DeHacker

Code Security Assessment

ButterSwap

Feb 17th, 2024

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	ButterSwap
Platform	ETH
website	https://www.butterswap.io/swap
Type	Dex
Language	Solidity
Codebase	https://github.com/butternetwork/butter-router-contracts

Vulnerability Summary

Vulnerability Level	**Total**	**Pending**	**Declined**	**Acknowledged**	**Partially Resolved**	**Resolved**
Critical	0	0	0	0	0	0
Major	1	0	0	0	0	1
Medium	1	0	0	0	0	1
Minor	0	0	0	0	0	0
Informational	2	0	0	0	0	2
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
EFD	Router.sol	fcf569207b108662bb2159b95e6e90480d76b5feb02d35b77175e0f619634259
DFE	ButterRouterV2.sol	a917c47d8587859fd43e0d9628b0022ee994ad0a19e5ee9f1786e363849c7878
SGE	FeeReceiver.sol	c5b5413a4e306905e058b2237b0b74542b85d62a757cafa23a12e7ea1f322451
SOI	SwapAdapter.sol	d4244e67bb5185323e6bdfe1005a47ecb04d783d6f3b87f221bcce991c92f77b

Findings

ID	Issue	Severity	Status
Global-01	Centralization Related Risks	Major	Resolved
SGE-01	Reentrancy Events	Medium	Resolved
SGE-02	Calls Inside a Loop	Informational	Resolved
DFE-01	Uninitialized Local Variables	Informational	Resolved

MAJOR

Global 01 | Centralization Related Risks

Isuue	Severity	Location	Status
Centralization Related Risks	Major	Router.sol: 01	Resolved

Description

The owner of the SwapAdapter contract has the following permissions:

· function rescueFunds()

The owner of the FeeReceiver contract has the following permissions:

· function editConverter()

· function addReleaseToken()

The owner will have the ability to influence the operation results of the protocol.

Recommendation

This finding describes the level of decentralization of the project, and it is recommended to strengthen security and improve the degree of decentralization from the following aspects:

· It is recommended that privileged addresses use multi-signature wallet addresses.
· For modification operations that affect protocol operation stability and key business parameters, it is recommended to implement time locks.

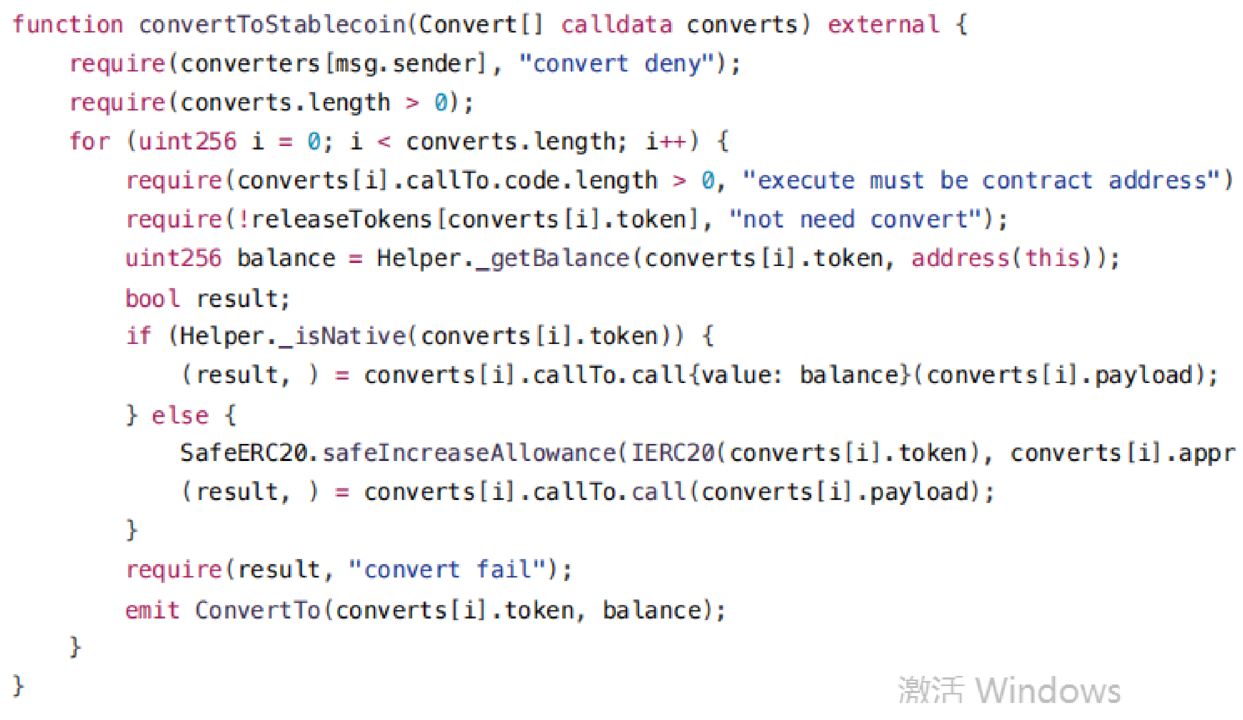
MEDIUM

SGE-01 | Reentrancy Events

Issue	Severity	Location	Status
Reentrancy Events	Medium	FeeReceiver.sol: 01	Resolved

Description

Line 46



```
function convertToStablecoin(Convert[] calldata converts) external {
    require(converters[msg.sender], "convert deny");
    require(converts.length > 0);
    for (uint256 i = 0; i < converts.length; i++) {
        require(converts[i].callTo.code.length > 0, "execute must be contract address")
        require(!releaseTokens[converts[i].token], "not need convert");
        uint256 balance = Helper._getBalance(converts[i].token, address(this));
        bool result;
        if (Helper._isNative(converts[i].token)) {
            (result, ) = converts[i].callTo.call{value: balance}(converts[i].payload);
        } else {
            SafeERC20.safeIncreaseAllowance(IERC20(converts[i].token), converts[i].appr
            (result, ) = converts[i].callTo.call(converts[i].payload);
        }
        require(result, "convert fail");
        emit ConvertTo(converts[i].token, balance);
    }
}
```

This may cause issues for offchain components that rely on the values of events.

Recommendation

Confirm that there are no logical issues, or eliminate the reentrant.

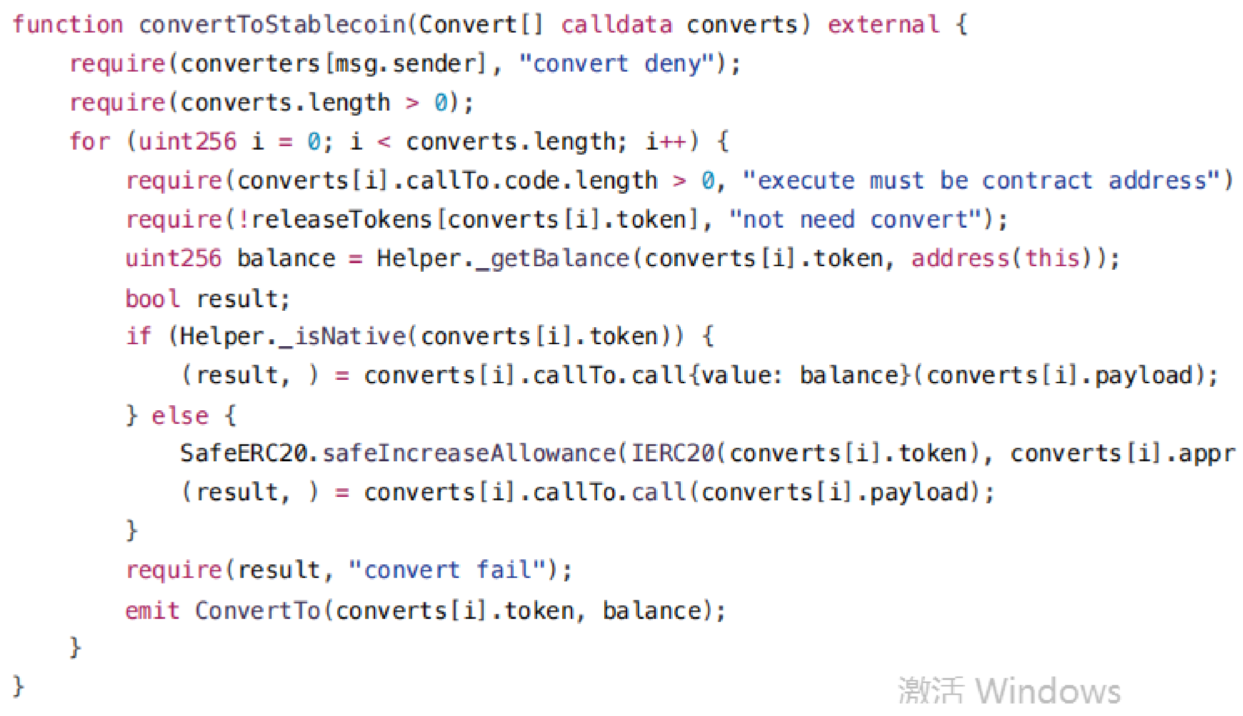
INFORMATIONAL

SGE-02 | Calls Inside a Loop

Issue	Severity	Location	Status
Calls Inside a Loop	Informational	FeeReceiver.sol: 02	Resolved

Description

Line 46



```
function convertToStablecoin(Convert[] calldata converts) external {
    require(converters[msg.sender], "convert deny");
    require(converts.length > 0);
    for (uint256 i = 0; i < converts.length; i++) {
        require(converts[i].callTo.code.length > 0, "execute must be contract address")
        require(!releaseTokens[converts[i].token], "not need convert");
        uint256 balance = Helper._getBalance(converts[i].token, address(this));
        bool result;
        if (Helper._isNative(converts[i].token)) {
            (result, ) = converts[i].callTo.call{value: balance}(converts[i].payload);
        } else {
            SafeERC20.safeIncreaseAllowance(IERC20(converts[i].token), converts[i].appr
            (result, ) = converts[i].callTo.call(converts[i].payload);
        }
        require(result, "convert fail");
        emit ConvertTo(converts[i].token, balance);
    }
}
```

Calls inside a loop might lead to a DoS attack.

Recommendation

External calls can fail accidentally or deliberately. To minimize the damage caused by such failures, it is often better to isolate each external call into its own transaction that can be initiated by the recipient of the call.

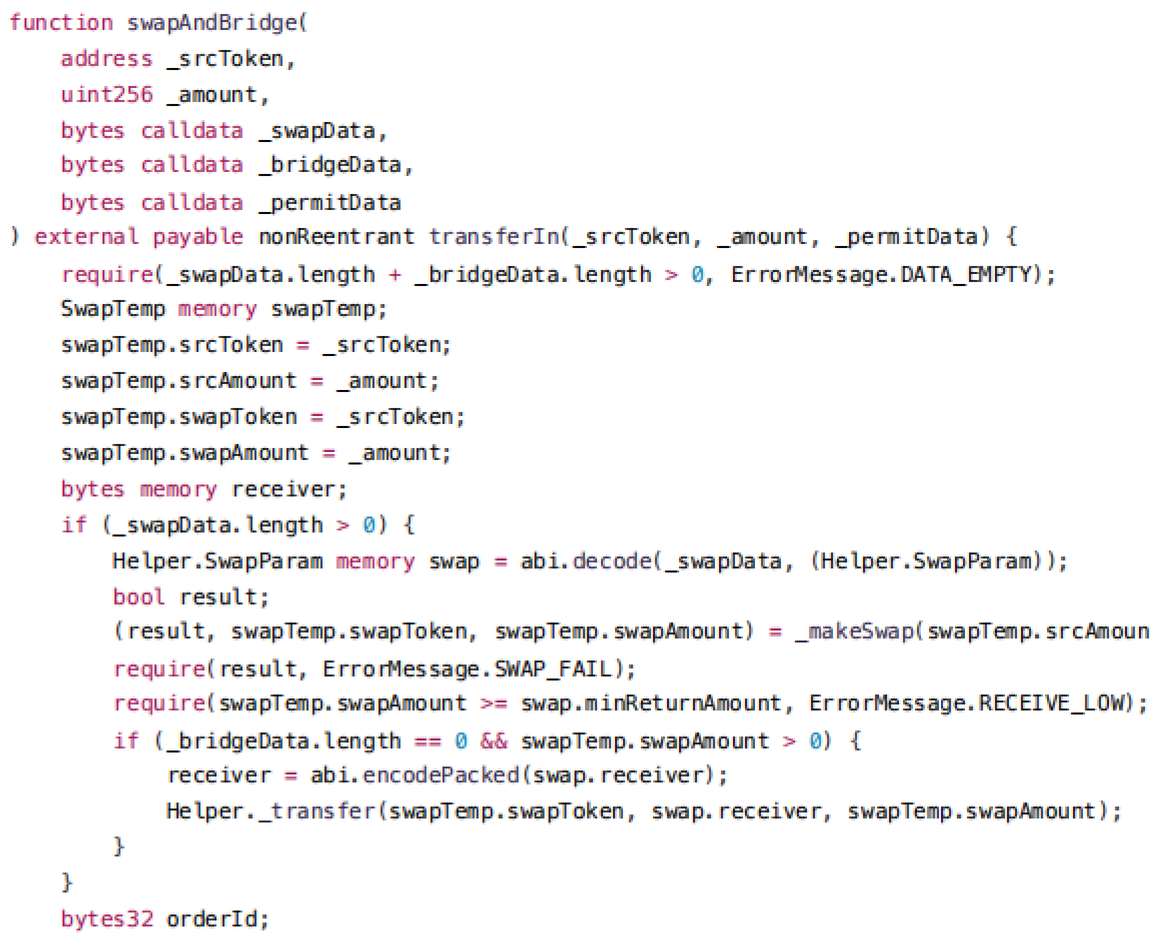
INFORMATIONAL

DEF-01 | Uninitialized Local Variables

Issue	Severity	Location	Status
Uninitialized Local Variables	Informational	ButterRouterV2.sol: 01	Resolved

Description

Line 61



```
function swapAndBridge(
    address _srcToken,
    uint256 _amount,
    bytes calldata _swapData,
    bytes calldata _bridgeData,
    bytes calldata _permitData
) external payable nonReentrant transferIn(_srcToken, _amount, _permitData) {
    require(_swapData.length + _bridgeData.length > 0, ErrorMessage.DATA_EMPTY);
    SwapTemp memory swapTemp;
    swapTemp.srcToken = _srcToken;
    swapTemp.srcAmount = _amount;
    swapTemp.swapToken = _srcToken;
    swapTemp.swapAmount = _amount;
    bytes memory receiver;
    if (_swapData.length > 0) {
        Helper.SwapParam memory swap = abi.decode(_swapData, (Helper.SwapParam));
        bool result;
        (result, swapTemp.swapToken, swapTemp.swapAmount) = _makeSwap(swapTemp.srcAmoun
        require(result, ErrorMessage.SWAP_FAIL);
        require(swapTemp.swapAmount >= swap.minReturnAmount, ErrorMessage.RECEIVE_LOW);
        if (_bridgeData.length == 0 && swapTemp.swapAmount > 0) {
            receiver = abi.encodePacked(swap.receiver);
            Helper._transfer(swapTemp.swapToken, swap.receiver, swapTemp.swapAmount);
        }
    }
    bytes32 orderId;
    if (_bridgeData.length > 0) {
        BridgeParam memory bridge = abi.decode(_bridgeData, (BridgeParam));
        swapTemp.toChain = bridge.toChain;
        receiver = bridge.receiver;
        orderId = _doBridge(msg.sender, swapTemp.swapToken, swapTemp.swapAmount, bridge
    }
    emit SwapAndBridge(
        orderId,
        msg.sender,
        swapTemp.srcToken,
        swapTemp.swapToken,
        swapTemp.srcAmount,
        swapTemp.swapAmount,
        block.chainid,
        swapTemp.toChain,
        receiver
      );
  }
```

Uninitialized local variables.

Recommendation

Initialize all the variables.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum	Cosmos
Eos	Substrate

TECH STACK

Python	Solidity
Rust	c++

CONTACTS

https://dehacker.io

https://twitter.com/dehackerio

https://github.com/dehacker/audits_public

https://t.me/dehackerio

https://blog.dehacker.io/

DeHacker

Feb 2024