// HALBORN

Polygon

FX Portal

Prepared by: **Halborn**

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

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 3
CONTACTS 3
1 EXECUTIVE OVERVIEW 4
1.1 INTRODUCTION 5
1.2 AUDIT SUMMARY 5
1.3 TEST APPROACH & METHODOLOGY 6
RISK METHODOLOGY 6
1.4 SCOPE 8
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 10
3 FINDINGS & TECH DETAILS 11
3.1 (HAL-01) UNCHECKED TRANSFERS - HIGH 13
Description 13
Code Location 13
Risk Level 14
Recommendations 14
3.2 (HAL-02) DOS WITH BLOCK GAS LIMIT - INFORMATIONAL 15
Description 15
Code Location 15
Risk Level 17
Recommendations 17
3.3 (HAL-03) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL 18
Description 18
Code Location 18

	Risk Level	19
	Recommendation	19
4	MANUAL TESTING	19
	Description	21
	Tests	21
5	AUTOMATED TESTING	23
5.1	STATIC ANALYSIS REPORT	27
	Description	27
	Results	27
5.2	AUTOMATED SECURITY SCAN	34
	MYTHX	34
	Description	34
	Results	34

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	06/13/2021	Ferran Celades
0.9	Document Edits	07/08/2021	Ferran Celades
1.0	Final Version	07/08/2021	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 7, 2021. The security assessment was scoped to the smart contract provided in the Github repository FxPortal 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.

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.

1.2 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.

In summary, Halborn identified few security risks, and recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts. 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 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:

/
- FxChild.sol
- FxRoot.sol

examples/
- erc1155-transfer/FxERC1155ChildTunnel.sol
- erc1155-transfer/FxERC1155RootTunnel.sol
- erc20-transfer/FxERC20ChildTunnel.sol
- erc20-transfer/FxERC20RootTunnel.sol
- erc721-transfer/FxERC721ChildTunnel.sol
- erc721-transfer/FxERC721RootTunnel.sol
- mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol
- mintable-erc20-transfer/FxMintableERC20RootTunnel.sol
- state-transfer/FxStateChildTunnel.sol
- state-transfer/FxStateRootTunnel.sol

lib/
- Address.sol
- Context.sol
- Create2.sol
- ERC1155.sol
- ERC1155Holder.sol
- ERC1155Receiver.sol
- ERC165.sol
- ERC20.sol
- ERC721.sol
- IERC1155.sol
- IERC1155MetadataURI.sol
- IERC1155Receiver.sol
- IERC165.sol
- IERC20.sol
- IERC721.sol

- IERC721Metadata.sol
- IERC721Receiver.sol
- Merkle.sol
- MerklePatriciaProof.sol
- Ownable.sol
- RLPReader.sol
- SafeMath.sol
- Strings.sol

tokens/
- FxERC1155.sol
- FxERC20.sol
- FxERC721.sol
- IFxERC1155.sol
- IFxERC20.sol
- IFxERC721.sol

tunnel/
- FxBaseChildTunnel.sol
- FxBaseRootTunnel.sol

Commit ID: d7c12b29293dbbbf662dc9c6868a99466584ca13

OUT-OF-SCOPE:

Contracts not listed above and external libraries and economics attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	0	2

LIKELIHOOD

IMPACT

			(HAL-01)	
(HAL-02) (HAL-03)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
UNCHECKED TRANSFERS	High	-
DOS WITH BLOCK GAS LIMIT	Informational	-
POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	-

FINDINGS & TECH DETAILS

3.1 (HAL-01) UNCHECKED TRANSFERS - HIGH

Description:

Several tokens do not revert in case of failure and return false. If one of these tokens is used in FX 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.

Code Location:

Listing 1: contracts/examples/erc20-transfer/FxERC20RootTunnel.sol (Lines 60)

```
53 function deposit(address rootToken, address user, uint256 amount,
       bytes memory data) public {
54     // map token if not mapped
55     if (rootToChildTokens[rootToken] == address(0x0)) {
56         mapToken(rootToken);
57     }
58
59     // transfer from depositor to this contract
60     ERC20(rootToken).transferFrom(
61         msg.sender,    // depositor
62         address(this), // manager contract
63         amount
64     );
65
66     // DEPOSIT, encode(rootToken, depositor, user, amount, extra
           data)
67     bytes memory message = abi.encode(DEPOSIT, abi.encode(
           rootToken, msg.sender, user, amount, data));
68     _sendMessageToChild(message);
69 }
```

Listing 2: contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol (Lines 33)

```
28 function deposit(address rootToken, address user, uint256 amount,
       bytes memory data) public  {
29     // map token if not mapped
30     require(rootToChildTokens[rootToken] != address(0x0), "
           FxMintableERC20RootTunnel: NO_MAPPING_FOUND");
31
32     // transfer from depositor to this contract
33     FxERC20(rootToken).transferFrom(
34         msg.sender,    // depositor
35         address(this), // manager contract
36         amount
37     );
38
39     // DEPOSIT, encode(rootToken, depositor, user, amount, extra
           data)
40     bytes memory message = abi.encode(DEPOSIT, abi.encode(
           rootToken, msg.sender, user, amount, data));
41     _sendMessageToChild(message);
42 }
```

Risk Level:

Likelihood - 4
Impact - 5

Recommendations:

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.

3.2 (HAL-02) 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 3: 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 4: 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 5: 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.

3.3 (HAL-03) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In the 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:

We noticed the use of public functions in the following contracts:

Listing 6: contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol (Lines 36)

```
36 function withdraw(address childToken, uint256 tokenId, bytes
       memory data) public {
37     IFxERC721 childTokenContract = IFxERC721(childToken);
38     // child token contract will have root token
39     address rootToken = childTokenContract.connectedToken();
```

Listing 7: contracts/tunnel/FxBaseChildTunnel.sol (Lines 33,38)

```
32 // set fxRootTunnel if not set already
33 function setFxRootTunnel(address _fxRootTunnel) public {
34     require(fxRootTunnel == address(0x0), "FxBaseChildTunnel:
           ROOT_TUNNEL_ALREADY_SET");
35     fxRootTunnel = _fxRootTunnel;
36 }
37
38 function processMessageFromRoot(uint256 stateId, address
       rootMessageSender, bytes calldata data) public override {
39     require(msg.sender == fxChild, "FxBaseChildTunnel:
           INVALID_SENDER");
```

```
    _processMessageFromRoot(stateId, rootMessageSender, data);
}
```

Listing 8: contracts/FxChild.sol (Lines 22)

```
22 function setFxRoot(address _fxRoot) public {
23     require(fxRoot == address(0x0));
24     fxRoot = _fxRoot;
25 }
```

Risk Level:

Likelihood - 1
Impact - 2

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.

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?

Tests:

First of all the source code was reviewed for uncoded features and/or missing code, for example the Listing 9 shows all the WARNING comments found.

Listing 9

```
contracts/lib/IERC721.sol
59:     * WARNING: Usage of this method is discouraged, use {safeTransferFrom} whenever possible.

contracts/lib/ERC721.sol
240:     * WARNING: Usage of this method is discouraged, use {_safeMint} whenever possible
```

After initial recon, some understanding on the different components of the platform was performed. No external dependencies were found and

all the contracts are linked to each other without relying to external parties.

However, the dependencies are flattened into the code, all of them are listed:

https://github.com/OpenZeppelin/openzeppelin-contracts

- SafeMath.sol
- Address.sol
- Context.sol
- Create2.sol
- ERC20.sol
- ERC1155.sol
- ERC1155Holder.sol
- ERC1155Receiver.sol
- ERC165.sol
- ERC721.sol
- IERC1155.sol
- IERC1155MetadataURI.sol
- IERC1155Receiver.sol
- IERC165.sol
- IERC20.sol
- IERC721.sol
- IERC721Metadata.sol
- IERC721Receiver.sol
- Ownable.sol
- Strings.sol

https://github.com/omgnetwork/plasma-mvp/blob/master/plasma/root_-chain/contracts/Merkle.sol

- Merkle.sol

https://github.com/ConsenSysMesh/rb-relay/blob/master/contracts/MerklePatriciaProof

- MerklePatriciaProof.sol

https://github.com/hamdiallam/Solidity-RLP/blob/master/contracts/RLPReader.sol

- RLPReader.sol

An example of a successful cross-chain ERC20 transfer can be seen on Figure 1.

Figure 1: Successful cross-chain transfer of an ERC20, on the left the ethereum testnet and on the right the bor chain

Validity of ERC tokens was performed, an example of those checks can be seen on Figure 2.

A successful deployment of the root tunnel can be seen on Figure 3.

As an example of the test performed, the result shown when trying to remap a token can be seen on Figure 4.

As a summary, no issues were found when performing all the stated manual tests of this section. This doesn't mean that other logical error could happen. However, on all the performed logical tests no flaws were detected.

```
# Check ERC20

## Check functions
[✓] totalSupply() is present
        [✓] totalSupply() -> () (correct return value)
        [✓] totalSupply() is view
[✓] balanceOf(address) is present
        [✓] balanceOf(address) -> () (correct return value)
        [✓] balanceOf(address) is view
[✓] transfer(address,uint256) is present
        [✓] transfer(address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] transferFrom(address,address,uint256) is present
        [✓] transferFrom(address,address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] approve(address,uint256) is present
        [✓] approve(address,uint256) -> () (correct return value)
        [✓] Approval(address,address,uint256) is emitted
[✓] allowance(address,address) is present
        [✓] allowance(address,address) -> () (correct return value)
        [✓] allowance(address,address) is view
[✓] name() is present
        [✓] name() -> () (correct return value)
        [✓] name() is view
[✓] symbol() is present
        [✓] symbol() -> () (correct return value)
        [✓] symbol() is view
[✓] decimals() is present
        [✓] decimals() -> () (correct return value)
        [✓] decimals() is view

## Check events
[✓] Transfer(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed
[✓] Approval(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed


        [✓] ERC20 has increaseAllowance(address,uint256)
```

Figure 2: Valid ERC20 implementation checks

```
>>> FxERC20RootTunnel.deploy("0x032339C9004741dDf711e08b4a4ebEdb9088D785",FxRoot[-1], FxERC20[-1], {'from':accounts[0]})
Transaction sent: 0xc40b5dfe9fe1cf9b08f78d8eada925c04bd9f664ac6bcece1b199223775572c7
  Gas price: 0.0 gwei   Gas limit: 2408327   Nonce: 86
  FxERC20RootTunnel.constructor confirmed - Block: 3079   Gas used: 2189389 (90.91%)
  FxERC20RootTunnel deployed at: 0xA64a9fAebaAAEC2392D164cCb377C2BB9FfDc1A9

<FxERC20RootTunnel Contract '0xA64a9fAebaAAEC2392D164cCb377C2BB9FfDc1A9'>
>>>
```

Figure 3: Valid Root tunnel deployment after setting up the entire environment

```
>>> FxERC20RootTunnel[0].mapToken(HalbornERC20[0], {'from':accounts[0], 'gas_limit':1000000, 'allow_revert':True})
Transaction sent: 0x0a1cfa5c12f7ead2a8066d9cd8f762930018316fd2fe2273df1b1f4e5b777182
  Gas price: 0.0 gwei   Gas limit: 1000000   Nonce: 140
  FxERC20RootTunnel.mapToken confirmed (FxERC20RootTunnel: ALREADY_MAPPED) - Block: 10606   Gas used: 22744 (2.27%)

<Transaction '0x0a1cfa5c12f7ead2a8066d9cd8f762930018316fd2fe2273df1b1f4e5b777182'>
>>>
```

Figure 4: Trying to remap an already mapped token results in an error

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:

contracts/tokens/FxERC721.sol

```
INFO:Detectors:
ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/ERC721.sol#336-355) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data)
(contracts/lib/ERC721.sol#340-351)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
FxERC721.setupMetaData(string,string)._name (contracts/tokens/FxERC721.sol#35) shadows:
        - ERC721._name (contracts/lib/ERC721.sol#23) (state variable)
FxERC721.setupMetaData(string,string)._symbol (contracts/tokens/FxERC721.sol#35) shadows:
        - ERC721._symbol (contracts/lib/ERC721.sol#26) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
FxERC721.initialize(address,address,string,string).fxManager_ (contracts/tokens/FxERC721.sol#15) lacks a zero-check on :
                - _fxManager = fxManager_ (contracts/tokens/FxERC721.sol#17)
FxERC721.initialize(address,address,string,string).connectedToken_ (contracts/tokens/FxERC721.sol#15) lacks a zero-check on :
                - _connectedToken = connectedToken_ (contracts/tokens/FxERC721.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (contracts/lib/ERC721.sol#340)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/E
RC721.sol#336-355) potentially used before declaration: retval == IERC721Receiver(to).onERC721Received.selector (contracts/lib/ERC721.sol#341)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (contracts/lib/ERC721.sol#342)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/E
RC721.sol#336-355) potentially used before declaration: reason.length == 0 (contracts/lib/ERC721.sol#343)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (contracts/lib/ERC721.sol#342)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/E
RC721.sol#336-355) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (contracts/lib/ERC721.sol#348)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Address.isContract(address) (contracts/lib/Address.sol#26-35) uses assembly
        - INLINE ASM (contracts/lib/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (contracts/lib/Address.sol#171-188) uses assembly
        - INLINE ASM (contracts/lib/Address.sol#180-183)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/ERC721.sol#336-355) uses assembly
        - INLINE ASM (contracts/lib/ERC721.sol#347-349)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (contracts/lib/Address.sol#171-188) is never used and should be removed
Address.functionCall(address,bytes) (contracts/lib/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (contracts/lib/Address.sol#89-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/lib/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/lib/Address.sol#114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/lib/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/lib/Address.sol#163-169) is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/lib/Address.sol#129-131) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/lib/Address.sol#139-145) is never used and should be removed
Address.sendValue(address,uint256) (contracts/lib/Address.sol#53-59) is never used and should be removed
Context._msgData() (contracts/lib/Context.sol#20-23) is never used and should be removed
```

contracts/tokens/FxERC20.sol

```
INFO:Detectors:
FxERC20.setupMetaData(string,string,uint8)._name (contracts/tokens/FxERC20.sol#35) shadows:
        - ERC20._name (contracts/lib/ERC20.sol#40) (state variable)
FxERC20.setupMetaData(string,string,uint8)._symbol (contracts/tokens/FxERC20.sol#35) shadows:
        - ERC20._symbol (contracts/lib/ERC20.sol#41) (state variable)
FxERC20.setupMetaData(string,string,uint8)._decimals (contracts/tokens/FxERC20.sol#35) shadows:
        - ERC20._decimals (contracts/lib/ERC20.sol#42) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
FxERC20.initialize(address,address,string,string,uint8).fxManager_ (contracts/tokens/FxERC20.sol#15) lacks a zero-check on :
                - _fxManager = fxManager_ (contracts/tokens/FxERC20.sol#17)
FxERC20.initialize(address,address,string,string,uint8).connectedToken_ (contracts/tokens/FxERC20.sol#15) lacks a zero-check on :
                - _connectedToken = connectedToken_ (contracts/tokens/FxERC20.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
SafeMath.div(uint256,uint256) (contracts/lib/SafeMath.sol#102-104) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/lib/SafeMath.sol#118-124) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/lib/SafeMath.sol#138-140) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/lib/SafeMath.sol#154-157) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/lib/SafeMath.sol#76-88) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.7.3 (contracts/lib/ERC20.sol#2) allows old versions
Pragma version0.7.3 (contracts/lib/IERC20.sol#2) allows old versions
Pragma version0.7.3 (contracts/lib/SafeMath.sol#2) allows old versions
Pragma version0.7.3 (contracts/tokens/FxERC20.sol#2) allows old versions
Pragma version0.7.3 (contracts/tokens/IFxERC20.sol#2) allows old versions
solc-0.7.3 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter FxERC20.setupMetaData(string,string,uint8)._name (contracts/tokens/FxERC20.sol#35) is not in mixedCase
Parameter FxERC20.setupMetaData(string,string,uint8)._symbol (contracts/tokens/FxERC20.sol#35) is not in mixedCase
Parameter FxERC20.setupMetaData(string,string,uint8)._decimals (contracts/tokens/FxERC20.sol#35) is not in mixedCase
Variable FxERC20._fxManager (contracts/tokens/FxERC20.sol#12) is not in mixedCase
Variable FxERC20._connectedToken (contracts/tokens/FxERC20.sol#13) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable FxERC20._connectedToken (contracts/tokens/FxERC20.sol#13) is too similar to FxERC20.initialize(address,address,string,string,uint8).connectedToken_ (contracts/tokens/FxERC20.sol#15)
Variable IFxERC20.initialize(address,address,string,string,uint8)._connectedToken (contracts/tokens/IFxERC20.sol#7) is too similar to FxERC20.initialize(address,address,string,string,uint8).
connectedToken_ (contracts/tokens/FxERC20.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

contracts/tokens/FxERC1155.sol

```
INFO:Detectors:
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (contracts/lib/ERC1155.sol#379) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (contracts/lib/ERC1155.sol#360) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (contracts/lib/ERC1155.sol#356) is a local variable never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (contracts/lib/ERC1155.sol#383) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (contracts/lib/ERC1155.sol#345-366) ignores return value by IERC1155Receiver(to).onERC1155Received(opera
tor,from,id,amount,data) (contracts/lib/ERC1155.sol#356-364)
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (contracts/lib/ERC1155.sol#368-389) ignores return value by IERC1155Receiver(to).onERC1155Batch
Received(operator,from,ids,amounts,data) (contracts/lib/ERC1155.sol#379-387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
FxERC1155.setupMetaData(string)._uri (contracts/tokens/FxERC1155.sol#28) shadows:
        - ERC1155._uri (contracts/lib/ERC1155.sol#30) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
FxERC1155.initialize(address,address,string).fxManager_ (contracts/tokens/FxERC1155.sol#12) lacks a zero-check on :
                - _fxManager = fxManager_ (contracts/tokens/FxERC1155.sol#14)
FxERC1155.initialize(address,address,string).connectedToken_ (contracts/tokens/FxERC1155.sol#12) lacks a zero-check on :
                - _connectedToken = connectedToken_ (contracts/tokens/FxERC1155.sol#15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (contracts/lib/ERC1155.sol#356)' in ERC1155._doSafeTransferAcceptanceCheck(address,ad
dress,address,uint256,uint256,bytes) (contracts/lib/ERC1155.sol#345-366) potentially used before declaration: response != IERC1155Receiver(to).onERC1155Received.selector (contracts/lib/ERC11
55.sol#357)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (contracts/lib/ERC1155.sol#360)' in ERC1155._doSafeTransferAcceptanceCheck(address,addr
ess,address,uint256,uint256,bytes) (contracts/lib/ERC1155.sol#345-366) potentially used before declaration: revert(string)(reason) (contracts/lib/ERC1155.sol#361)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (contracts/lib/ERC1155.sol#379)' in ERC1155._doSafeBatchTransferAcceptanceCh
eck(address,address,address,uint256[],uint256[],bytes) (contracts/lib/ERC1155.sol#368-389) potentially used before declaration: response != IERC1155Receiver(to).onERC1155BatchReceived.select
or (contracts/lib/ERC1155.sol#380)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (contracts/lib/ERC1155.sol#383)' in ERC1155._doSafeBatchTransferAcceptanceChec
k(address,address,address,uint256[],uint256[],bytes) (contracts/lib/ERC1155.sol#368-389) potentially used before declaration: revert(string)(reason) (contracts/lib/ERC1155.sol#384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Address.isContract(address) (contracts/lib/Address.sol#26-35) uses assembly
        - INLINE ASM (contracts/lib/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (contracts/lib/Address.sol#171-188) uses assembly
        - INLINE ASM (contracts/lib/Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (contracts/lib/Address.sol#171-188) is never used and should be removed
Address.functionCall(address,bytes) (contracts/lib/Address.sol#79-81) is never used and should be removed
Address.functionCall(address,bytes,string) (contracts/lib/Address.sol#89-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/lib/Address.sol#104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/lib/Address.sol#114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/lib/Address.sol#153-155) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/lib/Address.sol#163-169) is never used and should be removed
```

contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol

```
INFO:Detectors:
Reentrancy in FxERC20ChildTunnel._mapToken(bytes) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#70-90):
        External calls:
        - IFxERC20(childToken).initialize(address(this),rootToken,string(abi.encodePacked(name,SUFFIX_NAME)),string(abi.encodePacked(PREFIX_SYMBOL,symbol)),decimals) (contracts/examples/erc2
0-transfer/FxERC20ChildTunnel.sol#82)
        State variables written after the call(s):
        - rootToChildToken[rootToken] = childToken (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#85)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
FxBaseChildTunnel.setFxRootTunnel(address)._fxRootTunnel (contracts/tunnel/FxBaseChildTunnel.sol#33) lacks a zero-check on :
                - fxRootTunnel = _fxRootTunnel (contracts/tunnel/FxBaseChildTunnel.sol#35)
FxERC20ChildTunnel.constructor(address,address)._tokenTemplate (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#24) lacks a zero-check on :
                - tokenTemplate = _tokenTemplate (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in FxERC20ChildTunnel._mapToken(bytes) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#70-90):
        External calls:
        - IFxERC20(childToken).initialize(address(this),rootToken,string(abi.encodePacked(name,SUFFIX_NAME)),string(abi.encodePacked(PREFIX_SYMBOL,symbol)),decimals) (contracts/examples/erc2
0-transfer/FxERC20ChildTunnel.sol#82)
        Event emitted after the call(s):
        - TokenMapped(rootToken,childToken) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#86)
Reentrancy in FxERC20ChildTunnel.withdraw(address,uint256) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#29-47):
        External calls:
        - rootToken = childTokenContract.connectedToken() (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#32)
        - childTokenContract.burn(msg.sender,amount) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#43)
        Event emitted after the call(s):
        - MessageSent(message) (contracts/tunnel/FxBaseChildTunnel.sol#53)
                - _sendMessageToRoot(abi.encode(rootToken,childToken,msg.sender,amount)) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#46)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
FxERC20ChildTunnel._syncDeposit(bytes) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#92-110) uses assembly
        - INLINE ASM (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#106-108)
FxERC20ChildTunnel._isContract(address) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#113-119) uses assembly
        - INLINE ASM (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#115-117)
Create2.createClone(bytes32,address) (contracts/lib/Create2.sol#8-20) uses assembly
        - INLINE ASM (contracts/lib/Create2.sol#11-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Create2.createClone(bytes32,address) (contracts/lib/Create2.sol#8-20) is never used and should be removed
Create2.minimalProxyCreationCode(address) (contracts/lib/Create2.sol#23-29) is never used and should be removed
FxERC20ChildTunnel._mapToken(bytes) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#70-90) is never used and should be removed
FxERC20ChildTunnel._processMessageFromRoot(uint256,address,bytes) (contracts/examples/erc20-transfer/FxERC20ChildTunnel.sol#53-68) is never used and should be removed
```

contracts/examples/erc20-transfer/FxERC20RootTunnel.sol

```
INFO:Detectors:
FxERC20RootTunnel.deposit(address,address,uint256,bytes) (contracts/examples/erc20-transfer/FxERC20RootTunnel.sol#53-69) ignores return value by ERC20(rootToken).transferFrom(msg.sender,addr
ess(this),amount) (contracts/examples/erc20-transfer/FxERC20RootTunnel.sol#60-64)
FxERC20RootTunnel._processMessageFromChild(bytes) (contracts/examples/erc20-transfer/FxERC20RootTunnel.sol#72-82) ignores return value by ERC20(rootToken).transfer(to,amount) (contracts/exam
ples/erc20-transfer/FxERC20RootTunnel.sol#78-81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Reentrancy in FxERC20RootTunnel.mapToken(address) (contracts/examples/erc20-transfer/FxERC20RootTunnel.sol#30-51):
        External calls:
        - _sendMessageToChild(message) (contracts/examples/erc20-transfer/FxERC20RootTunnel.sol#42)
                - fxRoot.sendMessageToChild(fxChildTunnel,message) (contracts/tunnel/FxBaseRootTunnel.sol#68)
        State variables written after the call(s):
        - rootToChildTokens[rootToken] = childToken (contracts/examples/erc20-transfer/FxERC20RootTunnel.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
FxBaseRootTunnel.setFxChildTunnel(address)._fxChildTunnel (contracts/tunnel/FxBaseRootTunnel.sol#54) lacks a zero-check on :
                - fxChildTunnel = _fxChildTunnel (contracts/tunnel/FxBaseRootTunnel.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in FxERC20RootTunnel.mapToken(address) (contracts/examples/erc20-transfer/FxERC20RootTunnel.sol#30-51):
        External calls:
        - _sendMessageToChild(message) (contracts/examples/erc20-transfer/FxERC20RootTunnel.sol#42)
                - fxRoot.sendMessageToChild(fxChildTunnel,message) (contracts/tunnel/FxBaseRootTunnel.sol#68)
        Event emitted after the call(s):
        - TokenMapped(rootToken,childToken) (contracts/examples/erc20-transfer/FxERC20RootTunnel.sol#50)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Create2.createClone(bytes32,address) (contracts/lib/Create2.sol#8-20) uses assembly
        - INLINE ASM (contracts/lib/Create2.sol#11-17)
Merkle.checkMembership(bytes32,uint256,bytes32,bytes) (contracts/lib/Merkle.sol#5-37) uses assembly
        - INLINE ASM (contracts/lib/Merkle.sol#20-22)
RLPReader.toRlpItem(bytes) (contracts/lib/RLPReader.sol#19-31) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#26-28)
RLPReader.isList(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#60-69) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#63-65)
RLPReader.toRlpBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#74-88) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#82-84)
RLPReader.toUint(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#98-119) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#109-116)
RLPReader.toUintStrict(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#122-135) uses assembly
        - INLINE ASM (contracts/lib/RLPReader.sol#130-132)
RLPReader.toBytes(RLPReader.RLPItem) (contracts/lib/RLPReader.sol#137-152) uses assembly
```

contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol

```
INFO:Detectors:
FxMintableERC20RootTunnel.deposit(address,address,uint256,bytes) (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#28-42) ignores return value by FxERC20(rootToken).t
ransferFrom(msg.sender,address(this),amount) (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#33-37)
FxMintableERC20RootTunnel._processMessageFromChild(bytes) (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#45-76) ignores return value by tokenObj.transferFrom(addre
ss(this),to,amount) (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#71-75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
FxMintableERC20RootTunnel._processMessageFromChild(bytes) (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#45-76) ignores return value by tokenObj.approve(address(th
is),amount) (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
FxERC20.setupMetaData(string,string,uint8)._name (contracts/tokens/FxERC20.sol#35) shadows:
        - ERC20._name (contracts/lib/ERC20.sol#40) (state variable)
FxERC20.setupMetaData(string,string,uint8)._symbol (contracts/tokens/FxERC20.sol#35) shadows:
        - ERC20._symbol (contracts/lib/ERC20.sol#41) (state variable)
FxERC20.setupMetaData(string,string,uint8)._decimals (contracts/tokens/FxERC20.sol#35) shadows:
        - ERC20._decimals (contracts/lib/ERC20.sol#42) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
FxBaseRootTunnel.setFxChildTunnel(address)._fxChildTunnel (contracts/tunnel/FxBaseRootTunnel.sol#54) lacks a zero-check on :
                - fxChildTunnel = _fxChildTunnel (contracts/tunnel/FxBaseRootTunnel.sol#56)
FxMintableERC20RootTunnel.constructor(address,address,address)._rootTokenTemplate (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#23) lacks a zero-check on :
                - rootTokenTemplate = _rootTokenTemplate (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#24)
FxERC20.initialize(address,address,string,string,uint8).fxManager_ (contracts/tokens/FxERC20.sol#15) lacks a zero-check on :
                - _fxManager = fxManager_ (contracts/tokens/FxERC20.sol#17)
FxERC20.initialize(address,address,string,string,uint8).connectedToken_ (contracts/tokens/FxERC20.sol#15) lacks a zero-check on :
                - _connectedToken = connectedToken_ (contracts/tokens/FxERC20.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in FxMintableERC20RootTunnel._deployRootToken(address,string,string,uint8) (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#78-88):
        External calls:
        - FxERC20(rootToken).initialize(address(this),rootToken,name,symbol,decimals) (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#82)
        State variables written after the call(s):
        - rootToChildTokens[rootToken] = childToken (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#85)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
FxMintableERC20RootTunnel._isContract(address) (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#91-97) uses assembly
        - INLINE ASM (contracts/examples/mintable-erc20-transfer/FxMintableERC20RootTunnel.sol#93-95)
Create2.createClone(bytes32,address) (contracts/lib/Create2.sol#8-20) uses assembly
        - INLINE ASM (contracts/lib/Create2.sol#11-17)
Merkle.checkMembership(bytes32,uint256,bytes32,bytes) (contracts/lib/Merkle.sol#5-37) uses assembly
        - INLINE ASM (contracts/lib/Merkle.sol#20-22)
RLPReader.toRlpItem(bytes) (contracts/lib/RLPReader.sol#19-31) uses assembly
```

contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol

```
INFO:Detectors:
FxERC20.setupMetaData(string,string,uint8)._name (contracts/tokens/FxERC20.sol#35) shadows:
        - ERC20._name (contracts/lib/ERC20.sol#40) (state variable)
FxERC20.setupMetaData(string,string,uint8)._symbol (contracts/tokens/FxERC20.sol#35) shadows:
        - ERC20._symbol (contracts/lib/ERC20.sol#41) (state variable)
FxERC20.setupMetaData(string,string,uint8)._decimals (contracts/tokens/FxERC20.sol#35) shadows:
        - ERC20._decimals (contracts/lib/ERC20.sol#42) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
FxBaseChildTunnel.setFxRootTunnel(address)._fxRootTunnel (contracts/tunnel/FxBaseChildTunnel.sol#33) lacks a zero-check on :
                - fxRootTunnel = _fxRootTunnel (contracts/tunnel/FxBaseChildTunnel.sol#35)
FxMintableERC20ChildTunnel.constructor(address,address,address)._childTokenTemplate (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#27) lacks a zero-check on :
                - childTokenTemplate = _childTokenTemplate (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#28)
FxERC20.initialize(address,address,string,string,uint8).fxManager_ (contracts/tokens/FxERC20.sol#15) lacks a zero-check on :
                - _fxManager = fxManager_ (contracts/tokens/FxERC20.sol#17)
FxERC20.initialize(address,address,string,string,uint8).connectedToken_ (contracts/tokens/FxERC20.sol#15) lacks a zero-check on :
                - _connectedToken = connectedToken_ (contracts/tokens/FxERC20.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in FxMintableERC20ChildTunnel.withdraw(address,uint256) (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#71-96):
        External calls:
        - childTokenContract.burn(msg.sender,amount) (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#85)
        Event emitted after the call(s):
        - MessageSent(message) (contracts/tunnel/FxBaseChildTunnel.sol#53)
                - _sendMessageToRoot(abi.encode(rootToken,childToken,msg.sender,amount,metaData)) (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#95)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
FxMintableERC20ChildTunnel._syncDeposit(bytes) (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#119-137) uses assembly
        - INLINE ASM (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#133-135)
FxMintableERC20ChildTunnel._isContract(address) (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#140-146) uses assembly
        - INLINE ASM (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#142-144)
Create2.createClone(bytes32,address) (contracts/lib/Create2.sol#8-20) uses assembly
        - INLINE ASM (contracts/lib/Create2.sol#11-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
FxMintableERC20ChildTunnel._processMessageFromRoot(uint256,address,bytes) (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#102-116) is never used and should be remo
ved
FxMintableERC20ChildTunnel._syncDeposit(bytes) (contracts/examples/mintable-erc20-transfer/FxMintableERC20ChildTunnel.sol#119-137) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/lib/SafeMath.sol#102-104) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/lib/SafeMath.sol#118-124) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/lib/SafeMath.sol#138-140) is never used and should be removed
```

contracts/examples/erc1155-transfer/FxERC1155RootTunnel.sol

```
INFO:Detectors:
Reentrancy in FxERC1155RootTunnel.mapToken(address) (contracts/examples/erc1155-transfer/FxERC1155RootTunnel.sol#25-42):
        External calls:
        - _sendMessageToChild(message) (contracts/examples/erc1155-transfer/FxERC1155RootTunnel.sol#33)
                - fxRoot.sendMessageToChild(fxChildTunnel,message) (contracts/tunnel/FxBaseRootTunnel.sol#68)
        State variables written after the call(s):
        - rootToChildTokens[rootToken] = childToken (contracts/examples/erc1155-transfer/FxERC1155RootTunnel.sol#40)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (contracts/lib/ERC1155.sol#356) is a local variable never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (contracts/lib/ERC1155.sol#379) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (contracts/lib/ERC1155.sol#360) is a local variable never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (contracts/lib/ERC1155.sol#383) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (contracts/lib/ERC1155.sol#345-366) ignores return value by IERC1155Receiver(to).onERC1155Received(opera
tor,from,id,amount,data) (contracts/lib/ERC1155.sol#356-364)
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (contracts/lib/ERC1155.sol#368-389) ignores return value by IERC1155Receiver(to).onERC1155Batch
Received(operator,from,ids,amounts,data) (contracts/lib/ERC1155.sol#379-387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
FxBaseRootTunnel.setFxChildTunnel(address)._fxChildTunnel (contracts/tunnel/FxBaseRootTunnel.sol#54) lacks a zero-check on :
                - fxChildTunnel = _fxChildTunnel (contracts/tunnel/FxBaseRootTunnel.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (contracts/lib/ERC1155.sol#356)' in ERC1155._doSafeTransferAcceptanceCheck(address,ad
dress,address,uint256,uint256,bytes) (contracts/lib/ERC1155.sol#345-366) potentially used before declaration: response != IERC1155Receiver(to).onERC1155Received.selector (contracts/lib/ERC11
55.sol#357)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (contracts/lib/ERC1155.sol#360)' in ERC1155._doSafeTransferAcceptanceCheck(address,addr
ess,address,uint256,uint256,bytes) (contracts/lib/ERC1155.sol#345-366) potentially used before declaration: revert(string)(reason) (contracts/lib/ERC1155.sol#361)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (contracts/lib/ERC1155.sol#379)' in ERC1155._doSafeBatchTransferAcceptanceCh
eck(address,address,address,uint256[],uint256[],bytes) (contracts/lib/ERC1155.sol#368-389) potentially used before declaration: response != IERC1155Receiver(to).onERC1155BatchReceived.select
or (contracts/lib/ERC1155.sol#380)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (contracts/lib/ERC1155.sol#383)' in ERC1155._doSafeBatchTransferAcceptanceChec
k(address,address,address,uint256[],uint256[],bytes) (contracts/lib/ERC1155.sol#368-389) potentially used before declaration: revert(string)(reason) (contracts/lib/ERC1155.sol#384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in FxERC1155RootTunnel.mapToken(address) (contracts/examples/erc1155-transfer/FxERC1155RootTunnel.sol#25-42):
        External calls:
        - _sendMessageToChild(message) (contracts/examples/erc1155-transfer/FxERC1155RootTunnel.sol#33)
                - fxRoot.sendMessageToChild(fxChildTunnel,message) (contracts/tunnel/FxBaseRootTunnel.sol#68)
        Event emitted after the call(s):
        - TokenMapped(rootToken,childToken) (contracts/examples/erc1155-transfer/FxERC1155RootTunnel.sol#41)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (contracts/lib/Address.sol#26-35) uses assembly
        - INLINE ASM (contracts/lib/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (contracts/lib/Address.sol#171-188) uses assembly
```

contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol

```
Compilation warnings/errors on contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol:
Warning: Unused local variable.
   --> contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol:102:31:
    |
102 |           (address rootToken, address depositor, address user, uint256 id, uint256 ...
    |                               ^^^^^^^^^^^^^^^^

Warning: Unused local variable.
   --> contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol:112:31:
    |
112 |           (address rootToken, address depositor, address user, uint256[] memory id ...
    |                               ^^^^^^^^^^^^^^^^


INFO:Detectors:
Reentrancy in FxERC1155ChildTunnel._mapToken(bytes) (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#85-99):
        External calls:
        - IFxERC1155(childToken).initialize(address(this),rootToken,string(abi.encodePacked(uri))) (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#93)
        State variables written after the call(s):
        - rootToChildToken[rootToken] = childToken (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#95)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
FxBaseChildTunnel.setFxRootTunnel(address)._fxRootTunnel (contracts/tunnel/FxBaseChildTunnel.sol#33) lacks a zero-check on :
                - fxRootTunnel = _fxRootTunnel (contracts/tunnel/FxBaseChildTunnel.sol#35)
FxERC1155ChildTunnel.constructor(address,address)._tokenTemplate (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#22) lacks a zero-check on :
                - tokenTemplate = _tokenTemplate (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in FxERC1155ChildTunnel._mapToken(bytes) (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#85-99):
        External calls:
        - IFxERC1155(childToken).initialize(address(this),rootToken,string(abi.encodePacked(uri))) (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#93)
        Event emitted after the call(s):
        - TokenMapped(rootToken,childToken) (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#96)
Reentrancy in FxERC1155ChildTunnel.withdraw(address,uint256,uint256,bytes) (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#27-42):
        External calls:
        - rootToken = childTokenContract.connectedToken() (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#29)
        - childTokenContract.burn(msg.sender,id,amount) (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#38)
        Event emitted after the call(s):
        - MessageSent(message) (contracts/tunnel/FxBaseChildTunnel.sol#53)
                - _sendMessageToRoot(message) (contracts/examples/erc1155-transfer/FxERC1155ChildTunnel.sol#41)
```

contracts/examples/erc721-transfer/FxERC721RootTunnel.sol

```
INFO:Detectors:
Reentrancy in FxERC721RootTunnel.mapToken(address) (contracts/examples/erc721-transfer/FxERC721RootTunnel.sol#38-58):
        External calls:
        - _sendMessageToChild(message) (contracts/examples/erc721-transfer/FxERC721RootTunnel.sol#49)
                - fxRoot.sendMessageToChild(fxChildTunnel,message) (contracts/tunnel/FxBaseRootTunnel.sol#68)
        State variables written after the call(s):
        - rootToChildTokens[rootToken] = childToken (contracts/examples/erc721-transfer/FxERC721RootTunnel.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/ERC721.sol#336-355) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data)
(contracts/lib/ERC721.sol#340-351)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
FxBaseRootTunnel.setFxChildTunnel(address)._fxChildTunnel (contracts/tunnel/FxBaseRootTunnel.sol#54) lacks a zero-check on :
                - fxChildTunnel = _fxChildTunnel (contracts/tunnel/FxBaseRootTunnel.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (contracts/lib/ERC721.sol#340)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/E
RC721.sol#336-355) potentially used before declaration: retval == IERC721Receiver(to).onERC721Received.selector (contracts/lib/ERC721.sol#341)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (contracts/lib/ERC721.sol#342)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/E
RC721.sol#336-355) potentially used before declaration: reason.length == 0 (contracts/lib/ERC721.sol#343)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (contracts/lib/ERC721.sol#342)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/E
RC721.sol#336-355) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (contracts/lib/ERC721.sol#348)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in FxERC721RootTunnel.mapToken(address) (contracts/examples/erc721-transfer/FxERC721RootTunnel.sol#38-58):
        External calls:
        - _sendMessageToChild(message) (contracts/examples/erc721-transfer/FxERC721RootTunnel.sol#49)
                - fxRoot.sendMessageToChild(fxChildTunnel,message) (contracts/tunnel/FxBaseRootTunnel.sol#68)
        Event emitted after the call(s):
        - TokenMapped(rootToken,childToken) (contracts/examples/erc721-transfer/FxERC721RootTunnel.sol#57)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (contracts/lib/Address.sol#26-35) uses assembly
        - INLINE ASM (contracts/lib/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (contracts/lib/Address.sol#171-188) uses assembly
        - INLINE ASM (contracts/lib/Address.sol#180-183)
Create2.createClone(bytes32,address) (contracts/lib/Create2.sol#8-20) uses assembly
        - INLINE ASM (contracts/lib/Create2.sol#11-17)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (contracts/lib/ERC721.sol#336-355) uses assembly
        - INLINE ASM (contracts/lib/ERC721.sol#347-349)
Merkle.checkMembership(bytes32,uint256,bytes32,bytes) (contracts/lib/Merkle.sol#5-37) uses assembly
        - INLINE ASM (contracts/lib/Merkle.sol#20-22)
RLPReader.toRlpItem(bytes) (contracts/lib/RLPReader.sol#19-31) uses assembly
```

contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol

```
Compilation warnings/errors on contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol:
Warning: Unused local variable.
   --> contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol:100:29:
    |
100 |         (address rootToken, address depositor, address to, uint256 tokenId, byte ...
    |                             ^^^^^^^^^^^^^^^^


INFO:Detectors:
Reentrancy in FxERC721ChildTunnel._mapToken(bytes) (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#77-97):
        External calls:
        - IFxERC721(childToken).initialize(address(this),rootToken,string(abi.encodePacked(name,SUFFIX_NAME)),string(abi.encodePacked(PREFIX_SYMBOL,symbol))) (contracts/examples/erc721-trans
fer/FxERC721ChildTunnel.sol#89)
        State variables written after the call(s):
        - rootToChildToken[rootToken] = childToken (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
FxBaseChildTunnel.setFxRootTunnel(address)._fxRootTunnel (contracts/tunnel/FxBaseChildTunnel.sol#33) lacks a zero-check on :
                - fxRootTunnel = _fxRootTunnel (contracts/tunnel/FxBaseChildTunnel.sol#35)
FxERC721ChildTunnel.constructor(address,address)._tokenTemplate (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#25) lacks a zero-check on :
                - tokenTemplate = _tokenTemplate (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in FxERC721ChildTunnel._mapToken(bytes) (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#77-97):
        External calls:
        - IFxERC721(childToken).initialize(address(this),rootToken,string(abi.encodePacked(name,SUFFIX_NAME)),string(abi.encodePacked(PREFIX_SYMBOL,symbol))) (contracts/examples/erc721-trans
fer/FxERC721ChildTunnel.sol#89)
        Event emitted after the call(s):
        - TokenMapped(rootToken,childToken) (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#93)
Reentrancy in FxERC721ChildTunnel.withdraw(address,uint256,bytes) (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#36-54):
        External calls:
        - rootToken = childTokenContract.connectedToken() (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#39)
        - childTokenContract.burn(tokenId) (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#50)
        Event emitted after the call(s):
        - MessageSent(message) (contracts/tunnel/FxBaseChildTunnel.sol#53)
                - _sendMessageToRoot(abi.encode(rootToken,childToken,msg.sender,tokenId,data)) (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
FxERC721ChildTunnel._isContract(address) (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#109-115) uses assembly
        - INLINE ASM (contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol#111-113)
Create2.createClone(bytes32,address) (contracts/lib/Create2.sol#8-20) uses assembly
        - INLINE ASM (contracts/lib/Create2.sol#11-17)
```

Some of the tests reported a lack of zero check on the addresses. However, those addresses can only be set once. This is checked by making sure

that the initial state of the address is 0x0.

On the contracts/tokens/FxERC1155.sol the function _checkOnERC721Received is reported as ignoring the return value. This is a false positive since the calling function is checked with a try catch block.

The variable shadowing on the contract/tokens/* is done in purpose to override the values on the implementation.

All the re-entrancy issues reported as medium are false positives, since the values stored after the call are already initialized prior this call and will not be altered.

The unchecked return values during the transfer are explained on the report as “(HAL-01) UNCHECKED TRANSFERS” vulnerability.

5.2 AUTOMATED SECURITY SCAN

MYTHX:

Description:

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:

```
Report for contracts/FxRoot.sol
https://dashboard.mythx.io/#/console/analyses/4b8ef765-b519-471a-b846-c2fdb60bf72c
```

Line	SWC Title	Severity	Short Description
20	(SWC-100) Function Default Visibility	Low	Function visibility is not set.
24	(SWC-000) Unknown	Medium	Function could be marked as external.
29	(SWC-000) Unknown	Medium	Function could be marked as external.

Figure 5: contracts/FxRoot.sol

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”.

```
Report for contracts/FxChild.sol
https://dashboard.mythx.io/#/console/analyses/c491b725-7648-45b5-bd68-990029e7d411
```

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

Figure 6: contracts/FxChild.sol

```
Report for contracts/lib/RLPReader.sol
https://dashboard.mythx.io/#/console/analyses/1bcdf341-a078-440a-b113-054ef242e70b
```

Line	SWC Title	Severity	Short Description
50	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.
166	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.
241	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.

Figure 7: contracts/lib/RLPReader.sol

```
Report for examples/erc721-transfer/FxERC721ChildTunnel.sol
https://dashboard.mythx.io/#/console/analyses/a3f57762-241c-4185-9367-956402f7b7ad
```

Line	SWC Title	Severity	Short Description
25	(SWC-100) Function Default Visibility	Low	Function visibility is not set.
36	(SWC-000) Unknown	Medium	Function could be marked as external.
100	(SWC-131) Presence of unused variables	Low	Unused local variable "depositor".

```
Report for lib/Create2.sol
https://dashboard.mythx.io/#/console/analyses/a3f57762-241c-4185-9367-956402f7b7ad
```

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

```
Report for tunnel/FxBaseChildTunnel.sol
https://dashboard.mythx.io/#/console/analyses/a3f57762-241c-4185-9367-956402f7b7ad
```

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

Figure 8: contracts/examples/erc721-transfer/FxERC721ChildTunnel.sol

```
Report for examples/erc721-transfer/FxERC721RootTunnel.sol
https://dashboard.mythx.io/#/console/analyses/54579a33-5cd9-4c1b-aec6-e600e16611db
```

Line	SWC Title	Severity	Short Description
22	(SWC-100) Function Default Visibility	Low	Function visibility is not set.

```
Report for lib/ERC721.sol
https://dashboard.mythx.io/#/console/analyses/54579a33-5cd9-4c1b-aec6-e600e16611db
```

Line	SWC Title	Severity	Short Description
372	(SWC-131) Presence of unused variables	Low	Unused function parameter "tokenId".
372	(SWC-131) Presence of unused variables	Low	Unused function parameter "to".
372	(SWC-131) Presence of unused variables	Low	Unused function parameter "from".

Figure 9: contracts/examples/erc721-transfer/FxERC721RootTunnel.sol

THANK YOU FOR CHOOSING

// HALBORN