SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Prepared for:	Telcoin
Prepared by:	Sherlock
Lead Security Expert:	bughuntoor
Dates Audited:	March 4 - March 7, 2024
Prepared on:	March 22, 2024

Introduction

Telcoin leverages blockchain technology to provide access to low-cost, high-quality decentralized financial products for every mobile phone user in the world.

Scope

Repository: telcoin/telcoin-contracts

Branch: main

Commit: 817aff2da48312414a32f3723f2791f52491ae8d

For the detailed scope, see the contest details.

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues found

Medium	High
2	0

Issues not fixed or acknowledged

Medium	High
0	0

Issue M-1: Blacklisted accounts can still transact.

Source: https://github.com/sherlock-audit/2024-02-telcoin-platform-audit-update-judging/issues/4

Found by

0xkmg, Krace, Tendency, ZanyBonzy, ZdravkoHr., blutorque, bughuntoor, cawfree, merlin, neocrao, sa9933, smbv-1923, turvec

Summary

Accounts that have been blacklisted by the BLACKLISTER_ROLE continue to transact normally.

Vulnerability Detail

Currently, the only real effect of blacklisting an account is the seizure of Stablecoin funds:

```
/**
 * @notice Overrides Blacklist function to transfer balance of a blacklisted
↪  user to the caller.
 * @dev This function is called internally when an account is blacklisted.
 * @param user The blacklisted user whose balance will be transferred.
 */
function _onceBlacklisted(address user) internal override {
  _transfer(user, _msgSender(), balanceOf(user));
}
```

However, following a call to addBlackList(address), the blacklisted account may continue to transact using Stablecoin.

Combined with previous audit reports, which attest to the blacklist function's susceptibility to frontrunning, the current implementation of the blacklist operation can effectively be considered a no-op.

Impact

Medium, as this the failure of a manually administered security feature.

Code Snippet

Stablecoin.sol

Tool used

Manual Review

Recommendation

ERC20s that enforce blacklists normally prevent a sanctioned address from being able to transact:

Stablecoin.sol

```
+ error Blacklisted(address account);

+function _update(address from, address to, uint256 value) internal virtual
 ↪  override {
+
+  if (blacklisted(from)) revert Blacklisted(from);
+  if (blacklisted(to)) revert Blacklisted(to);
+
+  super._update(from, to, value);
+}
```

Discussion

sherlock-admin4

1 comment(s) were left on this issue during the judging contest.

takarez commented:

> valid; high(1)

sherlock-admin4

The protocol team fixed this issue in PR/commit https://github.com/telcoin/telcoin-contracts/pull/3.

spacegliderrrr

Fix looks good, blacklisted addresses can no longer send and receive tokens.

sherlock-admin4

The Lead Senior Watson signed off on the fix.

Issue M-2: Not all ERC20 tokens can be bridged because of hardcoded `PREDICATE_ADDRESS`

Source: https://github.com/sherlock-audit/2024-02-telcoin-platform-audit-update-judging/issues/26

Found by

ZdravkoHr., bughuntoor, cawfree

Summary

`PREDICATE_ADDRESS` in BridgeRelay.sol is hardcoded to the ERC20Predicate. This means that any tokens that use other predicates will not be bridgeable.

Vulnerability Detail

When the BridgeRelay.transferERCToBridge gets called, it approves the hardcoded `ERC20Predicate` to use the ERC20 tokens. However, the bridge uses more than one predicate to lock the tokens. Here the bridge retrieves the predicate address based on the type of token to be bridged. If a token that uses different predicate than the hardcoded is sent to the `BridgeRelay`, it will be forever stuck there since the right predicate will not have approval to transfer it. There also exists a risk that a token can change its predicate at any time.

Impact

Tokens that use different predicate will be forever stuck in the contract.

Code Snippet

```
function transferERCToBridge(IERC20 token) internal {
    //zero out approvals
    token.forceApprove(PREDICATE_ADDRESS, 0);
    // increase approval to necessary amount
    token.safeIncreaseAllowance(
        PREDICATE_ADDRESS,
        token.balanceOf(address(this))
    );
    //deposit
    POS_BRIDGE.depositFor(
        address(this),
        address(token),
        abi.encodePacked(token.balanceOf(address(this)))
    );
}
```

```
address predicateAddress = typeToPredicate[tokenType];
require(
    predicateAddress != address(0),
    "RootChainManager: INVALID_TOKEN_TYPE"
);
require(
    user != address(0),
    "RootChainManager: INVALID_USER"
);

ITokenPredicate(predicateAddress).lockTokens(
    _msgSender(),
    user,
    rootToken,
    depositData
);
```

Tool used

Manual Review

Recommendation

Instead of hardcoding the predicate, query the bridge when transferring the ERC20 token.

```
    function transferERCToBridge(IERC20 token) internal {
+      bytes32 tokenType = POS_BRIDGE.tokenToType(address(token));
+      bytes32 predicateAddress= POS_BRIDGE.typeToPredicate(tokenType);
        //zero out approvals
        token.forceApprove(PREDICATE_ADDRESS, 0);
        // increase approval to necessary amount
        token.safeIncreaseAllowance(
-           PREDICATE_ADDRESS,
+           predicateAddress
            token.balanceOf(address(this))
        );
        //deposit
        POS_BRIDGE.depositFor(
            address(this),
            address(token),
            abi.encodePacked(token.balanceOf(address(this)))
        );
    }
```

Discussion

sherlock-admin3

1 comment(s) were left on this issue during the judging contest.

takarez commented:

> invalid

amshirif

This isnt an issue. So we dont provide bridging as a service. We use it as a recovery method for people who send their tokens on the wrong network as a means of recovering them. However, tokens that we dont support we dont do anything with. We just allow them to sit there. The reason being is that if we were to bridging them over they still couldnt send them on our platform. We dont support and bridged tokens that us different predicates. Whats more, since we'd have to provide information in such as which bridge to use, it would require it to be a privileged method, which is something we're actively avoiding. We wish the function to be callable to anyone who wishes to attempt to push their tokens across.

WangSecurity

After additional discussion we came to the conclusion that this issue is valid, cause the predicated address may be changed by polygon at any moment and users funds will be lost with to method to recover them. Therefore, it is a Medium.

sherlock-admin4

The protocol team fixed this issue in PR/commit https://github.com/telcoin/telcoin-contracts/pull/8.

ABDuullahi

escalate

This is invalid according to sherlock rules

> Future issues: Issues that result out of a future integration/implementation that was not mentioned in the docs/README or because of a future change in the code (as a fix to another issue) are not valid issues.

polygon changing the address is a future implementation i believe that no one is sure whether it can be change or not, and also the sponsor confirm that they don't provide it as a service but rather to help users in case of sending it to a wrong network which also falls under user mistake

sherlock-admin2

> escalate
>
> This is invalid according to sherlock rules
>
> > Future issues: Issues that result out of a future integration/implementation that was not mentioned in the docs/README or because of a future change in the code (as a fix to another issue) are not valid issues.
>
> polygon changing the address is a future implementation i believe that no one is sure whether it can be change or not, and also the sponsor confirm that they don't provide it as a service but rather to help users in case of sending it to a wrong network which also falls under user mistake

You've created a valid escalation!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

WangSecurity

I can see where your assumption comes from and at some point I also thought it should be invalidated due to this rule. But, LSW explained what is meant why this rule cannot be applied here. The integration/implementation will remain the same - the code in scope will not change in any way.

The Future Issues rule means that the bug will arise due to changes in the code in scope of the contest. This time the issue might arise due to changes in the third party contract. It means telcoin contracts are not properly adjusted for the contract it interacts with. Since telcoin has no power over the polygon bridge, it might properly interact with it and account for its specifics.

Because of these reasons, we decided to keep it valid and medium severity. Hope it answers your questions.

ABDuullahi

i guess i misunderstood the `Future Issue` rule, but i will nonetheless leave it to the head of judging to interpret it more clearly or shed more light on what he thinks.

Czar102

From my understanding, despite this issue doesn't qualify for the rule mentioned in the escalation, it qualifies for another: opportunity loss.

The funds sent on other networks would normally be lost, but the protocol created a way to retrieve them. That may not work for some tokens, but it doesn't mean that the funds are lost – they are simply not retrieved. They would normally be lost when sent to a wrong address.

The protocol could be allowing for a wider group of tokens to be retrieved, yes. The fact that it doesn't is grounds for a good informational issue.

Please let me know if I misunderstood anything and in case I didn't, I'm planning to accept the escalation and invalidate the issue.

deadrosesxyz

@czar102 I don't think this should fall under "opportunity loss". Predicates can be changed by the polygon team, meaning that it's not simply a case of "it could support more tokens".

The contract is supposed to be able to retrieve tokens X. In certain scenario, it can happen that there's no way to retrieve same said tokens X, although contract is supposed to be able to do so. Since the likelihood is somewhat low, but would break the whole contract's functionality, Medium seems appropriate.

ABDuullahi

The whole report is based on the idea that `not all ERC20 can be bridged`. First the protocol doesn't provide the said bridging as a service, second is there is no `other token`. it should be invalidated i believe.

ZdravkoHr

@ABDuullahi, that's just the way i formatted the title. If the predicate is changed, no tokens at can be bridged at all, so "not all" is technically still true

amshirif

So we will be implementing a fix, but in a slightly different way than recommended. We will not take the predicate as a param because this would then need to become a protected function. Instead the ERC20 rescue function will allow tokens to be removed, however it will require the current predicate to prevent bridging.

ABDuullahi

@ZdravkoHr `Not all` in ur case means that all should be able to be bridged.

ZdravkoHr

@amshirif, thanks for your feedback. By the way, if you imolememt the fix, the predicate will be loaded from the bridge's storage, it won't be passed from the user. So the function can continue being permisionless

Czar102

> In certain scenario, it can happen that there's no way to retrieve same said tokens X, although contract is supposed to be able to do so. Since the likelihood is somewhat low, but would break the whole contract's functionality, Medium seems appropriate.

The funds are normally lost, and no users should send funds on the wrong chain in the usage of the protocol. Not recovering them is opportunity loss. This is not loss of funds from this point of view.

This can also be considered breaking smart contract functionality, but this functionality isn't core, so I'll consider it Low on these grounds.

deadrosesxyz

> Breaks core contract functionality, rendering the contract useless or leading to loss of funds.

Hey, if I understand this requirement for Medium severity, this issues does in fact break the core functionality of this particular contract and does render it useless.

Correct me if I'm wrong, but the severity depends on whether the broken functionality is core for the said smart contract and not based on whether the broken smart contract is core of the project?

WangSecurity

I agree with deadroses above. Even if it's not the main feature of Telcoin and doesn't break core protocol's functionality, it definetely renders contract useless. Thus, if the contract is useless, I assume it's equal to broken contract functionality of the particular contract (not the protocol). And as deadroses mentioned, the rule for medium is breaking the core functionality of the contract, not the protocol as a whole.

Czar102

After some internal discussions, I agree that this issue should stay as is. Planning to reject the escalation.

Czar102

Result: Medium Has duplicates

sherlock-admin3

Escalations have been resolved successfully!

Escalation status:

- ABDUullahi: rejected

spaceglidderrrr

Fix looks good, stuck tokens can now be retrieved by contract owner.

sherlock-admin4

The Lead Senior Watson signed off on the fix.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users’ sole risk and is the users’ responsibility.