MythX

REPORT 602B8E16EF24100018095DDF

Created	Tue Feb 16 2021 09:19:18 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	bl33234679@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
1fcfe222-dc9c-47ba-8af2-e59122f0439a	Whitelist.sol	3

Analysis 1fcfe222-dc9c-47ba-8af2-e59122f0439a MythX

Started	Tue Feb 16 2021 09:19:23 GMT+0000 (Coordinated Universal Time)
Finished	Tue Feb 16 2021 10:05:18 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Whitelist.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	3

ISSUES

LOW

SWC-100

Function visibility is not set.

The function definition of "null" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source file

Whitelist.sol

Locations

```
12 bytes32 public constant WHITELIST_ROLE = keccak256("WHITELIST_ROLE");
13
14 constructor() {
15 _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
16 }
17
18 function preRelayedCall(
```

LOW

SWC-107

A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Source file

GSN/BasePaymaster.sol

Locations

```
97 receive() external virtual payable {
98 require(address(relayHub) != address(0), "relay hub address not set");
99 relayHub.depositFor{value:msg.value}(address(this));
100 }
```

LOW

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

SWC-123

Source file

GSN/BasePaymaster.sol

Locations

```
97  receive() external virtual payable {
98  require(address(relayHub) != address(0), "relay hub address not set");
99  relayHub.depositFor{value:msg.value}(address(this));
100 }
```

Source file

Whitelist.sol

Locations

```
8
9   /// @title GNS whitelist contract
10  contract Whitelist is IWhitelist, AccessControl, BasePaymaster {
11
12  bytes32 public constant WHITELIST_ROLE = keccak256("WHITELIST_ROLE");
13
14  constructor() {
15  _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());
16  }
17
18  function preRelayedCall(
19  GsnTypes.RelayRequest calldata relayRequest,
20  bytes calldata signature,
21  bytes calldata approvalData,
22  uint256 maxPossibleGas
23  )
24  external
25  view
26  override
27  relayHubOnly
28  returns (
29  bytes memory context,
30  bool rejectOnRecipientRevert
31  )
32  {
33  require(
34  inWhitelist(relayRequest.request.from) ||
35  isAdmin(relayRequest.request.from),
36  "Address is not in whitelist"
37  );
38
39  _verifyForwarder(relayRequest);
40  _verifySignature(relayRequest, signature);
41  // _verifySigner or not ?
42  // _verifyGaslimit or not ?
43  return ("PreRelayedCall success", false);
44  }
45
46  function postRelayedCall(
47  bytes calldata context,
48  bool success,
49  uint256 gasUseWithoutPost,
50  GsnTypes.RelayData calldata relayData
51  )
52  external
53  override
```

```
relayHubOnly
{
}

function versionPaymaster()
external
pure
override
returns (string memory)
{
return "2.1.0";
}

/// @notice Quries whether an address has role admin
/// @param _account The address to be queried
/// @return `True` if `_account` has role admin
function isAdmin(address _account)
public
view
override
returns (bool)
{
return hasRole(DEFAULT_ADMIN_ROLE, _account);
}

/// @notice Queries whether an address has role whitelist
/// @param _account The address to be queried
/// @return `True` if `_account` has role whitelist
function inWhitelist(address _account)
public
view
override
returns (bool)
{
return hasRole(WHITELIST_ROLE, _account);
}

/// @notice Remove an address from role admin
/// @dev It throws if `msg.sender` does no have role admin.
/// It does not throw if address is not in role admin.
/// @param _account The address to be removed
function removeAdmin(address _account)
external
override
{
revokeRole(DEFAULT_ADMIN_ROLE, _account);
}

/// @notice Insert an address into role whitelist
/// @dev It does not throw if address already has role whitelist
/// @param _account The address to be inserted
function addWhitelist(address _account)
external
override
{
grantRole(WHITELIST_ROLE, _account);
}

/// @notice Remove an address from role whitelist
/// @dev It throws if `msg.sender` does no have role admin.
/// It does not throw if address is not in role whitelist.
/// @param _account The address to be removed
function removeWhitelist(address _account)
```

```
external
override
{
revokeRole(WHITELIST_ROLE, _account);
}

/// @notice Insert an address into role admin
/// @dev It does not throw if address already has role admin
/// @param _account The address to be inserted
function addAdmin(address _account)
external
override
{
grantRole(DEFAULT_ADMIN_ROLE, _account);
}
}
```