QuillAudits

AUDIT REPORT

April , 2025

For

GROWFITTER

Table of Content

Table of Content 02

Executive Summary 03

Number of Issues per Severity 05

Checked Vulnerabilities 06

Techniques & Methods 07

Types of Severity 09

Types of Issues 10

Medium Severity Issues 11

1. Missing Address Validation in Admin 11

2. Missing Direct Transfer Opt-in Check in NFT Minting 12

Low Severity Issues 13

1. The minimum mint Amount validation check missing in Minting. 13

Informational Severity Issues 14

1. Unused error code. 14

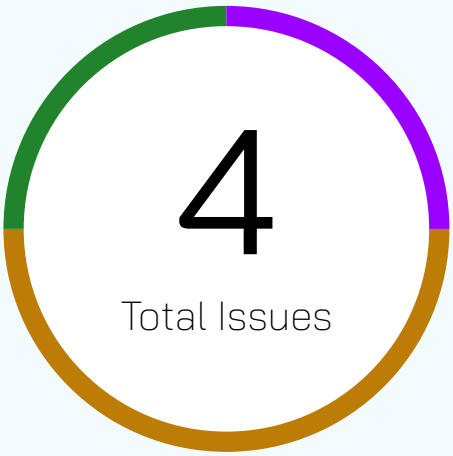
Closing Summary & Disclaimer 15

Executive Summary

Project name	Growfitter
Project URL	https://www.growfitter.com/
Overview	The LyncCards module is a Move language smart contract deployed on the Aptos blockchain that implements a digital collectible series called "Growfitter Squid Game NFT Card Collection." It enables the creation and distribution of 15 uniquely themed NFT cards inspired by the popular Squid Game series, each with detailed descriptions and metadata hosted on IPFS. The contract utilizes a resource account architecture with ED25519 cryptographic signature verification to secure the minting process, ensuring only authorized transactions can create new tokens. Administrative functions allow for management control and fund withdrawal, while the property version system differentiates between card types.
Audit Scope	The Scope of the Audit is to analyse the Code Quality ,Security and Correctness of Growfitter Smart Contract
Contracts in Scope	https://github.com/LYNC-WORLD/Aptos-Growfitter-Contracts Branch: Main Contract: lync_cards.move
Commit Hash	c9ad4ded91f1676ea4570d88b4255c836320b3c7
Language	Move
Blockchain	Aptos
Method	Manual Analysis, Functional Testing.

Review 1	2nd April 2025 to 5th April 2025
Updated Code Received	7th April 2025
Review 2	8 April 2025 - 9th April 2025
Fixed In	https://github.com/LYNC-WORLD/Aptos-Growfitter-Contracts/tree/ssw/audit-fix

Number of Issues per Severity



High	0 (0.00%)
Medium	2 (50.00%)
Low	1 (25.00%)
Informational	1 (25.00%)

Severity

Issues	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	0	**2**	**1**	**1**
Acknowledged	0	0	0	0
Partially Resolved	0	0	0	0

Checked Vulnerabilities

- ☑ Transaction-ordering dependence
- ☑ Timestamp dependence
- ☑ Integer overflow/underflow by bit operations
- ☑ Number of rounding errors
- ☑ Denial of service / logical oversights
- ☑ Access control
- ☑ Centralization of power
- ☑ Business logic contradicting the specification
- ☑ Code clones
- ☑ functionality duplication
- ☑ Gas usage
- ☑ Arbitrary token minting
- ☑ Unchecked CALL Return Values
- ☑ The flow of capability
- ☑ Witness Type

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code
- Use of best practices
- Code documentation and comments, match logic and expected behavior
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper
- Efficient use of gas
- Code is safe from re-entrancy and other vulnerabilities

The following techniques, methods, and tools were used to review all the smart contracts:

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Techniques and Methods

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational Issues

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

Security vulnerabilities identified that must be resolved and are currently unresolved.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/ impact of the security issue, but are not completely resolved.

Medium Severity Issues

Missing Address Validation in Admin

Resolved

Path

lync_cards.move

Function

update_admin_address()

Description

The update_admin_address function lacks validation check for the new admin address existence. This could lead to irreversible loss of admin control.

```
// update admin address
public entry fun update_admin_address(
    caller: &signer,
    new_admin_address: address
) acquires NFTData {
    let caller_address = signer::address_of(caller);
    let module_data = borrow_global_mut<NFTData>(@publisher);
    assert!(
        caller_address == module_data.admin_address,
        error::permission_denied(ENOT_AUTHORIZED)
    );
    module_data.admin_address = new_admin_address;
}
```

Recommendation

Implement a two-step process like First call to propose new admin ,Second call by new admin to confirm.

Missing Direct Transfer Opt-in Check in NFT Minting

Resolved

Path

lync_cards.move

Function

mint_nft()

Description

The mint_nft function using token::direct_transfer function but not checking if the receiver is opt in for the direct transfer.

```
token::direct_transfer(
                &resource_signer,
                receiver,
                tid,
                amount
            );
```

Recommendation

use get_direct_transfer function from the token.move to check this before calling token::direct_transfer.

Low Severity Issues

The minimum mint Amount validation check missing in Minting.

Resolved

Path

lync_cards.move

Function

mint_nft()

Description

The mint_nft function calls token::mint_token without validating the amount parameter. If the amount is zero, the transaction will revert in the token contract, wasting gas and creating unnecessary calls.

Recommendation

Add a check in the mint_nft function :

```
assert!(amount > 0, error::invalid_argument(EINVALID_AMOUNT));
```

Informational Severity Issues

Unused error code.

Resolved

Description

The ENOT_OPT_IN error code is defined but never used.

Recommendation

Remove the code / add a check for the error.

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of Growfitter. We performed our audit according to the procedure described above.

Some issues of Medium, Low and Informational severity were found. Some suggestions, gas optimizations and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

At QuillAudits, we have spent years helping projects strengthen their smart contract security. However, security is not a one-time event—threats evolve, and so do attack vectors. Our audit provides a security assessment based on the best industry practices at the time of review, identifying known vulnerabilities in the received smart contract source code.

This report does not serve as a security guarantee, investment advice, or an endorsement of any platform. It reflects our findings based on the provided code at the time of analysis and may no longer be relevant after any modifications. The presence of an audit does not imply that the contract is free of vulnerabilities or fully secure.

While we have conducted a thorough review, security is an ongoing process. We strongly recommend multiple independent audits, continuous monitoring, and a public bug bounty program to enhance resilience against emerging threats.

Stay proactive. Stay secure.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem

QuillAudits

7+ Years of Expertise	**1M+** Lines of Code Audited
$30B+ Secured in Digital Assets	**1400+** Projects Secured

Follow Our Journey

AUDIT REPORT

April , 2025

For

GROWFITTER

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com audits@quillaudits.com