Starksport
Audit Report

Tue Oct 31 2023

contact@scalebit.xyz https://twitter.com/scalebit_

ScaleBit

Starksport Audit Report

1 Executive Summary

1.1 Project Information

Description	An all-in-one Incubation Hub with a full stack DeFi platform including IDO Launchpad, NFT Marketplace, and Exchange
Type	Dex
Auditors	ScaleBit
Timeline	Fri Oct 13 2023 - Tue Oct 31 2023
Languages	Cairo
Platform	Starknet
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/starksport-project/SSContracts
Commits	19a0c4fed4f358104de3166ec1dc9abeb6b82f99

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
ERC7SSV2	ERC721_SS_V2.cairo	d96cf42a6c78c09b69013f94ccdf07ca5ad53021
ERC7SS	ERC721_SS.cairo	275ae36f9cedde1de64604a0a664f2d4b7c18f39
ERC2	ERC20.cairo	6ee64c8ae2fdba551980fe1b3d12996e26c142af
FAC	DEX/Factory.cairo	733bac480f5ec106671cd4fc531e2ab3ecd19541
FPR	DEX/FactoryProxy.cairo	b555e10b5cf306cdbf666852affd821797f131d2
PPR	DEX/PairProxy.cairo	720cc991a121ce02b9c90ffe8accdc367aba0f9a
PAI	DEX/Pair.cairo	c959ae17682dfd824a3493c7080187d20d8c27ff
RPR	DEX/RouterProxy.cairo	24b1b9291aa02d70d5bbf18605ff7c258ff28f6d
ROU	DEX/Router.cairo	03f4d0d2fec3ccb5507381edb67571c8a8c9b087
ERC2A	ERC20_Airdrop.cairo	b8fa533a9ea93205f0e5b71b20543fd40c97b3a8

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	6	0	6
Informational	0	0	0
Minor	3	0	3
Medium	1	0	1
Major	2	0	2
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Starksport to identify any potential issues and vulnerabilities in the source code of the Starksport smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 6 issues of varying severity, listed below.

ID	Title	Severity	Status
ERC-1	Unused Arguments	Minor	Acknowledged
FAC-1	Dead Store	Minor	Acknowledged
FAC-2	Unused Imports	Minor	Acknowledged
FPR-1	Single-step Admin Transfer Can be Dangerous	Major	Acknowledged
FPR-2	Centralization Risk	Major	Acknowledged
PAI-1	Use `sender.starksportswap_call()`	Medium	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the Starksport Smart Contract:

Admin

- Admin can upgrade the implementation contracts for the Factory, Pair, and Router contracts by calling the `upgrade` function.
- Admini can use the `claim_back` function to return a specific amount of tokens to the caller.

Fee-setter

- The Fee-Setter can invoke the `set_fee_to` function to designate the recipient of fees.
- The Fee-Setter can invoke the `set_fee_to_setter` function, enabling the transfer of the authority to designate the fee recipient.

User

- Users can call the `add_liquidity` function to add liquidity.
- Users can use the `remove_liquidity` function to remove liquidity.
- Users can invoke the `swap_exact_tokens_for_tokens` function to acquire the maximum output tokens based on the specified input tokens.
- Users can use the `swap_tokens_for_exact_tokens` function to exchange with the minimum input tokens required for the desired output tokens.
- Users can create a new pair pool by calling the `create_pair` function.
- Users can update the reserves to the current balance by calling the `sync` function.
- Users can withdraw excess amounts by invoking the `skim` function.
- Users who are whitelisted can utilize the `whitelist_claim` function to receive tokens.

4 Findings

ERC-1 Unused Arguments

Severity: Minor

Status: Acknowledged

Code Location:

ERC721_SS_V2.cairo#128-136;

ERC721_SS.cairo#98-106

Descriptions:

The `token_id` parameter is declared in the function's parameters, but it is not utilized within the function's body, rendering it redundant.

```
func tokenURI{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, range_check_ptr}(
    token_id: Uint256
) -> (token_uri_len: felt, token_uri: felt*) {
    // let (token_uri_len, token_uri) = ERC721_Metadata_tokenURI(token_id);
    // return (token_uri_len=token_uri_len, token_uri=token_uri);
    let (uri_len: felt, uri: felt*) = getContractURI();
    return (token_uri_len=uri_len, token_uri=uri);
}
```

Suggestion:

It is recommended to remove the unused `token_id` parameter from the function's parameters.

FAC-1 Dead Store

Severity: Minor

Status: Acknowledged

Code Location:

DEX/Factory.cairo#194

Descriptions:

Identified a situation in the code where variables are assigned values but are not utilized or referenced before a return statement. It's important to review and potentially remove these variables to improve code clarity and efficiency.

```
let (contract_address: felt) = get_contract_address();
```

The same issue is also found in other parts of the code, listed below.

```
ERC20_Airdrop.cairo:117:10
ERC20_Airdrop.cairo:125:11
ERC721_SS.cairo:259:10
ERC721_SS_V2.cairo:313:10
Factory.cairo:122:10
Factory.cairo:194:10
Router.cairo:359:14
Router.cairo:566:10
Router.cairo:617:10
```

Suggestion:

It is recommended to address variables that are assigned values but remain unused prior to a return statement. Optimizing the code by removing these unused variables will lead to cleaner and more efficient code.

FAC-2 Unused Imports

Severity: Minor

Status: Acknowledged

Code Location:

DEX/Factory.cairo#13

Descriptions:

The following imported modules are not referenced anywhere in the code, and they can be safely removed to declutter the codebase.

```
from starkware.cairo.common.math_cmp import is_le
```

The same issue is also found in other parts of the code, listed below.

```
ERC20_Airdrop.cairo:10:5
ERC20_Airdrop.cairo:14:5
ERC20_Airdrop.cairo:15:5
ERC20_Airdrop.cairo:17:42
ERC20_Airdrop.cairo:18:41
ERC20_Airdrop.cairo:18:58
ERC20_Airdrop.cairo:19:43
ERC20_Airdrop.cairo:7:5
ERC20_Airdrop.cairo:8:5
ERC20_Airdrop.cairo:9:5
ERC721_SS.cairo:14:5
ERC721_SS.cairo:15:5
ERC721_SS.cairo:18:58
ERC721_SS.cairo:19:43
ERC721_SS.cairo:31:5
ERC721_SS.cairo:32:5
ERC721_SS.cairo:8:5
ERC721_SS.cairo:9:5
ERC721_SS_V2.cairo:14:5
ERC721_SS_V2.cairo:15:5
ERC721_SS_V2.cairo:18:58
ERC721_SS_V2.cairo:19:43
ERC721_SS_V2.cairo:31:5
ERC721_SS_V2.cairo:32:5
```

```
ERC721_SS_V2.cairo:8:5
ERC721_SS_V2.cairo:9:5
Factory.cairo:13:45
Factory.cairo:9:44
Pair.cairo:11:47
Pair.cairo:19:5
Pair.cairo:20:5
Pair.cairo:29:5
Pair.cairo:34:42
```

Suggestion:

It is recommended to remove unused import statements to keep the codebase clean and improve maintainability.

FPR-1 Single-step Admin Transfer Can be Dangerous

Severity: Major

Status: Acknowledged

Code Location:

DEX/FactoryProxy.cairo#83-90;

DEX/PairProxy.cairo#83-89;

DEX/RouterProxy.cairo#83-89

Descriptions:

Single-step `admin` transfer means that if a wrong address was passed when transferring admin it can mean that the role is lost forever. If the admin permissions are given to the wrong address within the function `set_admin()`, the bad actor can update the implementation contract, injecting malicious code that could compromise user funds.

```
@external
func set_admin{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, range_check_ptr}(
        new_admin: felt
) {
    Proxy.assert_only_admin();
    Proxy._set_admin(new_admin);
    return ();
}
```

Suggestion:

It is recommended to use a two-step ownership transfer pattern, meaning ownership transfer gets to a "pending" state and the new owner should claim his new rights, otherwise, the old owner still has control of the contract.

FPR-2 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

DEX/FactoryProxy.cairo#75-81;

DEX/PairProxy.cairo#74-80;

DEX/RouterProxy.cairo#74-80

Descriptions:

The function `upgrade()` allows only the admin to change the implementation contract, which poses a centralization risk. If the new implementation contract chosen by the admin contains malicious code, it could potentially access and siphon user funds directly.

```
@external
func upgrade{syscall_ptr: felt*, pedersen_ptr: HashBuiltin*, range_check_ptr}(
    new_implementation: felt
) {
    Proxy.assert_only_admin();
    Proxy._set_implementation_hash(new_implementation);
    return ();
}
```

Suggestion:

It is recommended to take some measures to mitigate centralization risk.

PAI-1 Use `sender.starksportswap_call()`

Severity: Medium

Status: Acknowledged

Code Location:

DEX/Pair.cairo#633

Descriptions:

The `pair.swap()` function supports flash loan operations. Currently, the protocol calls the `to` address's callback function directly. This design has a potential security vulnerability. In this setup, if a user implements their own `starksportswap_call()` function and it happens to be insecure, it could introduce significant security risks. An attacker could exploit this situation.

```
let data_len_greater_than_zero = is_le(1, data_len);
  if (data_len_greater_than_zero == 1) {
      IStarkSportSwapCallee.starksportswap_call(
          contract_address=to,
          sender=caller_address,
          amount0Out=amount0Out,
          amount1Out=amount1Out,
          data_len=data_len,
          data=data,
      );
      tempvar syscall_ptr = syscall_ptr;
      tempvar pedersen_ptr = pedersen_ptr;
      tempvar range_check_ptr = range_check_ptr;
  }
```

An analogous incident occurred with the Primitive project, specifically with the Uniswap V2 integration (in the uniswapV2Call function). The problem there was that the Primitive Connector code didn't verify the initiator of the flash-swap operation ,it merely checked whether the callback came from Uniswap.

To mitigate this security concern, a safer approach would be to follow the Uniswap V3 model, where the protocol calls the msg.sender's callback function, which inherently verifies the initiator of the flash-swap operation. This would help prevent unauthorized or

potentially malicious calls to the `starksportswap_call()` function, enhancing the overall security of the protocol.

Suggestion:

It is recommended to change `to.starksportswap_call()` to `sender.starksportswap_call()`.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include 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 own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.