HALBORN

Yieldly.Finance - Multi Staking

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 20th, 2021 - November 9th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/20/2021	Gabi Urrutia
0.2	Document Edits	11/01/2021	Gokberk Gulgun
0.3	Final Document	11/09/2021	Gabi Urrutia
1.0	Remediation Plan	12/06/2021	Gokberk Gulgun
1.1	Remediation Plan Review	12/13/2021	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Yieldly.Finance is a lossless lottery staking platform where users can stake their Algorand coins to receive an entry. With the staking lottery, YLDY ASA token holders will benefit from a new feature to stake their assets. Each token holder also earns ASA tokens (YLDY) as reward for their staking contributions.

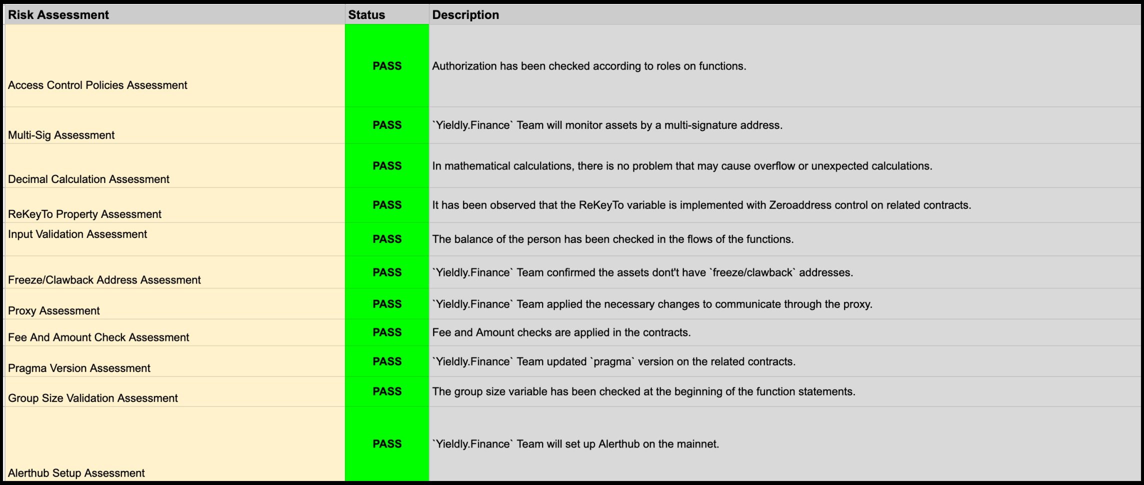
Yieldly.Finance engaged Halborn to conduct a security assessment on their Multi Smart contract beginning on October 20th, 2021 and ending November 9th, 2021. The security assessment was scoped to the Algorand lottery contracts and an audit of the security risk and implications regarding the changes introduced by the development team at Yieldly.Finance prior to its production release shortly following the assessment's deadline.

Though this security audit's outcome is satisfactory, only the most essential aspects were tested and verified to achieve objectives and deliverables set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure smart-contract development.

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned three full time security engineers to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

Risk Assessment Sheet



Risk Assessment	Status	Description
Access Control Policies Assessment	PASS	Authorization has been checked according to roles on functions.
Multi-Sig Assessment	PASS	`Yieldly.Finance` Team will monitor assets by a multi-signature address.
Decimal Calculation Assessment	PASS	In mathematical calculations, there is no problem that may cause overflow or unexpected calculations.
ReKeyTo Property Assessment	PASS	It has been observed that the ReKeyTo variable is implemented with Zeroaddress control on related contracts.
Input Validation Assessment	PASS	The balance of the person has been checked in the flows of the functions.
Freeze/Clawback Address Assessment	PASS	`Yieldly.Finance` Team confirmed the assets dont't have `freeze/clawback` addresses.
Proxy Assessment	PASS	`Yieldly.Finance` Team applied the necessary changes to communicate through the proxy.
Fee And Amount Check Assessment	PASS	Fee and Amount checks are applied in the contracts.
Pragma Version Assessment	PASS	`Yieldly.Finance` Team updated `pragma` version on the related contracts.
Group Size Validation Assessment	PASS	The group size variable has been checked at the beginning of the function statements.
Alerthub Setup Assessment	PASS	`Yieldly.Finance` Team will set up Alerthub on the mainnet.

The purpose of this audit to achieve the following:

- Ensure that smart contract functions are intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified few security risks that were acknowledged and addressed by t'Yieldly.Financet'.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit.While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart Contract manual code read and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions(buildr)
- Manual Assessment of use and safety for the critical Algorand variables and functions in scope to identify any arithmetic related vulnerability classes.
- Smart Contract Dynamic Analysis And Flow Testing

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident, and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. It's quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

EXECUTIVE OVERVIEW

1.4 SCOPE

Code related to Yieldly Multi Staking Contract

Specific commit of contract:
4aefdf30863f0252b96cb39dd64afdf18318b374

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	0	4

LIKELIHOOD

IMPACT

🟨	🟨	🟥	🟥	🟪
🟩	🟨	🟨	🟥	🟥
🟩	🟩	🟨 (HAL-01)	🟨	🟥
⬜	🟩	🟩	🟨	🟨
⬜ (HAL-02) (HAL-03) (HAL-04) (HAL-05)	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - MISSING INTEGER UNDERFLOW PROTECTION	Medium	SOLVED
HAL02 - LACK OF MAXIMUM FEE BOUND DEFINITION	Informational	NOT APPLICABLE
HAL03 - LACK OF MULTISIG PROGRAM	Informational	ACKNOWLEDGED
HAL04 - MISSING PROXY ASSET DEFINITION ON THE FUNCTIONS	Informational	NOT APPLICABLE
HAL05 - MISSING FREEZE/REVOKE ASSETS DEFINITION	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) MISSING INTEGER UNDERFLOW PROTECTION - MEDIUM

Description:

In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is outside the range that can be represented with a given number of bits, either larger than the maximum or lower than the minimum value.

Side Note:

Listing 1

```
b-  A minus B, where A and B are byte-arrays interpreted as big-
    endian unsigned integers. Fail on underflow.
```

Code Location:

Listing 2

```
resumeRewardCalc:
// Calculate the amount of rewards unlocked
byte "End_Date"          // "End_Date"
app_global_get           // bytex
byte "Start_Date"        // bytex "Start_Date"
app_global_get           // bytex bytex
b-                       // bytex
store 10                 // null
load 10                  // bytex
byte "End_Date"          // bytex "End_Date"
app_global_get           // bytex intx
load 13                  // bytex intx intx
b-                       // bytex intx  TODO {HANDLE UNDERFLOW}
b-                       // bytex
byte "Rewards_Locked"    // bytex "R
```

Listing 3

```
resumeRewardCalc:
// Calculate the amount of rewards unlocked
byte "End_Date"             // "End_Date"
app_global_get              // bytex
byte "Start_Date"           // bytex "Start_Date"
app_global_get              // bytex bytex
b-                          // bytex
store 10                    // null
load 10                     // bytex
byte "End_Date"             // bytex "End_Date"
app_global_get              // bytex intx
load 13                     // bytex intx intx
b-                          // bytex intx  TODO {HANDLE UNDERFLOW}
b-                          // bytex
byte "Rewards_Locked"       // bytex "Rewards_Locked"
app_global_get              // bytex intx
b*                          // bytex
load 10                     // bytex bytex
b/                          // bytex
byte "Rewards_Unlocked"     // intx "Rewards_Unlocked"
app_global_get              // intx intx
b-                          // intx
store 2                     // null
```

Recommendation:

The variable should be checked with pre-condition.

Remediation Plan:

SOLVED: Yieldly.Finance implemented necessary checks.

3.2 (HAL-02) LACK OF MAXIMUM FEE BOUND DEFINITION - INFORMATIONAL

Description:

The fee does not have an upper/lower limit, which may make liquidity provider make no profit. The fee is defined as constant, therefore fee could not set by an admin on the contracts.

Code Location:

Listing 4: Fee is Defined Constant

```
// Makes sure the fee from the first txn is at least 2000 (min amount)
gtxn 0 Fee                // Fee
int 2000                  // Fee intx
>=                        // 1||0
assert                    // null (if 0 then Failed)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider to defined setter function fee function. However, the function should have upper/lower limit on the fee setter function.

Remediation Plan:

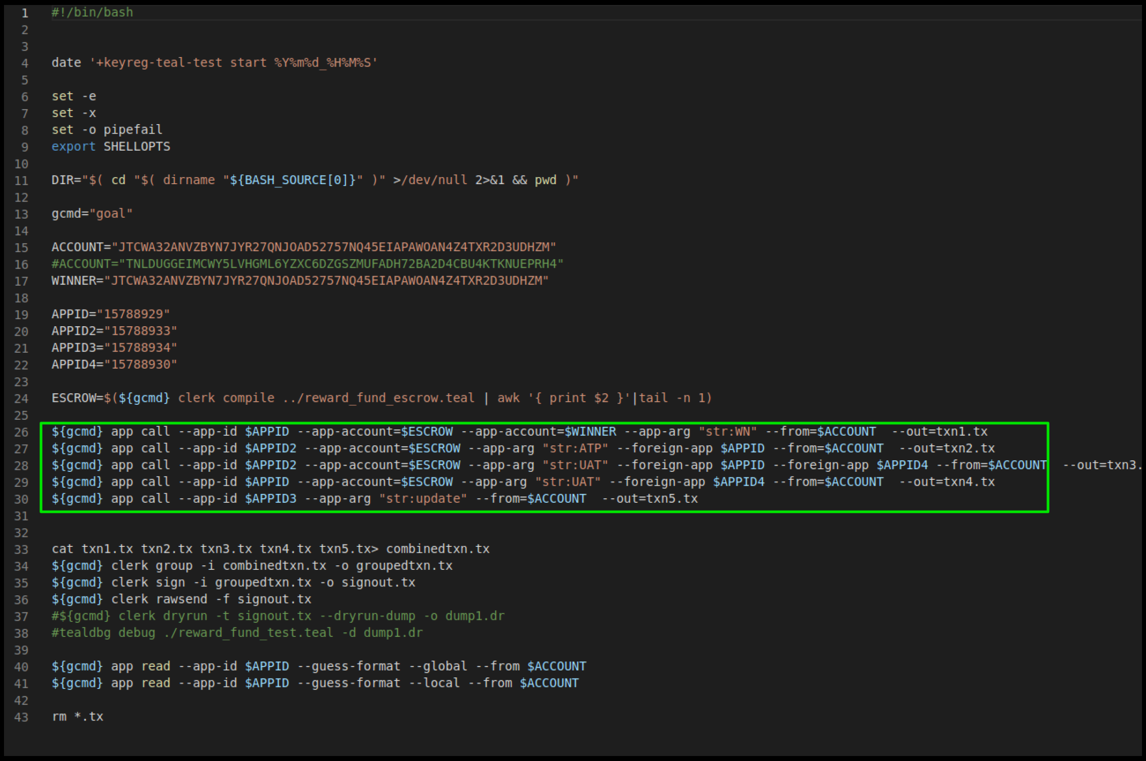
NOT APPLICABLE: The Yieldly.Finance claims that the implementation should be having minimum **2000** fee. The documentation has been reviewed and the implementation has been confirmed.

3.3 (HAL-03) LACK OF MULTISIG PROGRAM - INFORMATIONAL

Description:

The principal benefit of multisig is that it creates added redundancy in key management. While single signature addresses require only a single key for transactions, multisignature addresses require multiple keys. To protect against malicious admin, it may be necessary to use a multi signature. By using this mechanism, a malicious admin actions could be prevented.

Code Location:



```
#!/bin/bash


date '+keyreg-teal-test start %Y%m%d_%H%M%S'

set -e
set -x
set -o pipefail
export SHELLOPTS

DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

gcmd="goal"

ACCOUNT="JTCWA32ANVZBYN7JYR27QNJOAD52757NQ45EIAPAWOAN4Z4TXR2D3UDHZM"
#ACCOUNT="TNLDUGGEIMCWY5LVHGML6YZXC6DZGSZMUFADH72BA2D4CBU4KTKNUEPRH4"
WINNER="JTCWA32ANVZBYN7JYR27QNJOAD52757NQ45EIAPAWOAN4Z4TXR2D3UDHZM"

APPID="15788929"
APPID2="15788933"
APPID3="15788934"
APPID4="15788930"

ESCROW=$(${gcmd} clerk compile ../reward_fund_escrow.teal | awk '{ print $2 }'|tail -n 1)

${gcmd} app call --app-id $APPID --app-account=$ESCROW --app-account=$WINNER --app-arg "str:WN" --from=$ACCOUNT  --out=txn1.tx
${gcmd} app call --app-id $APPID2 --app-account=$ESCROW --app-arg "str:ATP" --foreign-app $APPID --from=$ACCOUNT  --out=txn2.tx
${gcmd} app call --app-id $APPID2 --app-account=$ESCROW --app-arg "str:UAT" --foreign-app $APPID --foreign-app $APPID4 --from=$ACCOUNT  --out=txn3.
${gcmd} app call --app-id $APPID --app-account=$ESCROW --app-arg "str:UAT" --foreign-app $APPID4 --from=$ACCOUNT  --out=txn4.tx
${gcmd} app call --app-id $APPID3 --app-arg "str:update" --from=$ACCOUNT  --out=txn5.tx


cat txn1.tx txn2.tx txn3.tx txn4.tx txn5.tx> combinedtxn.tx
${gcmd} clerk group -i combinedtxn.tx -o groupedtxn.tx
${gcmd} clerk sign -i groupedtxn.tx -o signout.tx
${gcmd} clerk rawsend -f signout.tx
#${gcmd} clerk dryrun -t signout.tx --dryrun-dump -o dump1.dr
#tealdbg debug ./reward_fund_test.teal -d dump1.dr

${gcmd} app read --app-id $APPID --guess-format --global --from $ACCOUNT
${gcmd} app read --app-id $APPID --guess-format --local --from $ACCOUNT

rm *.tx
```

Example Definition:

Listing 5: Multisig Implementation

```
goal account multisig new -T 2 account1 account2 account3 -d ~/node/data
goal clerk multisig signprogram -p /tmp/*.teal -a account1 -A account2 -o /tmp/simple.lsig -d ~/node/data
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

In the contract, The multi-signature should be implemented over a creator account.

Remediation Plan:

ACKNOWLEDGED: Yieldly.Finance consider to use multi-signature on the main net deployment.

3.4 (HAL-04) MISSING PROXY ASSET DEFINITION ON THE FUNCTIONS - INFORMATIONAL

Description:

In the Yieldly.Finance workflow, Escrow connection is made with a proxy contract. According to documentation, Escrow only allows transactions tied with proxy. But, in some functions, transactions don’t go through the Proxy asset.

Code Location:

Listing 6: winnerProgram Function (Lines 1)

```
let txn = await configs.winnerProgram(
        account2,
        escrowAddress,
        algoAppId,
        asaAppId,
        trackerAppId,
        winner,
        rateAppId
    );
```

Listing 7: assetOptoutApplication Function (Lines 1)

```
let txn1 = await configs.assetOptoutApplication(
        account1,
        escrowAddress,
        optingAppId,
        assetId
    );
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to construct transactions through a proxy which is interacting with escrow.

Remediation Plan:

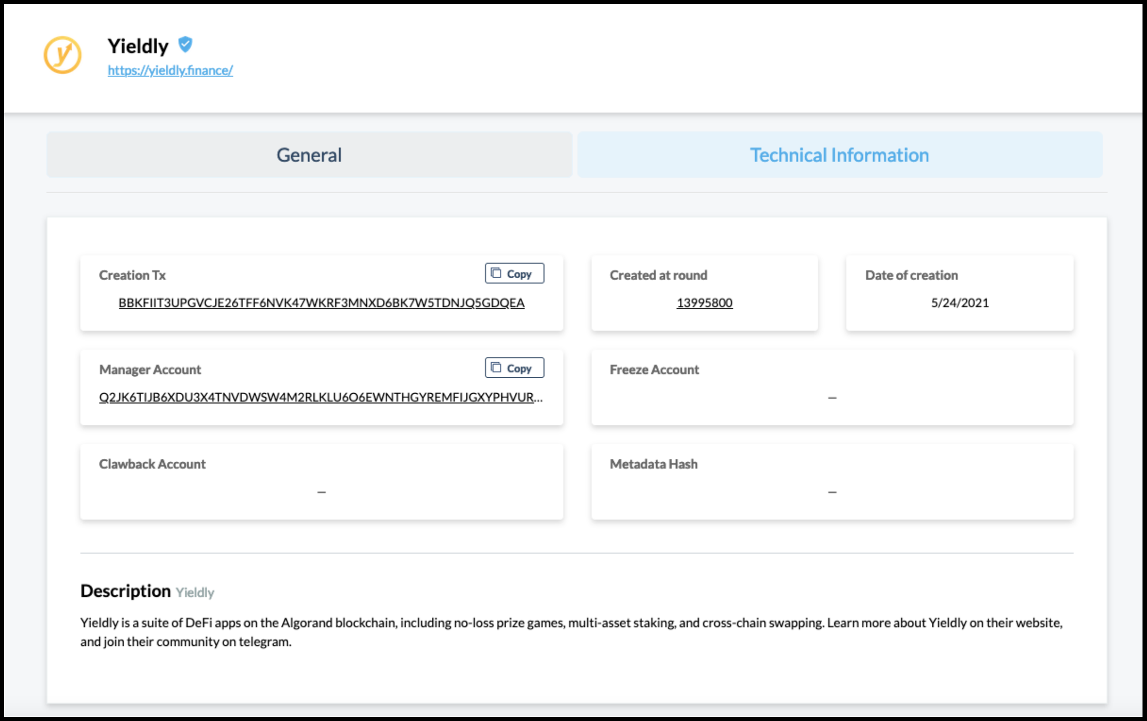
NOT APPLICABLE: Yieldly.Finance does not need to use proxy for the escrow asset after program version (5).

3.5 (HAL-05) MISSING FREEZE/REVOKE ASSETS DEFINITION - INFORMATIONAL

Description:

When an asset is created, the contract can provide a freeze address and a defaultfrozen state. If the defaultfrozen state is set to true the corresponding freeze address must issue unfreeze transactions, one per account, to allow trading of the asset to and from that account. This may be useful in situations that require holders of the asset to pass certain checks prior to ownership. (KYC/AML) The clawback address, if specified, is able to revoke the asset from any account and place them in any other account that has previously opted-in. This may be useful in situations where a holder of the asset breaches some set of terms that you established for that asset. You could issue a freeze transaction to investigate, and if you determine that they can no longer own the asset, you could revoke the assets.

Asset Explorer:



Yieldly
https://yieldly.finance/

Creation Tx
BBKFIIT3UPGVCJE26TFF6NVK47WKRF3MNXD6BK7W5TDNJQ5GDQEA

Created at round
13995800

Date of creation
5/24/2021

Manager Account
Q2JK6TIJB6XDU3X4TNVDWSW4M2RLKLU6O6EWNTHGYREMFIJGXYPHVUR...

Freeze Account
–

Clawback Account
–

Metadata Hash
–

Description Yieldly

Yieldly is a suite of DeFi apps on the Algorand blockchain, including no-loss prize games, multi-asset staking, and cross-chain swapping. Learn more about Yieldly on their website, and join their community on telegram.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

According to workflow, the application should activate freeze and revoke assets. If the application would rather ensure to asset holders that the application will never have the ability to revoke or freeze assets, set the clawback/freeze address to null.

Remediation Plan:

ACKNOWLEDGED: Yieldly.Finance does not need to use revoke or freeze feature on the assets. The Revoke and Freeze addresses are disabled.

THANK YOU FOR CHOOSING

HALBORN