SHERLOCK

SHERLOCK SECURITY REVIEW FOR

SHERLOCK

Prepared for:	Telcoin
Prepared by:	Sherlock
Lead Security Expert:	WATCHPUG
Dates Audited:	November 17 - November 20, 2022
Prepared on:	December 6, 2022

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

```
contracts/interfaces/IPlugin.sol
contracts/StakingModule.sol
contracts/libraries/AuxDataLib.sol
contracts/SimplePlugin.sol
contracts/feebuyback/IFeeBuyback.sol
contracts/feebuyback/TieredOwnership.sol
contracts/feebuyback/ISimplePlugin.sol
contracts/feebuyback/FeeBuyback.sol
```

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
6	0

Issues not fixed or acknowledged

Medium	High
0	0

Security experts who found valid issues

WATCHPUG	0x4non	rvierdiiev
hickuphh3	eierina	Deivitto
hyh	Bnke0x0	aphak5010
yixxas	0xheynacho	0xAgro
cccz	pashov	
rotcivegaf	Mukund	

Issue M-1: Flashloan `TEL` tokens to stake and exit in the same block can fake a huge amount of stake with minimal material cost

Source: https://github.com/sherlock-audit/2022-11-telcoin-judging/issues/83

Found by

WATCHPUG

Summary

`Checkpoints#getAtBlock()` can be faked with falshloan as it may return the value of the first checkpoint in the same block.

Vulnerability Detail

`Checkpoints#getAtBlock()` will return the value on check point #0 when there are two check points in the same block (#0 and #1).

Therefore, one can take a falshloan of TEL tokens to stake and exit in the same block, which will create two checkpoints.

Impact

Malicious user can fake their stake to gain a high percentage rewards with falshloan and avoid slashing.

Code Snippet

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L147-L149

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L403-L406

Tool used

Manual Review

Recommendation

Consider requiring the `exit` to be at least 1 block later than the blocknumber of the original stake.

Discussion

amshirif

https://github.com/telcoin/telcoin-staking/pull/9

dmitriia

Escalate for 30 USDC I apologize if missed some point, but how this can be used to drain funds from the protocol?

stakedByAt is used only at balanceOfAt(): https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L98-L104

Which is stand-alone view function: https://github.com/sherlock-audit/2022-11-telcoin/search?q=balanceOfAt

Slashing example is clear, but it uses latest(), not stakedByAt(): https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L403-L406 https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L356-L360

For the latest() Checkpoints will return the latest entry, after the flash loan: https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/utils/CheckpointsUpgradeable.sol#L94-L100

I.e. if an attacker front-run the slashing, his flashloan will be finished when slashing run and the latest, small, entry will be used in claiming.

Am I missing something here?

If not the severity should be Med as view function is impacted, which can backfire downstream, but that's an assumption typically meaning downgrading the severity.

sherlock-admin

> Escalate for 30 USDC I apologize if missed some point, but how this can be used to drain funds from the protocol?
>
> stakedByAt is used only at balanceOfAt(): https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L98-L104
>
> Which is stand-alone view function: https://github.com/sherlock-audit/2022-11-telcoin/search?q=balanceOfAt
>
> Slashing example is clear, but it uses latest(), not stakedByAt(): https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L403-L406 https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L356-L360
>
> For the latest() Checkpoints will return the latest entry, after the flash loan: https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/utils/CheckpointsUpgradeable.sol#L94-L100

> I.e. if an attacker front-run the slashing, his flashloan will be finished when slashing run and the latest, small, entry will be used in claiming.
>
> Am I missing something here?
>
> If not the severity should be Med as view function is impacted, which can backfire downstream, but that's an assumption typically meaning downgrading the severity.

You've created a valid escalation for 30 USDC!

To remove the escalation from consideration: Delete your comment. To change the amount you've staked on this escalation: Edit your comment **(do not create a new comment)**.

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

hrishibhat

Escalation accepted

After consulting with the sponsors, judges took a deep dive on the issue and decided to make this a medium severity.

sherlock-admin

> Escalation accepted
>
> After consulting with the sponsors, judges took a deep dive on the issue and decided to make this a medium severity.

This issue's escalations have been accepted!

Contestants' payouts and scores will be updated according to the changes made on this issue.

Issue M-2: Unsafe ERC20 methods

Source: https://github.com/sherlock-audit/2022-11-telcoin-judging/issues/82

Found by

WATCHPUG, yixxas, hyh, Bnke0x0, hickuphh3, 0x4non, pashov, rvierdiiev, Deivitto, eierina, rotcivegaf, aphak5010, 0xheynacho, Mukund, 0xAgro

Summary

Using unsafe ERC20 methods can revert the transaction for certain tokens.

Vulnerability Detail

There are many Weird ERC20 Tokens that won't work correctly using the standard `IERC20` interface.

For example, `IERC20(token).transferFrom()` and `IERC20(token).transfer()` will fail for some tokens as they may not conform to the standard IERC20 interface. And if `_aggregator` does not always consume all the allowance given at L72, the transaction will also revert on the next call, because there are certain tokens that do not allow approval of a non-zero number when the current allowance is not zero (eg, USDT).

Impact

The contract will malfunction for certain tokens.

Code Snippet

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/fee-buyback/FeeBuyback.sol#L94-L97

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/fee-buyback/FeeBuyback.sol#L47-L82

Tool used

Manual Review

Recommendation

Consider using `SafeERC20` for `transferFrom`, `transfer` and `approve`.

Discussion

amshirif

https://github.com/telcoin/telcoin-staking/pull/6

Issue M-3: `FeeBuyback` native token can not be rescued

Source: https://github.com/sherlock-audit/2022-11-telcoin-judging/issues/80

Found by

WATCHPUG

Summary

Lack of methods to rescue native tokens trapped in the `FeeBuyback` contract.

Vulnerability Detail

Like ERC20 tokens, the native token may also get stuck in the `FeeBuyback` contract for all sorts of reasons.

For example, at L77, the `_aggregator` is called with a `msg.value`, which means that the native token can be used as an inToken for the swap. Therefore, part of the input native token can be sent back to the FeeBuyback contract as a leftover.

However, the current implementation of `rescueERC20()` only supports rescue ERC20 tokens.

Impact

The leftover native tokens trapped in the contract can not be rescued.

Code Snippet

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/fee-buyback/FeeBuyback.sol#L77-L78

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/fee-buyback/FeeBuyback.sol#L94-L97

Tool used

Manual Review

Recommendation

Consider adding support to rescue native tokens.

Discussion

amshirif

https://github.com/telcoin/telcoin-staking/pull/10

Issue M-4: Native funds can be lost by submit() as msg.value isn't synchronized with amount

Source: https://github.com/sherlock-audit/2022-11-telcoin-judging/issues/76

Found by

hyh

Summary

When used with native funds FeeBuyback#submit() doesn't check for the `amount` argument to correspond to `msg.value` actually linked to the call.

Vulnerability Detail

This can lead either to bloating or to underpaying of the actual fee depending on the mechanics that will be used to call submit(). I.e. as two values can differ, and only one can be correct, the difference is a fund loss either to the `owner` (when the fee is overpaid) or to `recipient` (when the fee is underpaid vs correct formula).

Impact

Net impact is a fund loss proportional to the difference of the `amount` and `msg.value`. This can be either incomplete setup (native funds case isn't fully covered in a calling script) or an operational mistake (it is covered correctly, but a wrong value was occasionally left from a testing, and so on) situation.

Setting the severity to be medium as this is conditional on the actual usage of submit().

Code Snippet

submit() uses `msg.value`, which can differ from `amount`:

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/fee-buyback/FeeBuyback.sol#L35-L82

```
/**
 * @notice submits wallet transactions
 * @dev a secondary swap may occur
 * @dev staking contract updates may be made
 * @dev function can be paused
 * @param wallet address of the primary transaction
 * @param walletData bytes wallet data for primary transaction
 * @param token address the token that is being swapped from in a secondary
 ↪  transaction
 * @param amount uint256 the quantity of the token being swapped
 * @param swapData bytes swap data from primary transaction
 * @return boolean representing if a referral transaction was made
 */
function submit(address wallet, bytes memory walletData, address token, address
 ↪  recipient, uint256 amount, bytes memory swapData) external override payable
 ↪  onlyOwner() returns (bool) {
  //Perform user swap first
  ...

  //check if this is a referral transaction
  //if not exit execution
  if (token == address(0) || recipient == address(0) || amount == 0 ) {
    return false;
  }

  //if swapped token is in TEL, no swap is necessary
  //do simple transfer from and submit
  if (token == address(_telcoin)) {
    ...
  }

  //MATIC does not allow for approvals
  //ERC20s only
  if (token != MATIC) {
    IERC20(token).transferFrom(_safe, address(this), amount);
    IERC20(token).approve(_aggregator, amount);
  }

  //Perform secondary swap from fee token to TEL
  //do simple transfer from and submit
  (bool swapResult,) = _aggregator.call{value: msg.value}(swapData);
  require(swapResult, "FeeBuyback: swap transaction failed");
  _telcoin.approve(address(_referral), _telcoin.balanceOf(address(this)));
  require(_referral.increaseClaimableBy(recipient,
 ↪  _telcoin.balanceOf(address(this))), "FeeBuyback: balance was not adjusted");
  return true;
}
```

I.e. the funds in the native case aren't checked (can be zero, can be 100x of the fee needed), provided `amount` is just ignored.

Tool used

Manual Review

Recommendation

In order to maintain the uniform approach consider requiring that `amount` does exactly correspond to `msg.value`, when MATIC is used, for example:

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/fee-buyback/FeeBuyback.sol#L68-L73

```
    //MATIC does not allow for approvals
    //ERC20s only
    if (token != MATIC) {
      IERC20(token).transferFrom(_safe, address(this), amount);
      IERC20(token).approve(_aggregator, amount);
+   } else {
+     require(amount == msg.value, "FeeBuyback: wrong amount");
    }
```

Discussion

amshirif

https://github.com/telcoin/telcoin-staking/pull/10

Issue M-5: `slash()` can be frontrunned to avoid the penalty imposed on them

Source: https://github.com/sherlock-audit/2022-11-telcoin-judging/issues/45

Found by

cccz, yixxas, hickuphh3

Summary

I believe `slash()` is used to take funds away from a user when they misbehave. However, a malicious user can frontrun this operation or the `pause()` function and call `fullClaimAndExit()` to fully exit before the penalty can affect them.

Vulnerability Detail

Malicious users who have intentionally committed some offenses that would lead to getting slashed can listen to the mempool and frontrun the `slash()` or `pause()` function call by the protocol to protect all his assets before slashing can happen.

Impact

Slashing mechanism implemented can be bypassed by malicious user.

Code Snippet

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L403-L406

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L202-L207

Tool used

Manual Review

Recommendation

I implore the sponsors to explore alternatives to this slashing mechanism as they can be easily bypassed, especially so by sophisticated users who presumably are the ones who will be getting slashed.

Discussion

amshirif

https://github.com/telcoin/telcoin-staking/pull/21

Issue M-6: Slashing fails if claims revert

Source: https://github.com/sherlock-audit/2022-11-telcoin-judging/issues/5

Found by

hickuphh3

Summary

Slashing claims yields for the slashed account as part of the process. Should claims revert, slashing attempts will revert too.

Vulnerability Detail

Slashing calls the underlying `_claimAndExit()` function, which claims yield from all plugins. Should one or more claims fail, slashing will revert as well.

Impact

Failing claims brick the slashing functionality until the erroneous plugin(s) are removed. During which, the slashed user could have claimed his yield and exited.

Code Snippet

https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L403-L406 https://github.com/sherlock-audit/2022-11-telcoin/blob/main/contracts/StakingModule.sol#L356-L379

Tool used

Manual Review

Recommendation

Create another `slash()` method that skips claiming yields of the slashed account.

Discussion

amshirif

https://github.com/telcoin/telcoin-staking/pull/16