrappedLINK.sol#165-171) uses assembly
        - INLINE ASM (contracts/WrappedLINK.sol#167-169)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
WrappedERC20.unwrap(uint256,uint256) (contracts/WrappedERC20.sol#54-57) compares to a boolean constant:
        -require(bool,string)(chain_ids[chain_id] == true,CHAIN_ID_NOT_SUPPORTED) (contracts/WrappedERC20.sol#55)
WrappedLINK.unwrap(uint256,uint256) (contracts/WrappedLINK.sol#56-59) compares to a boolean constant:
        -require(bool,string)(chain_ids[chain_id] == true,CHAIN_ID_NOT_SUPPORTED) (contracts/WrappedLINK.sol#57)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
```

```
INFO:Detectors:
Parameter WrappedERC20.mint(address,uint256,address,uint256,bytes32).fee_address (contracts/WrappedERC20.sol#38) is not in mixedCase
Parameter WrappedERC20.mint(address,uint256,address,uint256,bytes32).fee_amount (contracts/WrappedERC20.sol#39) is not in mixedCase
Function WrappedERC20.add_supported_chain_id(uint256) (contracts/WrappedERC20.sol#49-52) is not in mixedCase
Parameter WrappedERC20.add_supported_chain_id(uint256).chain_id (contracts/WrappedERC20.sol#49) is not in mixedCase
Parameter WrappedERC20.unwrap(uint256,uint256).chain_id (contracts/WrappedERC20.sol#54) is not in mixedCase
Function WrappedERC20.migrate_bridge_role(address) (contracts/WrappedERC20.sol#59-63) is not in mixedCase
Parameter WrappedERC20.migrate_bridge_role(address).new_bridge_role_address (contracts/WrappedERC20.sol#59) is not in mixedCase
Function WrappedERC20.add_swap_token(address,uint256) (contracts/WrappedERC20.sol#65-85) is not in mixedCase
Parameter WrappedERC20.add_swap_token(address,uint256).contract_address (contracts/WrappedERC20.sol#65) is not in mixedCase
Parameter WrappedERC20.add_swap_token(address,uint256).supply_increment (contracts/WrappedERC20.sol#65) is not in mixedCase
Function WrappedERC20.remove_swap_token(address,uint256) (contracts/WrappedERC20.sol#87-109) is not in mixedCase
Parameter WrappedERC20.remove_swap_token(address,uint256).contract_address (contracts/WrappedERC20.sol#88) is not in mixedCase
Parameter WrappedERC20.remove_swap_token(address,uint256).supply_decrement (contracts/WrappedERC20.sol#89) is not in mixedCase
Parameter WrappedERC20.isContract(address)._addr (contracts/WrappedERC20.sol#134) is not in mixedCase
Variable WrappedERC20._bridge_roles (contracts/WrappedERC20.sol#12) is not in mixedCase
Variable WrappedERC20._swap_tokens (contracts/WrappedERC20.sol#22) is not in mixedCase
Variable WrappedERC20.chain_ids (contracts/WrappedERC20.sol#24) is not in mixedCase
Parameter WrappedLINK.mint(address,uint256,address,uint256,bytes32).fee_address (contracts/WrappedLINK.sol#40) is not in mixedCase
Parameter WrappedLINK.mint(address,uint256,address,uint256,bytes32).fee_amount (contracts/WrappedLINK.sol#41) is not in mixedCase
Function WrappedLINK.add_supported_chain_id(uint256) (contracts/WrappedLINK.sol#51-54) is not in mixedCase
Parameter WrappedLINK.add_supported_chain_id(uint256).chain_id (contracts/WrappedLINK.sol#51) is not in mixedCase
Parameter WrappedLINK.unwrap(uint256,uint256).chain_id (contracts/WrappedLINK.sol#56) is not in mixedCase
Function WrappedLINK.migrate_bridge_role(address) (contracts/WrappedLINK.sol#61-65) is not in mixedCase
Parameter WrappedLINK.migrate_bridge_role(address).new_bridge_role_address (contracts/WrappedLINK.sol#61) is not in mixedCase
Function WrappedLINK.add_swap_token(address,uint256) (contracts/WrappedLINK.sol#67-87) is not in mixedCase
Parameter WrappedLINK.add_swap_token(address,uint256).contract_address (contracts/WrappedLINK.sol#67) is not in mixedCase
Parameter WrappedLINK.add_swap_token(address,uint256).supply_increment (contracts/WrappedLINK.sol#67) is not in mixedCase
Function WrappedLINK.remove_swap_token(address,uint256) (contracts/WrappedLINK.sol#89-111) is not in mixedCase
Parameter WrappedLINK.remove_swap_token(address,uint256).contract_address (contracts/WrappedLINK.sol#90) is not in mixedCase
Parameter WrappedLINK.remove_swap_token(address,uint256).supply_decrement (contracts/WrappedLINK.sol#91) is not in mixedCase
Parameter WrappedLINK.transferAndCall(address,uint256,bytes)._to (contracts/WrappedLINK.sol#143) is not in mixedCase
Parameter WrappedLINK.transferAndCall(address,uint256,bytes)._value (contracts/WrappedLINK.sol#144) is not in mixedCase
Parameter WrappedLINK.transferAndCall(address,uint256,bytes)._data (contracts/WrappedLINK.sol#145) is not in mixedCase
Parameter WrappedLINK.contractFallback(address,uint256,bytes)._to (contracts/WrappedLINK.sol#157) is not in mixedCase
Parameter WrappedLINK.contractFallback(address,uint256,bytes)._value (contracts/WrappedLINK.sol#158) is not in mixedCase
Parameter WrappedLINK.contractFallback(address,uint256,bytes)._data (contracts/WrappedLINK.sol#159) is not in mixedCase
Parameter WrappedLINK.isContract(address)._addr (contracts/WrappedLINK.sol#165) is not in mixedCase
Variable WrappedLINK._bridge_roles (contracts/WrappedLINK.sol#14) is not in mixedCase
Variable WrappedLINK._swap_tokens (contracts/WrappedLINK.sol#24) is not in mixedCase
Variable WrappedLINK.chain_ids (contracts/WrappedLINK.sol#26) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (node_modules/@openzeppelin/contracts/utils/Context.sol#21)" inContext (node_modules/@openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#60-62)
symbol() should be declared external:
        - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#68-70)
decimals() should be declared external:
        - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#85-87)
        - WrappedERC20.decimals() (contracts/WrappedERC20.sol#31-33)
        - WrappedLINK.decimals() (contracts/WrappedLINK.sol#33-35)
totalSupply() should be declared external:
        - ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#92-94)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#99-101)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#130-133)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#148-156)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#170-173)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#189-195)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#19-21)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#34-39)
mint(address,uint256,address,uint256,bytes32) should be declared external:
        - WrappedERC20.mint(address,uint256,address,uint256,bytes32) (contracts/WrappedERC20.sol#35-47)
add_supported_chain_id(uint256) should be declared external:
        - WrappedERC20.add_supported_chain_id(uint256) (contracts/WrappedERC20.sol#49-52)
unwrap(uint256,uint256) should be declared external:
        - WrappedERC20.unwrap(uint256,uint256) (contracts/WrappedERC20.sol#54-57)
migrate_bridge_role(address) should be declared external:
        - WrappedERC20.migrate_bridge_role(address) (contracts/WrappedERC20.sol#59-63)
add_swap_token(address,uint256) should be declared external:
        - WrappedERC20.add_swap_token(address,uint256) (contracts/WrappedERC20.sol#65-85)
remove_swap_token(address,uint256) should be declared external:
        - WrappedERC20.remove_swap_token(address,uint256) (contracts/WrappedERC20.sol#87-109)
swap(address,uint256) should be declared external:
        - WrappedERC20.swap(address,uint256) (contracts/WrappedERC20.sol#111-132)
mint(address,uint256,address,uint256,bytes32) should be declared external:
        - WrappedLINK.mint(address,uint256,address,uint256,bytes32) (contracts/WrappedLINK.sol#37-49)
add_supported_chain_id(uint256) should be declared external:
        - WrappedLINK.add_supported_chain_id(uint256) (contracts/WrappedLINK.sol#51-54)
unwrap(uint256,uint256) should be declared external:
        - WrappedLINK.unwrap(uint256,uint256) (contracts/WrappedLINK.sol#56-59)
migrate_bridge_role(address) should be declared external:
        - WrappedLINK.migrate_bridge_role(address) (contracts/WrappedLINK.sol#61-65)
add_swap_token(address,uint256) should be declared external:
        - WrappedLINK.add_swap_token(address,uint256) (contracts/WrappedLINK.sol#67-87)
remove_swap_token(address,uint256) should be declared external:
        - WrappedLINK.remove_swap_token(address,uint256) (contracts/WrappedLINK.sol#89-111)
swap(address,uint256) should be declared external:
        - WrappedLINK.swap(address,uint256) (contracts/WrappedLINK.sol#113-134)
transferAndCall(address,uint256,bytes) should be declared external:
        - WrappedLINK.transferAndCall(address,uint256,bytes) (contracts/WrappedLINK.sol#142-153)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

According to the test results, most of the findings found by slither were considered as false positives. Relevant findings were reviewed by the auditors.

3.12 AUTOMATED SECURITY SCAN RESULTS

Description:

Halborn used automated security scanners to assist with detection of well known security issues, and identify low-hanging fruit on the scoped contract targeted 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 MythX to locate any vulnerabilities. Security Detections are only in scope, and the analysis was pointed towards issues with the in-scope smart contracts.

Results:

Report for contracts/Roles.sol
https://dashboard.mythx.io/#/console/analyses/8c0312af-aa06-4fab-b975-4e31ec8503e9

Line	SWC Title	Severity	Short Description
34	(SWC-115) Authorization through tx.origin	Low	Use of tx.origin as a part of authorization control.

Report for contracts/WrappedERC20.sol
https://dashboard.mythx.io/#/console/analyses/8c0312af-aa06-4fab-b975-4e31ec8503e9

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
22	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
26	(SWC-100) Function Default Visibility	Low	Function visibility is not set.
55	(SWC-115) Authorization through tx.origin	Low	Use of "tx.origin" as a part of authorization control.
55	(SWC-115) Authorization through tx.origin	Low	Use of tx.origin as a part of authorization control.

Report for node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
https://dashboard.mythx.io/#/console/analyses/8c0312af-aa06-4fab-b975-4e31ec8503e9

Line	SWC Title	Severity	Short Description
212	(SWC-115) Authorization through tx.origin	Low	Use of tx.origin as a part of authorization control.
256	(SWC-115) Authorization through tx.origin	Low	Use of tx.origin as a part of authorization control.
282	(SWC-115) Authorization through tx.origin	Low	Use of tx.origin as a part of authorization control.
283	(SWC-115) Authorization through tx.origin	Low	Use of tx.origin as a part of authorization control.

Figure 1: Mythx Results

All relevant findings were founded in the manual code review.

THANK YOU FOR CHOOSING

HALBORN