GooseFX

Swap Program Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 25th, 2021 - November 3rd, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/25/2021	Piotr Cielas
0.2	Document Edits	10/28/2021	Piotr Cielas
0.3	Final Draft	11/03/2021	Piotr Cielas
0.3	Final Draft Review	11/04/2021	Gabi Urrutia
1.0	Remediation Plan	11/05/2021	Piotr Cielas
1.1	Remediation Plan Review	11/09/2021	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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

GooseFX is a full suite DeFi platform to trade crypto, tokenized stocks, NFTs and liquidity providing built on Solana and Serum DEX.

GooseFX engaged Halborn to conduct a security assessment on their Swap program beginning on October 25th, 2021 and ending November 3rd, 2021. This security assessment was scoped to the gfx-swap repository and an audit of the security risk and implications regarding the changes introduced by the development team at GooseFX prior to its production release shortly following the assessments deadline.

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned one full time security engineer to audit the security of the program. The engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of this audit to achieve the following:

- Ensure that program functions are intended.
- Identify potential security issues with the program.

In summary, Halborn identified some security risks that **were addressed by GooseFX team**.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual view of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the program audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of programs 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, purpose, and use of the platform.
- Manual code read and walkthrough.
- Manual Assessment of use and safety for the critical Rust variables and functions in scope to identify any arithmetic related vulnerability classes.
- Fuzz testing. (Halborn custom fuzzing tool)
- Checking the test coverage. (cargo tarpaulin)
- Scanning of Rust files for vulnerabilities.(cargo audit)

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

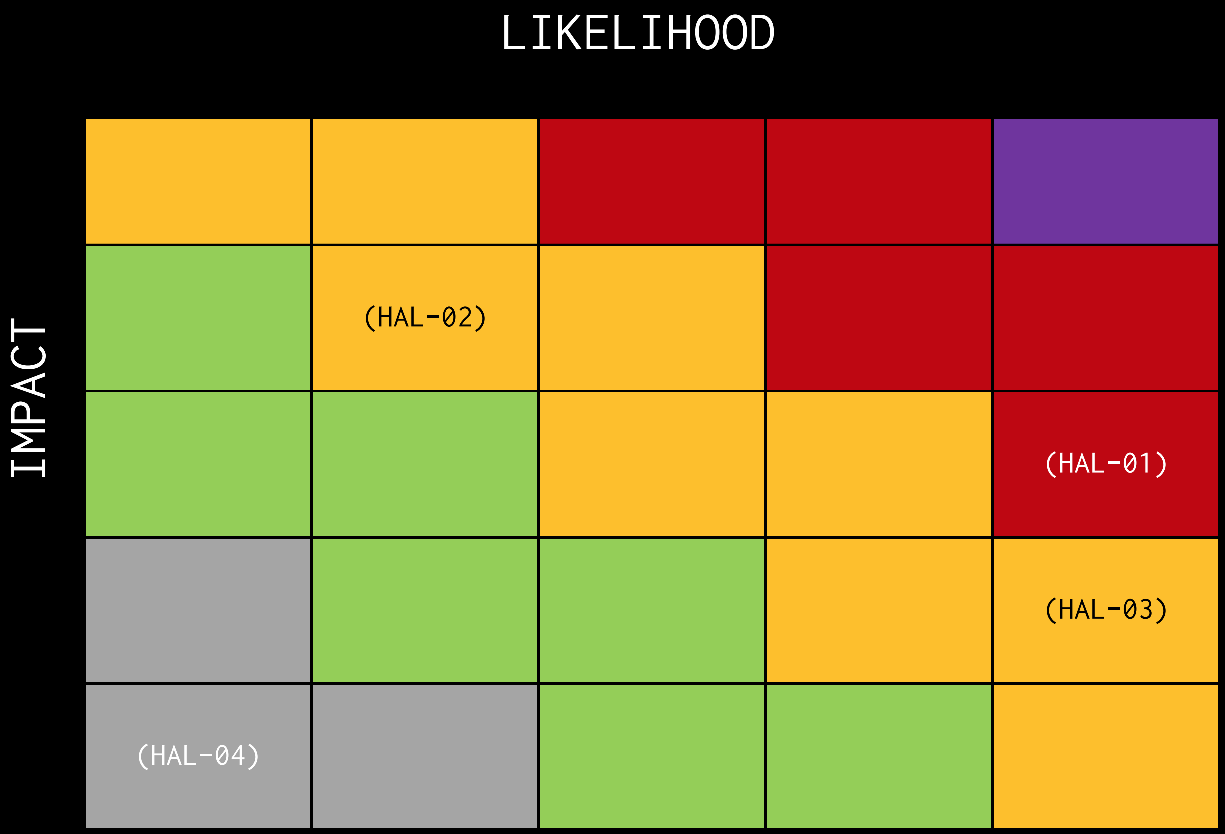
1.4 SCOPE

This review was scoped to the Swap Solana program.

1. Swap program
   (a) Repository: gfx-swap
   (b) Commit ID: 5dbb4bb4cf4d2a6ce0529402956cb6375cd826f7

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	2	0	1



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
TOKENS IN FEE VAULTS LOCKED INDEFINITELY	High	SOLVED - 11/03/2021
POOLS CANNOT BE SUSPENDED	Medium	SOLVED - 11/03/2021
OFFSET POOL TOKEN MISMATCH	Medium	SOLVED - 11/03/2021
CONSTRAINT FUNCTION ATTRIBUTES ARE NOT PRESERVED	Informational	SOLVED - 10/25/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) TOKENS IN FEE VAULTS LOCKED INDEFINITELY - HIGH

Description:

Users pay operational fees to fee_vault accounts for withdrawing liquidity and swapping tokens. On pool initialisation, the fee_vault token account authority is set to the relevant pool's account address, a PDA created from the gfx_swap program id, static seed and the pool address. The gfx_swap program does not allow transferring any fees from vaults which means they are locked in these accounts indefinitely.

Code Location:

Listing 1: contexts/initialize.rs (Lines 22)

```
11 #[derive(Accounts)]
12 #[instruction(seed: [u8; 32], pool_bump: u8, lp_bump: u8)]
13 pub struct Initialize<'info> {
14     pub admin: AccountInfo<'info>, // admin account can do
           privileged operations
15
16     #[account(
17         init,
18         seeds = [Pool::IDENT, &seed],
19         bump = pool_bump,
20         payer = payer,
21     )]
22     pub pool: Account<'info, Pool>,
23
```

Listing 2: cli/src/bin/create_pool.rs (Lines 53)

```
43 let client = Client::new_with_options(
44     Cluster::Devnet,
45     CliKeypair::copy(&admin),
46     CommitmentConfig::confirmed(),
47 );
48 let program = client.program(program_id);
```

```

// seed for creating the pool
let seed = solana_sdk::signature::Keypair::new().pubkey();

let (pool, pool_bump) = Pool::get_address_with_bump(&program_id, &
    seed.to_bytes());
let (lp_mint, lp_bump) = LPMint::get_address_with_bump(&program_id
    , &seed.to_bytes());

println!("Creating the LP pool ...");

```

Listing 3: cli/src/bin/create_pool.rs (Lines 20)

```
fn get_address_with_bump(program_id: &Pubkey, seed: &[u8]) -> (
    Pubkey, u8) {
    Pubkey::find_program_address(&[Self::IDENT, seed], program_id)
}
```

Risk Level:

Likelihood - 5
Impact - 3

Recommendations:

Consider implementing either a governance function to allow transferring tokens from pool fee vaults or change fee vault authority accounts to a one with matching private key.

Remediation Plan:

SOLVED: The GooseFX team fixed the issue in commit fe70730908889551ce653e18f8dd9b0d1ddfd6ee.

3.2 (HAL-02) POOLS CANNOT BE SUSPENDED - MEDIUM

Description:

suspended is one of the fields in the Pool struct. If set to true, all operations on a pool are suspended--swapping,depositing and withdrawing are blocked. This field is set to false by default and its value cannot be modified as the program does not feature any instructions that update pool parameters.

Code Location:

Listing 4: states/pool.rs (Lines 27)

```
11 #[account]
12 #[derive(Default, Debug)]
13 pub struct Pool {
14     pub seed: [u8; 32],
15     pub bump: u8,
16     pub lp_bump: u8,
17     pub admin: Pubkey,
18     // sorted by token mint addresses
19     pub token_mint_1: Pubkey,
20     pub token_mint_2: Pubkey,
21     pub token_vault_1: Pubkey,
22     pub token_vault_2: Pubkey,
23     pub mint: Pubkey, // the LP token mint
24     pub fee_vault: Pubkey,
25     pub fees: Fees,
26     pub curve: SwapCurve,
27     pub suspended: bool,
28 }
```

Listing 5: contexts/initialize.rs

```
let (token_vault_1, token_vault_2) =
    (token_a_vault.mint, token_b_vault.mint).sort(token_a_vault,
      token_b_vault)?;
```

```
let (token_mint_1, token_mint_2) = (token_a_mint, token_b_mint).
    sort_self()?;

pool.admin = admin.key();
pool.seed = seed;
pool.bump = pool_bump;
pool.lp_bump = lp_bump;
pool.token_mint_1 = token_mint_1.key();
pool.token_mint_2 = token_mint_2.key();
pool.token_vault_1 = token_vault_1.key();
pool.token_vault_2 = token_vault_2.key();
pool.mint = lp_token_mint.key();
pool.fee_vault = lp_token_ata_fee.key();
pool.fees = fees;
pool.curve = curve;
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendations:

Implement an instruction handler for updating pool parameters.

Remediation Plan:

SOLVED: The GooseFX team fixed the issue in commit fe70730908889551ce653e18f8dd9b0d1ddfd6ee.

3.3 (HAL-03) OFFSET CURVE TOKEN MISMATCH - MEDIUM

Description:

The offset curve is one of the swap price curves the gfx_swap program offers. On pool initialization, in the process function in initialize.rs user specify to which of the two tokens the offset should be added. This function however also sorts the tokens by mint address and in case token_b mint happens to be a lower number than token_a the offset will actually be added to token_a instead because the calculator has no concept of token ordering. This might cause the pool config to be exactly opposite to whatever it is expected to be thereby incurring a minimum loss of $0.64 to the pool's creator as of the time of writing.

Code Location:

Listing 6: states/pool.rs (Lines 27)

```
#[account]
#[derive(Default, Debug)]
pub struct Pool {
    pub seed: [u8; 32],
    pub bump: u8,
    pub lp_bump: u8,
    pub admin: Pubkey,
    // sorted by token mint addresses
    pub token_mint_1: Pubkey,
    pub token_mint_2: Pubkey,
    pub token_vault_1: Pubkey,
    pub token_vault_2: Pubkey,
    pub mint: Pubkey, // the LP token mint
    pub fee_vault: Pubkey,
    pub fees: Fees,
    pub curve: SwapCurve,
    pub suspended: bool,
}
```

Listing 7: contexts/initialize.rs

```rust
let (token_vault_1, token_vault_2) =
    (token_a_vault.mint, token_b_vault.mint).sort(token_a_vault, token_b_vault)?;
let (token_mint_1, token_mint_2) = (token_a_mint, token_b_mint).sort_self()?;

pool.admin = admin.key();
pool.seed = seed;
pool.bump = pool_bump;
pool.lp_bump = lp_bump;
pool.token_mint_1 = token_mint_1.key();
pool.token_mint_2 = token_mint_2.key();
pool.token_vault_1 = token_vault_1.key();
pool.token_vault_2 = token_vault_2.key();
pool.mint = lp_token_mint.key();
pool.fee_vault = lp_token_ata_fee.key();
pool.fees = fees;
pool.curve = curve;
```

Listing 8: src/curve/swap_curve.rs (Lines 46)

```rust
33 /// All the supported curve types. We do not use the trait object solution in the SPL.
34 /// Instead, we use enums.
35 #[enum_dispatch]
36 #[repr(C)]
37 #[derive(Clone, Debug, AnchorDeserialize, AnchorSerialize)]
38 pub enum SwapCurve {
39     /// Uniswap-style constant product curve, invariant = token_a_amount * token_b_amount
40     ConstantProductCurve,
41     /// Flat line, always providing 1:1 from one token to another
42     ConstantPriceCurve,
43     /// Stable, like uniswap, but with wide zone of 1:1 instead of one point
44     StableCurve,
45     /// Offset curve, like Uniswap, but the token B side has a faked offset
46     OffsetCurve,
47 }
```

Listing 9: curve/calculators/offset.rs (Lines 20)

```
/// Offset curve, uses ConstantProduct under the hood, but adds an
    offset to
/// one side on swap calculations
#[derive(Clone, Debug, Default, PartialEq, AnchorDeserialize,
    AnchorSerialize)]
pub struct OffsetCurve {
    /// Amount to offset the token B liquidity account
    pub token_b_offset: u64,
}
```

Risk Level:

Likelihood - 5
Impact - 2

Recommendations:

Remember to assign the offset to the user-selected token on pool initialisation.

Remediation Plan:

SOLVED: The GooseFX team fixed this issue in commit 57f0e97da665c943b22870e70c3314fdaba19e8b.

3.4 (HAL-04) CONSTRAINT FUNCTION ATTRIBUTES ARE NOT PRESERVED - INFORMATIONAL

Description:

The suspended function defined in src/constraints.rs is marked with the #[throws] attribute which causes the function to return an error defined in the attribute's argument. This function is used by the access_control macro by constraints of several instructions defined in lib.rs. The access_control macro however does not preserve attributes which means the suspended function will never throw an error.

Code Location:

Listing 10: contexts/constraints.rs (Lines 5)

```
5 #[throws(ProgramError)]
6 pub fn suspended(pool: &Account<'_, Pool>) {
7     require!(!pool.suspended, Suspended);
8 }
```

Listing 11: src/lib.rs (Lines 51)

```
48 #[account]
49 #[derive(Default, Debug)]
50 #[throws(ProgramError)]
51 #[access_control(suspended(&ctx.accounts.pool))]
52 pub fn deposit2(
53     ctx: Context<Deposit2>,
54     lp_token_amount: u64,
55     maximum_token_a_amount: u64,
56     maximum_token_b_amount: u64,
57 ) {
58     ctx.accounts.process(
59         lp_token_amount,
60         maximum_token_a_amount,
61         maximum_token_b_amount,
    )?
}
```

anchor-attributes-access-control crate:

Listing 12: src/lib.rs (Lines 71)

```
let item_fn = parse_macro_input!(input as syn::ItemFn);

let fn_vis = item_fn.vis;
let fn_sig = item_fn.sig;
let fn_block = item_fn.block;

let fn_stmts = fn_block.stmts;

proc_macro::TokenStream::from(quote! {
    #fn_vis #fn_sig {

        #(#access_control)*

        #(#fn_stmts)*
    }
})
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Upgrade Anchor to commit 5fa263ff176a6f3f1e8fb1e6667da0ec5999b6e9.

Remediation Plan:

SOLVED: The GooseFX team fixed the issue in commit 91e84b0322696ce9f89906c22e3134de57964294.

AUTOMATED TESTING

4.1 AUTOMATED ANALYSIS

Description:

Halborn used automated security scanners to assist with detection of well known security issues and vulnerabilities. Among the tools used was cargo audit, a security scanner for vulnerabilities reported to the RustSec Advisory Database. All vulnerabilities published in https://crates.io are stored in a repository named The RustSec Advisory Database. cargo audit is a human-readable version of the advisory database which performs a scanning on Cargo.lock. Security Detections are only in scope. All vulnerabilities shown here were already disclosed in above report. However, to better assist the developers maintaining this code, the auditors are including the output with the dependencies tree, and this is included in the cargo audit output to better know the dependencies affected by unmaintained and vulnerable crates.

ID	**package**	**Short Description**
RUSTSEC-2020-0159	chrono	Potential segfault in localtime_r invocations
RUSTSEC-2021-0119	nix	Out-of-bounds write in nix::unistd::getgrouplist
RUSTSEC-2021-0080	tar	Links in archive can create arbitrary directories
RUSTSEC-2020-0071	time	Potential segfault in the time crate
RUSTSEC-2021-0115	zeroize_derive	#[zeroize(drop)] doesn't implement Drop for enums

THANK YOU FOR CHOOSING

// HALBORN