// HALBORN

Yieldly.Finance - Algorand Compound Staking

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **November 9th, 2021 - November 26th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/09/2021	Gabi Urrutia
0.2	Document Edits	11/15/2021	Gokberk Gulgun
0.3	Final Draft	11/26/2021	Gokberk Gulgun
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 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 Compound Smart contract beginning on November 9th, 2021 and ending November 26th, 2021. The security assessment was scoped to the Algorand compound contracts and an audit of the security risk and implications regarding the changes introduced by the development team at Yieldly.Finance before its production release shortly following the assessment's deadline.

1.2 AUDIT SUMMARY

The team at Halborn was provided three weeks for the engagement and assigned two 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 solved and acknowledged by Yieldly.Finance team.

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

1.4 SCOPE

Code related to Yieldly Compound Staking Contract

Commit ID: 53be0e839a6c018aa934993a3edeccdbf384b4ba

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	0	7

LIKELIHOOD

IMPACT

		(HAL-01)		
(HAL-02) (HAL-03) (HAL-04) (HAL-05) (HAL-06) (HAL-07) (HAL-08)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - REWARDS ARE NOT UPDATED AFTER THE STAKE/WITHDRAW PROCESSES	Medium	SOLVED
HAL02 - LACK OF TEST CASE ON THE INTERNAL VARIABLE UPDATE	Informational	SOLVED
HAL03 - MISSING EMERGENCY WITHDRAW FUNCTION FOR THE USERS	Informational	ACKNOWLEDGED
HAL04 - AFTER THE POOL RATIO UPDATE THE USER CLAIMABLES ARE NOT UPDATED	Informational	ACKNOWLEDGED
HAL05 - LACK OF START DATE CHECK ON THE POOL CREATION	Informational	FUTURE RELEASE
HAL06 - LACK OF MULTISIG PROGRAM	Informational	ACKNOWLEDGED
HAL07 - MISSING PROXY ASSET DEFINITION ON THE FUNCTIONS	Informational	NOT APPLICABLE
HAL08 - MISSING FREEZE/REVOKE ASSETS DEFINITION	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) REWARDS ARE NOT UPDATED AFTER THE STAKE/WITHDRAW PROCESSES - MEDIUM

Description:

During the stake or withdraw progress, update rewards function not called by a user. According to **Rewards-Unlocked** variable, **Pool-Amount** is updated.

Code Location:

Listing 1

```
update_rewards:
// Set last updated to this timestamp
byte "Last_Updated"      // "Last_Updated"
global LatestTimestamp   // "Last_Updated" LatestTimestamp
itob
app_global_put           // null

// Calculates the amount of rewards to be unlocked
// Eqn: ((((End - Start) - (End - Current)) / (End - Start)) * Rewards_Locked) - Rewards_Unlocked
// Justification: Allows for dynamic calculation of reward unlocks after end date or start date is
// shifted

global LatestTimestamp   // LatestTimestamp
itob
byte "End_Date"          // LatestTimestamp "End_Date"
app_global_get           // LatestTimestamp intx
b>=                      // 1||0
bz setTimeStamp          // null (if 0 goto setTimeStamp)

byte "End_Date"          // "End_Date"
app_global_get           // intx
store 13                 // null

```

Recommendation:

Consider updating rewards amount after stake or withdraw processes.

Remediation Plan:

SOLVED - Yieldly.Finance claims that It is executed during noop operations that are specified as user interactions, which includes staking/withdrawing, and admin interactions such as emergency withdrawing.

3.2 (HAL-02) LACK OF TEST CASE ON THE UPDATE REWARDS FUNCTION - INFORMATIONAL

Description:

During the dynamic tests, It has been observed that update rewards function has lack of test cases. The tests cases should be added into repository.

Code Location:

Listing 2: Update Rewards Function

```
update_rewards:
// Set last updated to this timestamp
byte "Last_Updated"      // "Last_Updated"
global LatestTimestamp   // "Last_Updated" LatestTimestamp
itob
app_global_put           // null

// Calculates the amount of rewards to be unlocked
// Eqn: ((((End - Start) - (End - Current)) / (End - Start)) * Rewards_Locked) - Rewards_Unlocked
// Justification: Allows for dynamic calculation of reward unlocks after end date or start date is
// shifted

global LatestTimestamp   // LatestTimestamp
itob
byte "End_Date"          // LatestTimestamp "End_Date"
app_global_get           // LatestTimestamp intx
b>=                      // 1||0
bz setTimeStamp          // null (if 0 goto setTimeStamp)

byte "End_Date"          // "End_Date"
app_global_get           // intx
store 13                 // null
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Ensure that update rewards functionality is working according to workflow.

Remediation Plan:

SOLVED - Yieldly.Finance added the necessary test cases into the repository.

3.3 (HAL-03) MISSING EMERGENCY WITHDRAW FUNCTION FOR THE USERS - INFORMATIONAL

Description:

During the dynamic tests, It has been observed that only admin can call emergency withdraw functions. The user could not call emergency withdraw function for their funds.

Code Location:

Listing 3: Admin Only Function

```
emerg_withdraw:
global GroupSize        // GroupSize
int 1                   // GroupSize intx
==                      // 1||0
assert                  // null (if 0 then Failed)

callsub admin_check     // goto admin_check
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider to review structure and allows the user to withdraw their funds when the emergency status is occurred.

Remediation Plan:

ACKNOWLEDGED - Yieldly.Finance claims that the behavior is intended.

3.4 (HAL-04) AFTER THE POOL RATIO UPDATE THE USER CLAIMABLES ARE NOT UPDATED - INFORMATIONAL

Description:

During the static analysis, It has been observed that when pool ratio is updated, the user claimable are not updated on the withdrawal functionality.

Code Location:

Listing 4: Commented Out Function

```
// Update New Pool Ratio
callsub update_pool_ratio   // goto update_pool_ratio

// // Update claimable amount
// callsub update_user_claimable   // goto update_user_claimable
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider activating update_user_claimable function.

Remediation Plan:

ACKNOWLEDGED - Yieldly.Finance added the function only for testing purpose.

3.5 (HAL-05) LACK OF START DATE CHECK ON THE POOL CREATION - INFORMATIONAL

Description:

During the dynamic testing, It has been seen that start_Date variable is not checked. When the new pool is created, Start-Date should be equal or more than the Latest Timestamp.

Code Location:

Listing 5: Start Date Has Not Been Checked

```
byte "Admin"              // "Admin"
txn Sender                // "Admin" Sender
app_global_put            // null

//* Pool Dates */
byte "Start_Date"         // "Start_Date"
txn ApplicationArgs 0     // "Start_Date" ApplicationArgs
app_global_put            // null

byte "End_Date"           // "Start_Date"
txn ApplicationArgs 1     // "Start_Date" ApplicationArgs
app_global_put            // null
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider to check start date is more than the latest timestamp.

Remediation Plan:

PENDING: Yieldly.Finance will fix the vulnerability on the future release.

3.6 (HAL-06) LACK OF MULTISIG PROGRAM - INFORMATIONAL

Description:

The principal benefit of multi-signature is that it creates added redundancy in key management. While single signature addresses require only a single key for transactions, multi-signature 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 6: 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 considers to use multi-signature on the mainnet deployment.

3.7 (HAL-07) 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 7: winnerProgram Function (Lines 1)

```
1 let txn = await configs.winnerProgram(
2       account2,
3       escrowAddress,
4       algoAppId,
5       asaAppId,
6       trackerAppId,
7       winner,
8       rateAppId
9     );
```

Listing 8: assetOptoutApplication Function (Lines 1)

```
1 let txn1 = await configs.assetOptoutApplication(
2       account1,
3       escrowAddress,
4       optingAppId,
5       assetId
6     );
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

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

Remediation Plan:

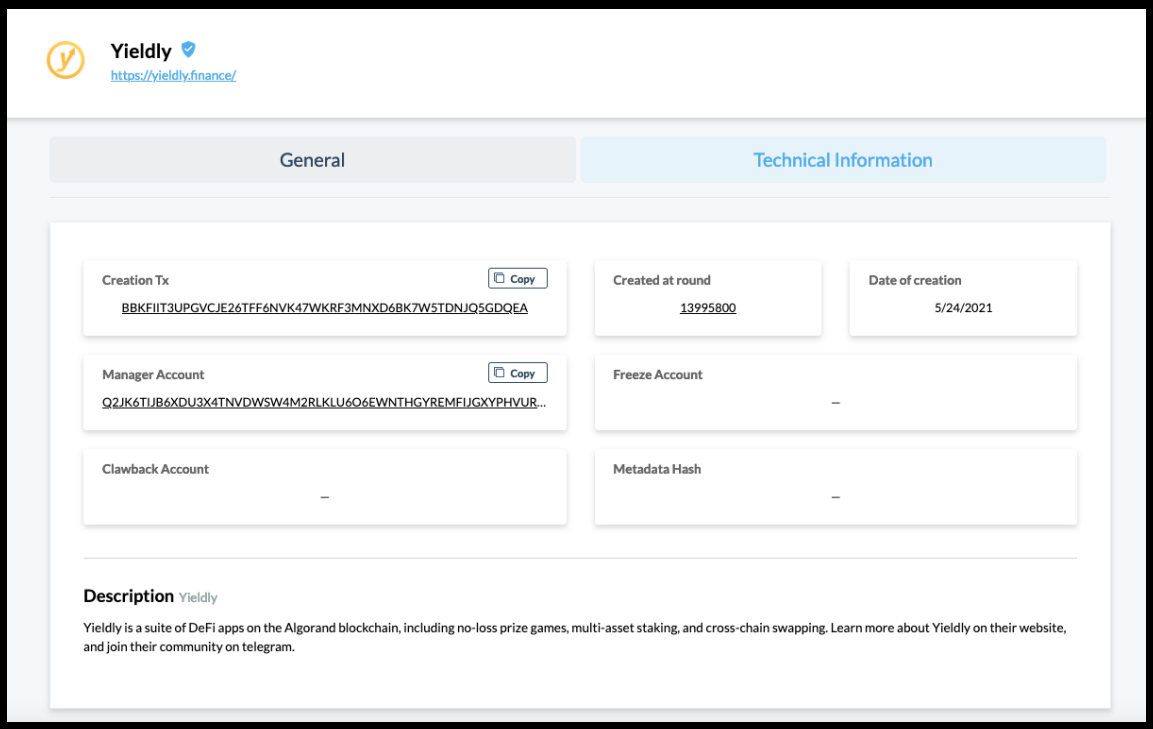
NOT APPLICABLE: Yieldly.Finance does not need to use proxy for the escrow asset after program version (5).

3.8 (HAL-08) MISSING FREEZE/REVOKE ASSETS DEFINITION - INFORMATIONAL

Description:

When an asset is created, the contract can provide a freeze address and a default frozen state. If the default frozen 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 before ownership. (KYC/AML) The clawback address, if specified, can 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:



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 disabled.

THANK YOU FOR CHOOSING

// HALBORN