Soex Smart Contract

SMART CONTRACT AUDIT REPORT

October 2024

ExVul

www.exvul.com

Table of Contents

1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by Soex to review smart contract implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

1.1 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood can be: High, Medium and Low and impact are categorized into for: High, Medium, Low, Informational. Severity is determined by likelihood and impact and can be classified into five categories accordingly, Critical, High, Medium, Low, Informational shown in table 1.1.

Likelihood \ IMPACT	*Informational*	*Low*	*Medium*	*High*
High	*Informational*	Medium	High	Critical
Medium	*Informational*	Low	Medium	High
Low	*Informational*	Low	Low	Medium

Table 1.1 Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy contracts on our private test environment and run

tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- Basic Coding Bugs: We first statically analyze given smart contracts with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- Code and business security testing: We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.
- Additional Recommendations: We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

Category	Assessment Item
Basic Coding Assessment	Apply Verification Control
	Authorization Access Control
	Forged Transfer Vulnerability
	Forged Transfer Notification
	Numeric Overflow
	Transaction Rollback Attack
	Transaction Block Stuffing Attack
	Soft Fail Attack
	Hard Fail Attack
	Abnormal Memo
	Abnormal Resource Consumption
	Secure Random Number
Advanced Source Code Scrutiny	Asset Security
	Cryptography Security
	Business Logic Review
	Source Code Functional Verification
	Account Authorization Control
	Sensitive Information Disclosure
	Circuit Breaker
	Blacklist Control
	System API Call Analysis
	Contract Deployment Consistency Check
	Semantic Consistency Checks

Category	Assessment Item
Additional Recommendations	Following Other Best Practices

Table 1.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

1.2 Severity Classification

Bugs are classified on a 5-level scale: Critical, High, Medium, Low, Informational.

Critical

- Manipulation of governance vote results, deviation from the vote result, resulting in a direct change in the intended effect of the original result: This effect is when the governance vote results within the platform are tampered with or changed, resulting in a different result than what voters on the governance voting platform originally intended. Attacks on governance are critical because governance often has privileged access to sensitive protocol functions and/or may have custody of vaults that could be drained by the attacker.
- Direct theft of any user funds other than unclaimed proceeds, whether static or dynamic: Theft of user funds is the worst case scenario for a project. If an attacker can manipulate the system so that users lose money during transfers and the attacker gains, it is considered direct theft of user funds. If users lose their shares, principal, vault balances, etc., it is theft of user funds.
- Direct theft of any user funds, whether static or dynamic. If an attacker can manipulate the system so that users receive nothing or less than the price set by the user, it is considered direct theft of user funds.
- Permanently freeze funds, this includes freezing the contracts holding tokens so that users can no longer withdraw their funds. It may also include destroying funds so that the owner can no longer access them. This also includes things like self-destructing implementation contracts so that the proxy becomes useless. The impact here is that funds within the system are no longer available.
- Unauthorized minting of NFTs: Unauthorized minting can have the effect that users can mint more NFTs than they should, or they can mint NFTs without providing any tokens.
- Predictable or manipulable RNGs, leading to abuse of delegates or NFTs: This attack gives the attacker an unfair advantage over the rarity of the NFT, as the rarity of NFTs is often determined using an RNG. An attacker can exploit this attack vector by setting up bots that front-run.
- Unintentional changes to what the NFT represents (e.g. token URI, payload, art content): Some NFTs store a link at the smart contract level that refers to the actual content of the NFT. An unintentional change is when a user is able to manipulate the represented value of an NFT, which causes the NFT to display content that may not belong to the NFT.

- Unintended consequences of Maximum Extractable Value (MEV) exploration: MEV is a measure of the profits miners (or validators, sorters, etc.) can make by arbitrarily including, excluding, or reordering transactions in blocks they generate. This can include front-running, sandwich attacks, and liquidations. Opportunities for MEV mean that people not participating in the protocol are profiting. In some cases (such as projects with relayers, meta-transactions, liquidators, guardians, etc.), this may be a core part of the protocol, in which case this particular opportunity may be excluded from scope.
- Protocol bankruptcy: Some protocols offer yields to some users that are paid for by other users (e.g., Compound lenders owe borrowers the yield provided). If the calculation is wrong, it may result in users being owed more than other users are owed. This is bankruptcy. Alternatively, the protocol's debt may otherwise exceed its assets. Of course, this does not include "bank runs" where funds cannot be temporarily extracted from the protocol, but the protocol is otherwise well-collateralized.
- Signature replay attack,Signature replay attacks should ensure that signatures can only be executed as many times as they are designed. Multiple executions may directly steal funds, including cross-chain and cross-method replays.

High

- Theft of unclaimed yields, yields are any assets distributed as rewards for participating in the system. Any theft of these rewards before they are distributed or claimed is classified as theft of unclaimed yields.
- Theft of unclaimed royalties, royalties are any assets distributed as rewards for participating in the system. Any theft of these rewards before they are distributed or claimed is classified as theft of unclaimed royalties.
- Permanent freeze of unclaimed yields: yields are any assets distributed as rewards for participating in the system. Whenever an attacker can prevent yield from being transferred from the contract, such as by making the harvest() function always fail, this means that yield is permanently frozen.
- Permanent freeze of unclaimed royalties: royalties are any assets distributed as rewards for participating in the system. Unclaimed Royald are considered frozen as long as an attacker can prevent users from permanently claiming Royald.
- Temporary freezing of NFTs: This classification refers to the temporary freezing of user NFTs that the attacker does not own. Whenever the attacker can prevent users from accessing NFTs by moving them to another address that the protocol still controls, the NFTs are temporarily frozen and the protocol can restore them to their original location.
- Greatly disrupting the project's operating logic: This classification refers to the attacker's serious damage to the project's operating logic through vulnerabilities. Whenever the attacker is able to manipulate or interfere with the key operating process of the smart contract, causing the expected operating logic of the protocol to deviate or block, it will have a great negative impact.
- Receiving excessive rewards: This classification refers to the attacker's multiple or excessive withdrawal of rewards that should belong to users or protocols through smart contract vulnerabilities. When an attacker is able to manipulate the reward distribution process so that the actual rewards received exceed the expected or designed amount, it will pose a threat to the project's economic model or user funds.

Medium

- Smart Contract Cannot Run Due to Lack of Token Funds: This classification refers to a bug that marks a smart contract as not being able to run or work properly due to lack of token funds. In some cases, a smart contract cannot pay any rewards for staked tokens because the contract does not hold any funds or accept any reimbursements. Another example is the LINK tokens required to pay for certain Chainlink services. If these services are necessary for the system to function properly and the funds can (or may) run out, that would be a bug.
- Block Stuffing: This classification refers to an attacker who can create many transactions that will enter a block or win a flashbots auction to take over an entire block to prohibit others from making transactions to the smart contract.
- Malicious Sabotage (e.g., the attacker has no profit motive, but causes damage to users or the protocol): Malicious sabotage is when an attacker calls certain functions of a smart contract that puts it in a suboptimal state, preventing normal function execution for any user. This causes the user to lose money by sending a transaction, but when the smart contract returns to normal, the user will be able to call the function again to complete it. In this case, the attacker harms the user by requiring the user to send another transaction. The attacker does not profit, but they do harm the user or the protocol.
- Gas theft occurs when an attacker causes a relayer to perform a call to a contract controlled by the attacker that consumes significantly more gas than they should, and that gas is used for the attacker's benefit. Another example of this is when a relayer is misconfigured and accepts high gas limits. In this case, an attacker can use these high gas limits to cause the relayer to perform unrelated, complex transactions that will use gas that does not exceed that limit for their own benefit.
- Unbounded gas consumption: Any loop over an array of arbitrary size is potentially vulnerable to unbounded gas consumption. If an attacker can add enough items to cause the gas used to call a function to exceed the block gas limit, this can cause a denial of service attack and prevent the function from being called.
- Certain circumstances can lead to loss of funds: This classification refers to the ability of an attacker to exploit a contract vulnerability under certain operations or conditions to cause a loss of funds. When certain conditions are met, such as lax bounds checking or a misordering of certain function calls, an attacker can exploit the vulnerability to extract or transfer funds, resulting in a loss of assets for users or the protocol.
- Certain circumstances may cause the project to be obstructed: This category refers to the situation where attackers can exploit contract vulnerabilities to hinder the normal operation of the project under certain operations or under certain conditions. This usually occurs when boundary conditions are improperly handled or function call logic is improper, which limits or prevents certain key functions from being executed normally, causing some or all operations of the project to be temporarily interrupted or stuck.
- There are significant differences between project logic and code logic: This category refers to the fact that the project has a large deviation from the actual implementation of the contract code at the design or intention level, resulting in the inability of users or participants to perform as expected. This difference may lead to operational results that are not in line with the original design intention, such as incorrect fund flows, improper operating permissions, or abnormal system behavior, which affects the user experience and even endangers the overall security and stability of the project.

Low

- The contract fails to deliver the promised return, but does not lose value: This is when the code does not work as expected, that is, there are some logical errors, but the logical errors do not affect the funds of the protocol or user funds.
- Project logic or funds are damaged in extreme cases: This category refers to the core logic or fund security of the project may be affected under extreme conditions (such as abnormal market fluctuations or unexpected system behavior). This situation may cause funds to be lost, the project to operate unexpectedly, or key functions to stagnate, which in turn harms the interests of users or the stability of the project, and may require urgent code fixes or parameter adjustments to mitigate risks.
- Unexpected user behavior caused by loose logic restrictions: This category refers to the logic restrictions of smart contracts that are not strict enough, allowing users to perform actions beyond their intentions. This may lead to abuse of functions or unexpected operations. Although such vulnerabilities do not always directly endanger funds, they will affect the security and stability of the project and may cause adverse consequences for users or the system.
- Logical errors caused by informal code writing or linking: This category refers to potential logical errors in smart contracts caused by irregular code writing or external linking issues. Such errors may cause the contract to behave inconsistently with expectations or cause abnormal operations in certain circumstances. Although it may not directly lead to financial loss, it will affect the reliability and stability of the contract and increase the risk of unexpected behavior.

Informational

- Unused code snippets: This category refers to unused code snippets in the contract. These codes will not affect the normal operation or functional performance of the contract, but will increase the complexity of the code, may cause confusion and reduce the readability and maintainability of the code. Although it does not directly lead to security risks or operational problems, removing unused code helps improve the quality of the contract in terms of code neatness and standardization.
- Lack of event notifications for important operations: This category refers to the fact that key operations in smart contracts do not issue event notifications, resulting in the inability of the outside world to know the changes of important states in real time. This defect may cause insufficient transparency in contract interactions, affect users' tracking of transaction processes, and increase the difficulty of operation management. Although it does not directly affect the operation of the contract, the lack of event notifications will reduce the efficiency of monitoring and auditing contracts, and have a negative impact on the observability of users and protocols.
- Calculations that have no effect on contract operation: This category refers to the inclusion of calculation operations in smart contracts that do not affect the core logic or operation of the contract. Such calculations will not interfere with the flow of funds or transaction logic, but may occupy some additional storage or computing resources. Although it does not cause direct security risks, removing or optimizing such calculations can improve the overall clarity of the code and contract performance.
- Possible meaningless calls: This category refers to calls that may be included in the contract without performing any actual operations or effects. These calls neither affect the flow of funds nor change the contract status. Although they do not pose a direct security risk, meaningless calls increase transaction costs. Optimizing such calls can help improve the performance of the contract and code clarity.

2. FINDINGS OVERVIEW

2.1 Project Info And Contract Address

Project Name: Soex

Audit Time: September 2, 2024 – October 31, 2024

Language: Rust

File Name	Link
Soex	https://github.com/soexdev/soex-protocol 041a25073b115d2922536d428ae5590e9a78ffb9

2.2 Summary

Severity	Found	
Critical	1	
High	3	
Medium	15	
Low	14	
Informational	7	

2.3 Key Findings

ID	Severity	Findings Title	Status	Confirm
NVE-001	Critical	Complete consistency of signature information may lead to signature abuse	Fixed	Confirmed
NVE-002	High	Problem with calling initialize method repeatedly	Fixed	Confirmed
NVE-003	High	Mint cvt round check should be done before user transfer	Fixed	Confirmed
NVE-004	High	Signature replay handle_claim_hvt_share no limit	Fixed	Confirmed
NVE-005	Medium	cvt mint nft_id may not be what you expect	Fixed	Confirmed
NVE-006	Medium	collection.burn_idx is not updated when unfollow	Fixed	Confirmed
NVE-007	Medium	quit Signature does not verify the owner of nft	Fixed	Confirmed
NVE-008	Medium	Admin calls handle_verify_cvt_sol_account twice, which may cause DOS transfer	Fixed	Confirmed
NVE-009	Medium	OG mint parameter configuration is incorrect	Fixed	Confirmed
NVE-010	Medium	Maximum value limit of mint	Fixed	Confirmed
NVE-011	Medium	May be preemptively initialized	Fixed	Confirmed
NVE-012	Medium	unfollow price bucket may not be consistent with expectations	Ignored	Confirmed
NVE-013	Medium	Class B HVT No Charge Fixed 0.01 sol	Fixed	Confirmed
NVE-014	Medium	Name verification problem	Fixed	Confirmed
NVE-015	Medium	Problem with arbitrary user calling the initialize method	Fixed	Confirmed
NVE-016	Medium	Multiplication before division causes precision problems	Fixed	Confirmed
NVE-017	Medium	Cancelling the pledge does not update the total amount of pledged	Fixed	Confirmed

ID	Severity	Findings Title	Status	Confirm
NVE-018	Medium	The status of the unstaking was not updated in time	Fixed	Confirmed
NVE-019	Medium	There is no fixed logic of charging 0.01 sol when claiming LP rewards	Fixed	Confirmed
NVE-020	Low	Stake and unstake cannot be frozen	Fixed	Confirmed
NVE-021	Low	Stake and unstake do not check if amount is not 0	Ignored	Confirmed
NVE-022	Low	Any caller can preemptively modify is_open_sol_reward_pool	Fixed	Confirmed
NVE-023	Low	Test environment configuration	Fixed	Confirmed
NVE-024	Low	Spelling Errors	Fixed	Confirmed
NVE-025	Low	Transfer unverified user balance	Ignored	Confirmed
NVE-026	Low	Unfollow uses price buckets for all fee modes	Ignored	Confirmed
NVE-027	Low	Any user preemptively initializes	Fixed	Confirmed
NVE-028	Low	Calculation accuracy issues	Ignored	Confirmed
NVE-029	Low	Check owner does not work	Ignored	Confirmed
NVE-030	Low	The handle_init method has a preemptive initialization problem	Fixed	Confirmed
NVE-031	Low	No judgment admin privilege role is not a zero address	Fixed	Confirmed
NVE-032	Low	cvt_sol_account setting error will not be executed later	Ignored	Confirmed
NVE-033	Low	Check if the claim has been opened, otherwise the start slot time may be reset	Fixed	Confirmed
NVE-034	Informational	Computational issues	Ignored	Confirmed
NVE-035	Informational	handle_init is empty	Fixed	Confirmed
NVE-036	Informational	cvt mint no events	Fixed	Confirmed

ID	Severity	Findings Title	Status	Confirm
NVE-037	Informational	hvt mint no events	Fixed	Confirmed
NVE-038	Informational	Reward calculation accuracy issue	Ignored	Confirmed
NVE-039	Informational	If closed is equal to collection.closed, an error is returned	Fixed	Confirmed
NVE-040	Informational	should have msg when set max supply	Fixed	Confirmed

Table 2.3: Key Audit Findings

3. DETAILED DESCRIPTION OF FINDINGS

3.1 Complete consistency of signature information may lead to signature abuse

ID:	NVE-001	**Location:**	treasury/src/instructions/handle_receive_sol.rs
Severity:	Critical	**Category:**	Verification Control
Likelihood:	High	**Impact:**	High

Description:

Comparing the codes of dao and treasury, we found that they are identical except for the name, and the msg information of handle_transfer is also exactly the same.

dao uses treasury as the signature, which leads to problems. If the user withdraws money from treasury, then if there is the same nonce in dao, for example, both are the first transaction, then there may be a problem of duplicate transfer.

```rust
use anchor_lang::system_program;

pub fn handle_receive_sol(
    ctx: Context<ReceiveSol>,
    address: Pubkey,
    input_amount: u64,
    output_amount: u64,
    nonce: u64,
    signature: [u8; 64],
    payload: Vec<u8>,
    treasury: u8,
) -> Result<()> {
    let config = ctx.accounts.global_config.load()?;

    if ctx.accounts.output_account.owner != address {
        return Err(ErrorCode::InvalidOutputAccount.into());
    }

    let addr_nonce = ctx.accounts.address_manager.nonce + 1;
    assert_eq!(addr_nonce, nonce);

    let mut msg = vec![];
    msg.extend(address.to_bytes());
    msg.extend(payload.len().to_le_bytes());
    msg.extend(payload.clone());
    msg.extend(input_amount.to_le_bytes());
    msg.extend(output_amount.to_le_bytes());
    msg.extend(nonce.to_le_bytes());
    msg.extend(ctx.accounts.output_account.mint.to_bytes());
    msg.extend(treasury.to_le_bytes());

    let hash = keccak::hash(&msg).to_bytes();
    msg!("hash {}", Pubkey::new_from_array(hash));
    let ix: Instruction = load_instruction_at_checked(2, &ctx.accounts.ix_sysvar
    utils::verify_ed25519_ix(&ix, &config.oracle.to_bytes(), &hash, &signature)?
```

Recommendations:

Exvul Web3 Security recommends performing inconsistency checks on signature information.

Result: Confirmed

Fix Result: Fixed

The signatures of DAO and Treasury are distinguished.

```
31    let mut msg = vec![];
32    msg.extend((0x64616f as u64).to_le_bytes());
33    msg.extend(address.to_bytes());
```

```
31    let mut msg = vec![];

32    msg.extend(address.to_bytes());
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.2 Problem with calling initialize method repeatedly

ID:	NVE-002	Location:	cvt/src/lib.rs
Severity:	High	Category:	Verification Control
Likelihood:	Medium	Impact:	High

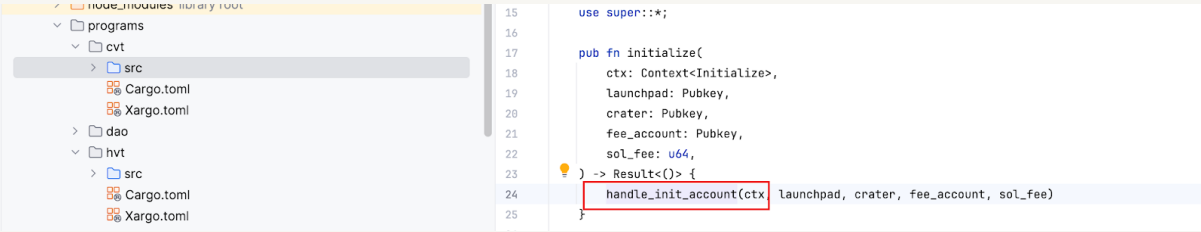
Description:

The current initialize method does not have logic to prevent repeated calls, which means that even if the contract has been successfully initialized, anyone can still call initialize again to reset the state of the contract.

Possible problems caused by repeated calls to initialize:

Overwriting existing initial settings: For example, if the initialization has already set up the administrator account and other key configurations, repeated calls may overwrite these settings or even be abused by malicious users.

Logical inconsistency: The state of the smart contract may be accidentally overwritten, resulting in unexpected results in subsequent operations.



Recommendations:

Exvul Web3 Security recommends setting a state identifier in the contract (such as is_initialized) to track whether the contract has been initialized. If it has been initialized, subsequent initialization calls will be rejected. Or use the built-in load_init().

Result: Confirmed

Fix Result: Fixed

Added inited check:

```
require!((ctx.accounts.collection.inited & 1) == 0, CVTErrorCode::AlreadyInited);
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.3 Mint cvt round check should be done before user transfer

ID:	NVE-03	Location:	launchpad/src/instructions/mint_cvt.rs
Severity:	High	Category:	Business Issues
Likelihood:	Medium	Impact:	High

Description:

The mint cvt round check should be done before the user transfer. It is possible that at the end of the round the user transfers sol but there is no mint cvt.

```
msg!("Transfer WSOL to contract account");
transfer_token_to_pool(
    &ctx.accounts.user_sol_account,
    ctx.accounts.sol_account.to_account_info(),
    ctx.accounts.sol_token.to_account_info(),
    mint_fee,
    &ctx.accounts.token_program,
    ctx.accounts.signer.to_account_info(),
    ctx.accounts.sol_token.decimals,
)?;

ctx.accounts.sol_account.reload()?;
```

Recommendations:

Exvul Web3 Security recommends that mint cvt round check should be done before user transfer.

Result: Confirmed

Fix Result: Fixed

Modified the inspection order:

```
57      if ctx.accounts.release_config.is_ended() {
58          return Ok(());
59      }
60
61      let overflow :u64  = mint_fee - global_config.floor_price;
62      if overflow > 0 {
63          let overflow_fee :u64  = overflow * ctx.accounts.overflow_config.percent / 100;
64          msg!("overflow_fee_account:{}",ctx.accounts.overflow_fee_account.to_account_info().key
65          let cpi_context :CpiContext<Transfer>  = CpiContext::new(
66              ctx.accounts.system_program.to_account_info(),
67              accounts: system_program::Transfer {
68                  from: ctx.accounts.signer.to_account_info().clone(),
69                  to:ctx.accounts.overflow_fee_account.to_account_info(),
70              },
71          );
72          system_program::transfer(cpi_context,overflow_fee)?;
73
74          mint_fee = mint_fee - overflow_fee;
75          msg!("Transfer wsol to overflow_fee account");
76      }
77
78      msg!("Transfer wsol to contract account");
79      transfer_token_to_pool(
80          &ctx.accounts.user_sol_account,
81          ctx.accounts.sol_account.to_account_info(),
82          ctx.accounts.sol_token.to_account_info(),
83          mint_fee,
84          &ctx.accounts.token_program,
85          ctx.accounts.signer.to_account_info(),
86          ctx.accounts.sol_token.decimals,
87      )?;
88
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.4 Signature replay handle_claim_hvt_share no limit

ID:	NVE-04	Location:	hvt_launchpad/src/instructions/handle_claim_hvt_share.rs
Severity:	High	Category:	Business Issues
Likelihood:	Medium	Impact:	High

Description:

Signature replay handle_claim_hvt_share no limit.

```
pub fn handle_claim_hvt_share(
    ctx: Context<ClaimHvtShare>,
    cvt_name: String,
    hvt_name: String,
    price: u64,
    id: Vec<u8>,
    outdate_time: u64,
    signature: Vec<u8>,
) -> Result<()> {
    msg!("Claim hvt share");

    let config :&Box<Account<Config>> = &ctx.accounts.config;
    let collection :Ref<CollectionAccount> = ctx.accounts.collection.load()?;

    // Check if the mint_account balance is greater than 1
    if ctx.accounts.mint_account.amount <= 1 {
        return Err(ProgramError::InsufficientFunds.into());
    }

    let timestamp :u64 = Clock::get()?.unix_timestamp as u64;
    require!(timestamp < outdate_time, HvtErrorCode::Outdated);

    // Verify the ed25519 signature
    let mut message :Vec<u8> = vec![];
    message.extend_from_slice(&ctx.accounts.payer.key().to_bytes());
    message.extend_from_slice(&cvt_name.as_bytes());
    message.extend_from_slice(&hvt_name.as_bytes());
    message.extend_from_slice(&price.to_le_bytes());
    message.extend_from_slice(&id);
    message.extend_from_slice(&outdate_time.to_le_bytes());

```

Recommendations:

Exvul Web3 Security recommends adding user nonce to prevent replay.

Result: Confirmed

Fix Result: Fixed

Added nonce to avoid replay:

```
27
28      let address_nonce :u64 =  ctx.accounts.nonce_account.nonce + 1;
29      require!(address_nonce == nonce,HvtErrorCode::InvalidNonce);
30
31      // Verify the ed25519 signature
32      let mut message :Vec<u8> = vec![];
33      message.extend_from_slice(&ctx.accounts.payer.key().to_bytes());
34      message.extend_from_slice(&cvt_name.as_bytes());
35      message.extend_from_slice(&hvt_name.as_bytes());
36      message.extend_from_slice(&claim_amount.to_le_bytes());
37      message.extend_from_slice(&tracing_id.as_bytes());
38      message.extend_from_slice(&nonce.to_le_bytes());
```

Fixed version:

https://github.com/soexdev/soex-protocol

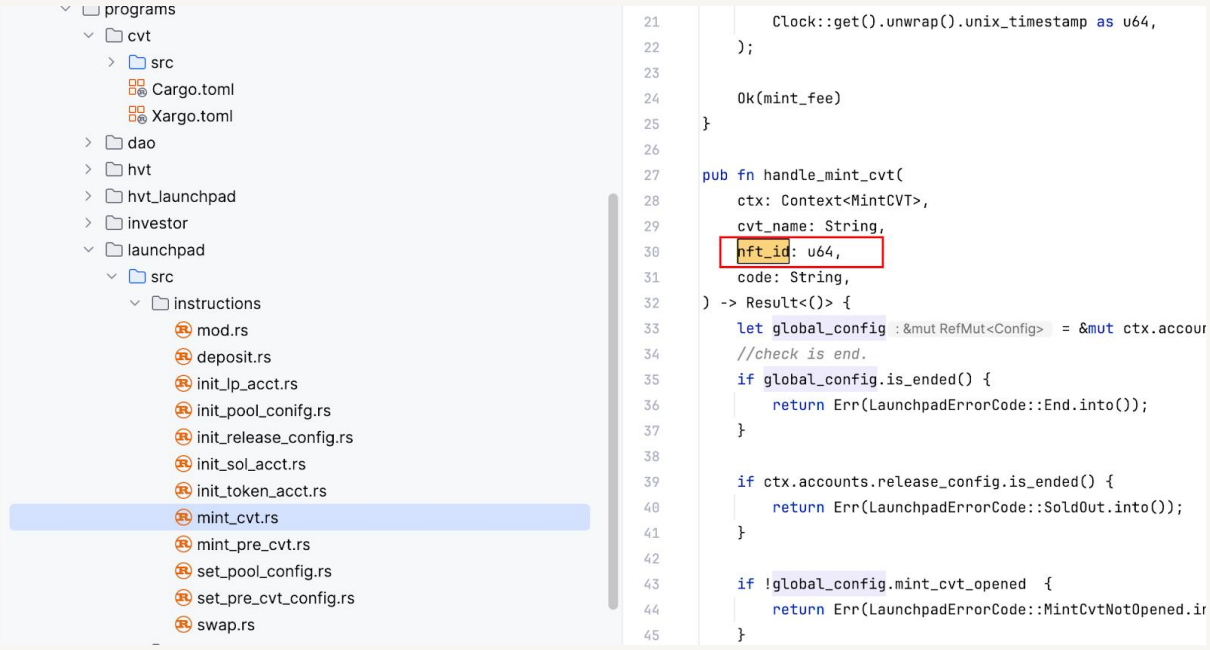
041a25073b115d2922536d428ae5590e9a78ffb9

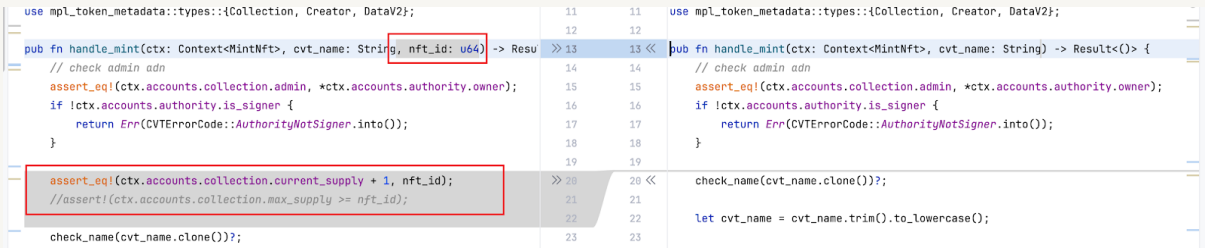
3.5 cvt mint nft_id may not be what you expect

ID:	NVE-05	**Location:**	launchpad/src/instructions/mint_cvt.rs
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Medium

Description:

Currently, mint will specify nft_id in launchpad, but mint has removed this parameter in cvt, which may cause inconsistency between nft id and the specified one.





Recommendations:

Exvul Web3 Security recommends that you determine if the nft_id parameter is changed.

Result: Confirmed

Fix Result: Fixed

nft_id has been deleted:

```
pub fn handle_mint_cvt(ctx: Context<MintCVT>, cvt_name: String, code: String) -> Result<()> {
    let global_config : &mut RefMut<Config> = &mut ctx.accounts.global_config.load_mut()?;
    //check is end.
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

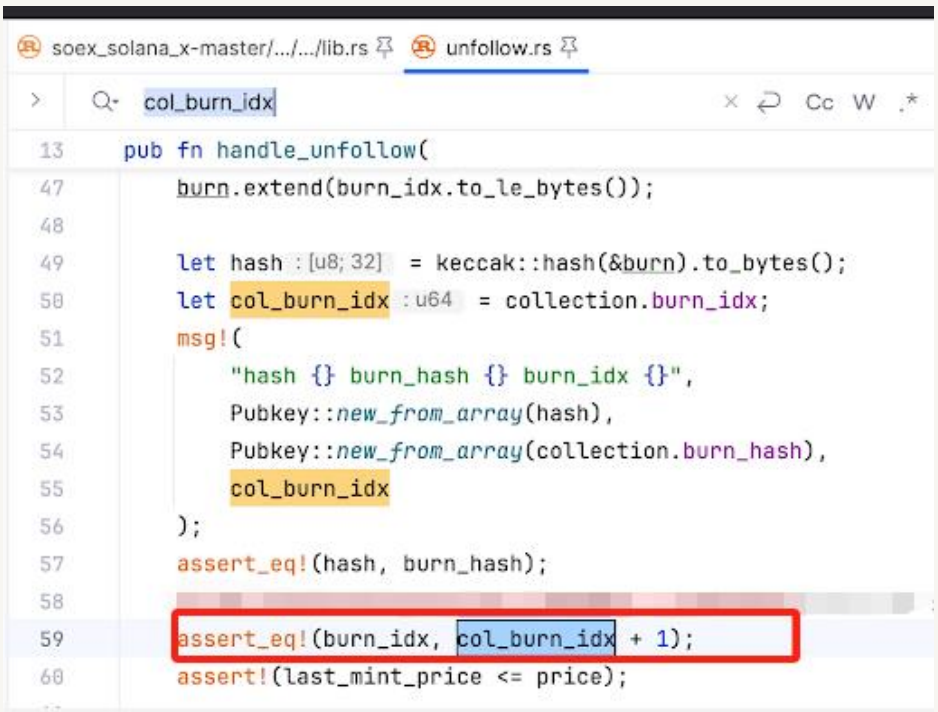
3.6 collection.burn_idx is not updated when unfollow

ID:	NVE-006	Location:	hvt_launchpad/src/instructions/unfollow.rs
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	Medium

Description:

When unfollowing, the signature is used to unfollow, including burn_idx.

The requirement is that burn_idx must be collection.burn_idx +1. The problem is that collection.burn_idx is not updated, so the burn_idx passed in is a fixed value. Passing in a fixed value is meaningless. This should be because collection.burn_idx is not updated.



Recommendations:

Exvul Web3 Security recommends checking burn_idx.

Result: Confirmed

Fix Result: Fixed

Code updated.

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.7 quit Signature does not verify the owner of nft

ID:	NVE-007	Location:	premium_cvt/src/instructions/handle_quit.rs
Severity:	Medium	Category:	Verification Control
Likelihood:	Medium	Impact:	Medium

Description:

When quitting, a signature is required to quit, but when minting, only the emit event is recorded. When quitting, the signature may be obtained by other users.

```
8  use anchor_lang::system_program;
9
10 pub fn handle_quit(ctx: Context<QuitNft>, nft_id: u16, signature: [u8; 64]) -> Result<()> {
11     let collection = &mut ctx.accounts.collection.load_mut()?;
12
13     let index = (nft_id >> 3) as usize;
14     let position = nft_id & 0x07;
15     let bit = (collection.ids[index] >> position) & 1;
16     assert_eq!(bit, 1);
17
18     let mut msg = vec![];
19     msg.extend(ctx.accounts.payer.key().to_bytes());
20     msg.extend(nft_id.to_le_bytes());
21     msg.extend(collection.closed.to_le_bytes());
22
23     let hash = keccak::hash(&msg).to_bytes();
24     msg!("hash {}", Pubkey::new_from_array(hash));
25
26     let ix: Instruction = load_instruction_at_checked(2, &ctx.accounts.ix_sysvar)?;
27     utils::verify_ed25519_ix(&ix, &collection.creater.to_bytes(), &hash, &signature)?;
28
29     let input_amount = collection.price;
```

Recommendations:

Exvul Web3 Security recommends verifying the owner signature of nft.

Result: Confirmed

Fix Result: Fixed

Add judgment:

require!(user_storage.is_nft_owner(nft_id as u32), ErrorCode::InvalidOwner);

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.8 Admin calls handle_verify_cvt_sol_account twice, which may cause DOS transfer

ID:	NVE-008	Location:	premium_cvt/src/instructions/handle_transfer.rs
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	Medium

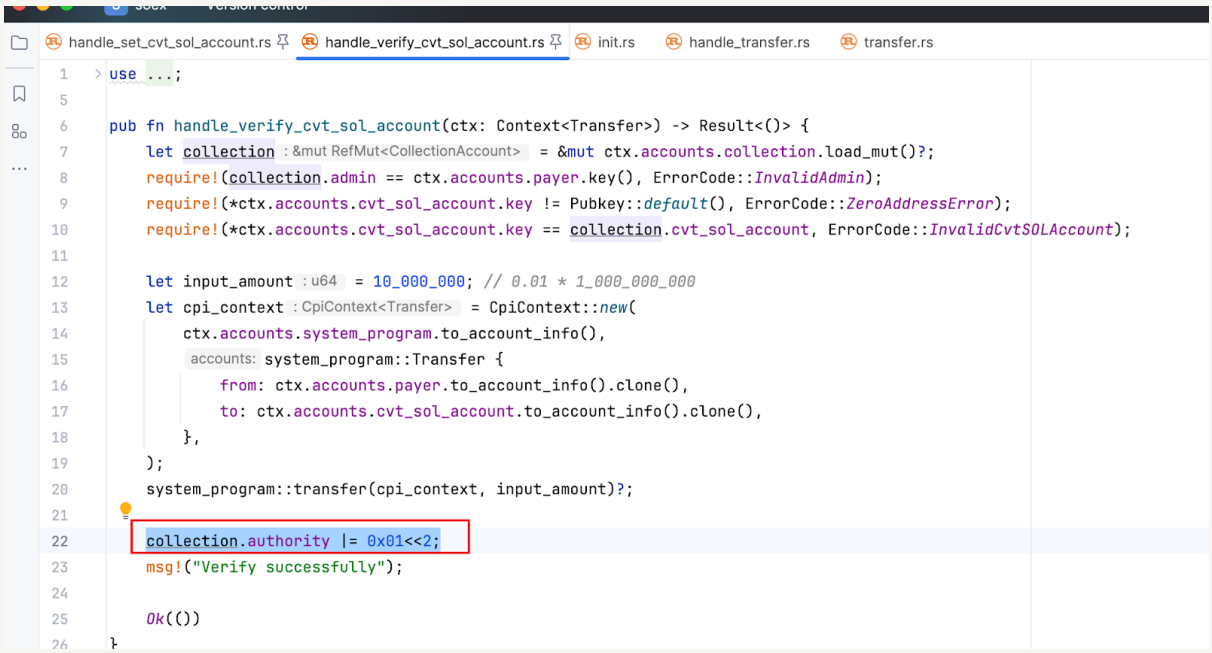
Description:

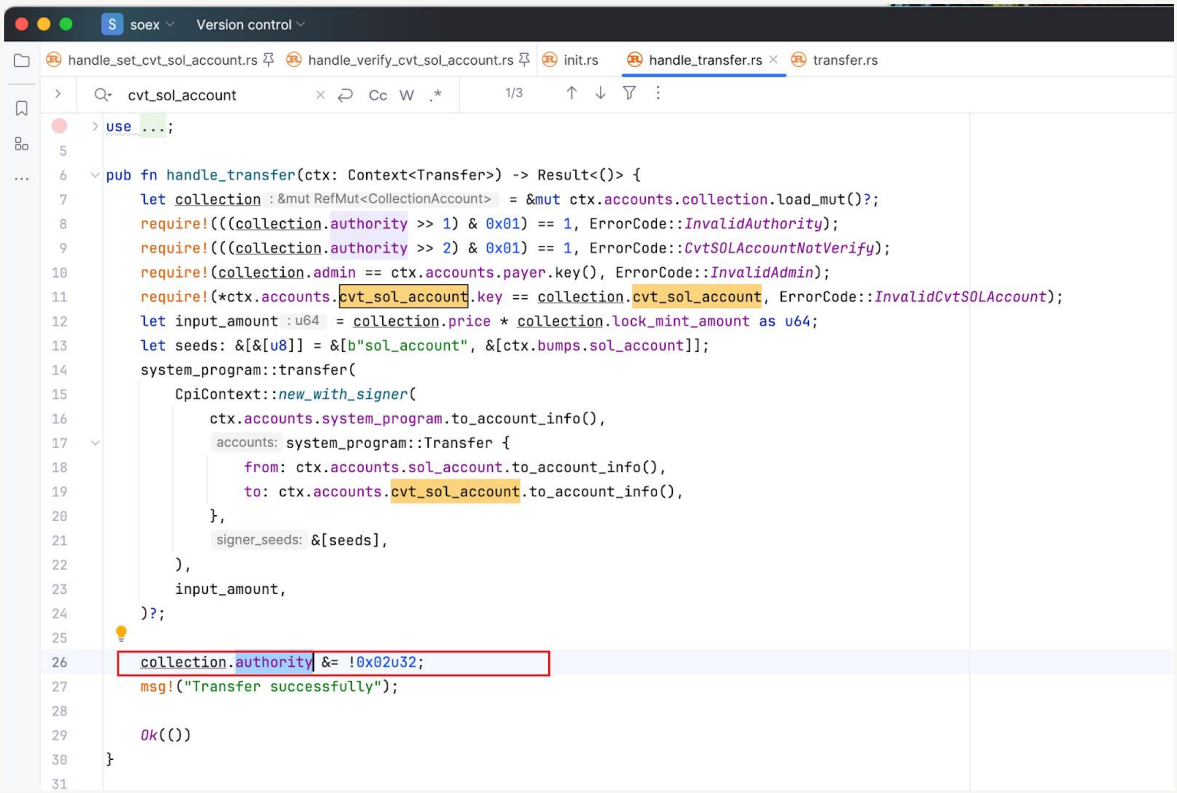
According to the existing code logic, the change of the third flag bit should be:

handle_verify_cvt_sol_account 1

handle_transfer 0 collection.authority &= !0x02u32; clear the flag bit

It should be checked that handle_verify_cvt_sol_account cannot be called twice, otherwise the transfer may fail.





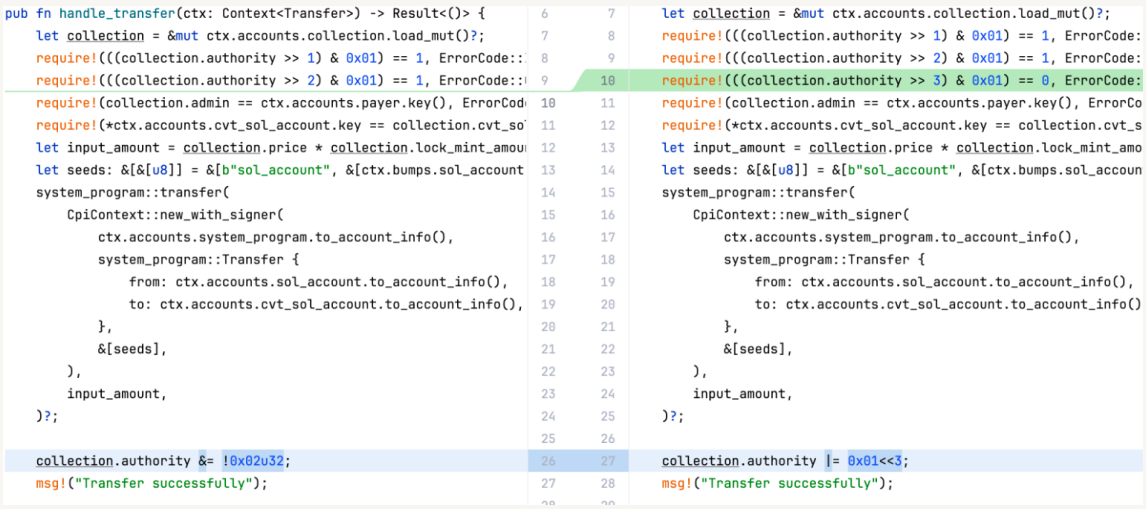
Recommendations:

Exvul Web3 Security recommends checking that handle_verify_cvt_sol_account cannot be called twice.

Result: Confirmed

Fix Result: Fixed

Add judgment:



Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

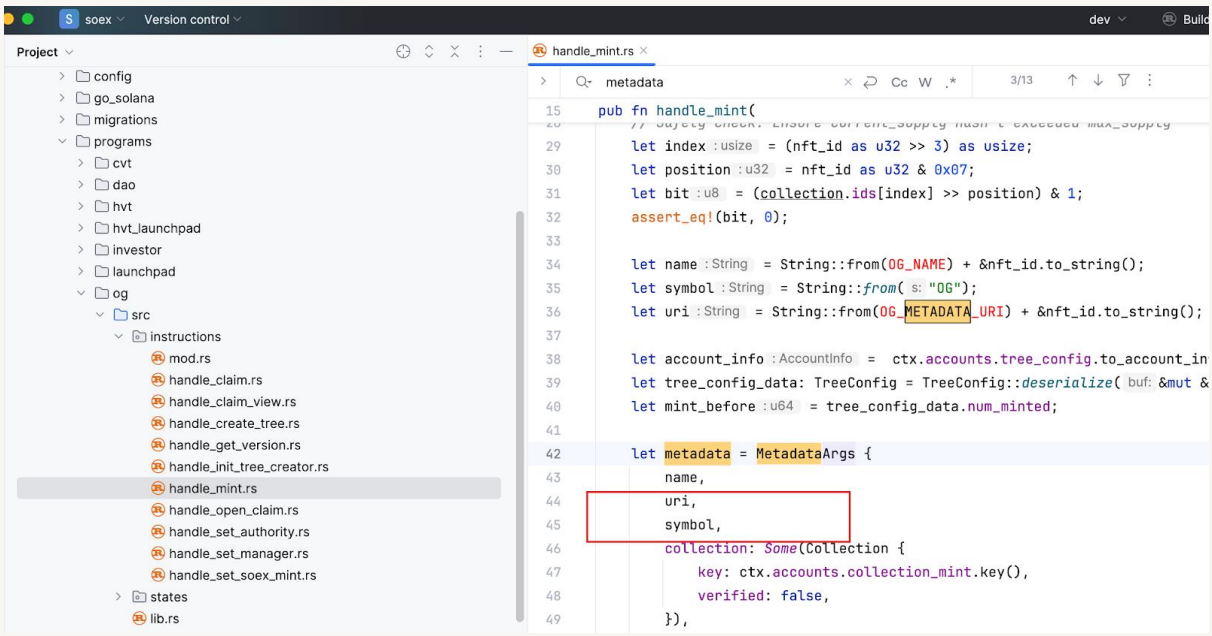
3.9 OG mint parameter configuration is incorrect

ID:	NVE-009	Location:	og/src/instructions/handle_mint.rs
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	Medium

Description:

When OG mint, nft parameters are name, uri, symbol.

However, the order of this construction method is name symbol uri, which will cause parameter confusion.



handle_mint.rs metadata_args.rs

```
This code was AUTOGENERATED using the kinobi library. Please DO NOT EDIT THIS
visitors to add features, then rerun kinobi to update it.

[https://github.com/metaplex-foundation/kinobi]

7
8   use ...;
15
16  #[derive(BorshSerialize, BorshDeserialize, Clone, Debug, Eq, Pa
17  #[cfg_attr(feature = "serde", derive(serde::Serialize, serde::D
18  pub struct MetadataArgs {
        The name of the asset
20      pub name: String,
        The symbol for the asset
22      pub symbol: String,
        URI pointing to JSON representing the asset
24      pub uri: String,
```

Recommendations:

Exvul Web3 Security recommends ensuring parameters are consistent.

Result: Confirmed

Fix Result: Fixed

Correctly modified:

```
42  42      let metadata = MetadataArgs {
43  43          name,
44      -       uri,
45  44          symbol,
    45  +       uri,
46  46          collection: Some(Collection {
47  47              key: ctx.accounts.collection_mint.key(),
48  48              verified: false,
... ...
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.10 Maximum value limit of mint

ID:	NVE-010	Location:	og/src/instructions/handle_mint.rs
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	Medium

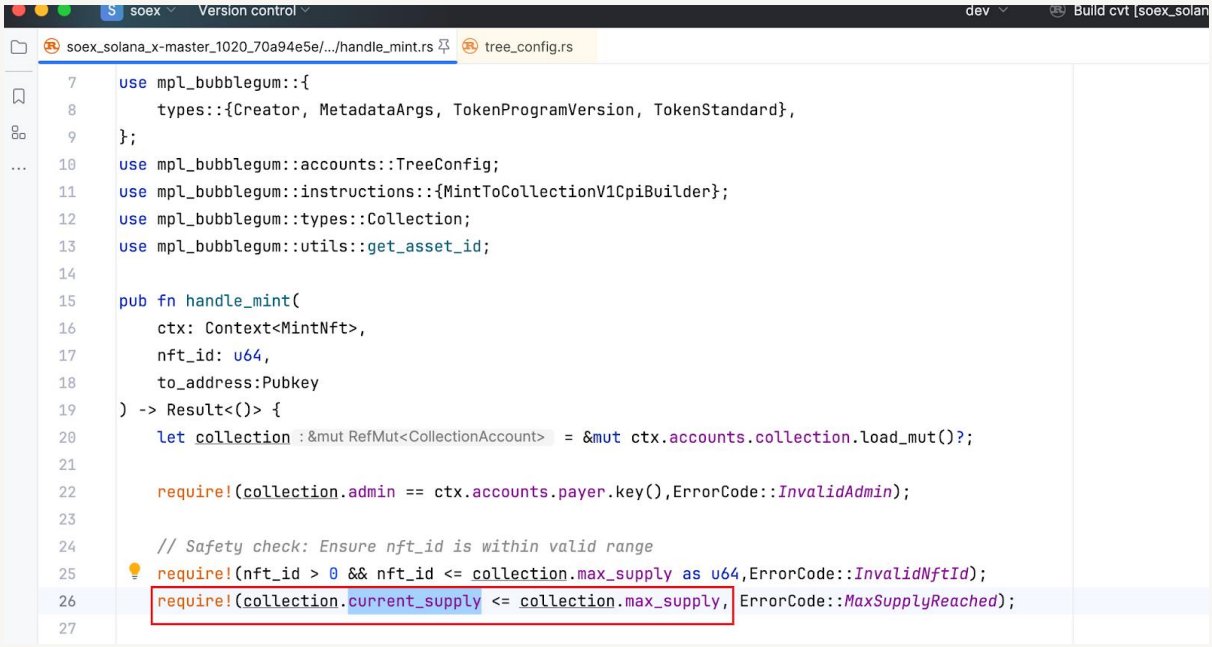
Description:

The maximum value is limited to <=.

If max is 1, the current supply is 1.

require!(collection.current_supply <= collection.max_supply

It can still pass this check, resulting in a mint number of 2 exceeding max.



Recommendations:

Exvul Web3 Security recommends that it should be set to < , otherwise it may over-mint.

Result: Confirmed

Fix Result: Fixed

Correctly modified:

```
24  24      // Safety check: Ensure nft_id is within valid range
25  25      require!(nft_id > 0 && nft_id <= collection.max_supply as u64,ErrorCode::InvalidNftId);
26          require!(collection.current_supply <= collection.max_supply, ErrorCode::MaxSupplyReached);
    26  +   require!(collection.current_supply < collection.max_supply, ErrorCode::MaxSupplyReached);
27  27
28  28      // Safety check: Ensure current_supply hasn't exceeded max_supply
29  29      let index = (nft_id as u32 >> 3) as usize;
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.11 May be preemptively initialized

ID:	NVE-011	Location:	premium_cvt/src/lib.rs
Severity:	Medium	Category:	Verification Control
Likelihood:	Low	Impact:	High

Description:

init is used to design the admin role and may be set first.

```
11
12  #[program]
13  pub mod premium_cvt {
14      use ...;
17
18      pub fn init(
19          ctx: Context<Initialize>,
20          admin: Pubkey,
21          creater: Pubkey,
22          max_supply: u32,
23      ) -> Result<()> {
24          let collection : &mut RefMut<CollectionAccount> = &mut ctx.accounts.collection.load_init()?;
25          collection.admin = admin;
26          collection.max_supply = max_supply;
27          collection.creater = creater;
28          collection.price = 5 * 1_000_000_000;
29          Ok(())
30      }
```

Recommendations:

Exvul Web3 Security recommends setting privileged roles during initialization.

Result: Confirmed

Fix Result: Fixed

Add judgment:

require!(ctx.accounts.payer.key() == ADMIN_PUBKEY, ErrorCode::AdminAccessRequired);

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.12 unfollow price bucket may not be consistent with expectations

ID:	NVE-012	Location:	hvt_launchpad/src/instructions/unfollow.rs
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	Medium

Description:

The unfollow price depends on the previous mint price, which is put into a price bucket. The unfollow price depends on the last price stored in the price bucket. Similar to the previous problem, there is also a slippage problem. The price stored in the price bucket is 5 5.05 5.1 5. If multiple unfollow users cannot determine the specific amount they will receive.

Recommendations:

Exvul Web3 Security recommends adding slippage.

Result: Confirmed

Fix Result: Ignored

Customer responses had little impact.

3.13 Class B HVT No Charge Fixed 0.01 sol

ID:	NVE-013	Location:	hvt_launchpad/src/instructions/follow.rs
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	Medium

Description:

> Each claim requires a 0.01 SOL protocol fee.

According to the documentation, non-sol soex tokens will be charged an additional 0.01 sol fee, but there is no corresponding logic in the code.

```
128         )?;
129     } else {
130         transfer_token_to_pool(
131             &ctx.accounts.token_account,
132             ctx.accounts.cvt_vault_account.to_account_info(),
133             ctx.accounts.mint.to_account_info(),
134             price,
135             &ctx.accounts.token_program,
136             ctx.accounts.payer.to_account_info(),
137             ctx.accounts.mint.decimals,
138         )?;
139 
140         let fee :u64 = ((price as u128 * other_fee as u128) / 10000) as u64;
141 
142         transfer_token_to_pool(
143             &ctx.accounts.token_account,
144             ctx.accounts.mint_account.to_account_info(),
145             ctx.accounts.mint.to_account_info(),
146             fee,
147             &ctx.accounts.token_program,
148             ctx.accounts.payer.to_account_info(),
149             ctx.accounts.mint.decimals,
150         )?;
151     }
```

Recommendations:

Exvul Web3 Security recommends adding a fee.

Result: Confirmed

Fix Result: Fixed

Added fee collection:

```
    let claim_fee : u64  = 1e7 as u64;
    msg!("Claimed fee: {}", claim_fee);
    let cpi_context : CpiContext<Transfer>  = CpiContext::new(
        ctx.accounts.system_program.to_account_info(),
        accounts: system_program::Transfer {
            from: ctx.accounts.payer.to_account_info().clone(),
            to:ctx.accounts.fee_account.to_account_info(),
        },
    );
    system_program::transfer(cpi_context,claim_fee)?;
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.14 Name verification problem

ID:	NVE-014	**Location:**	cvt/src/utils.rs
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Medium

Description:

Name verification problem The domain name may contain spaces or other special characters. When checking the domain name, use trim to determine whether the domain name contains spaces. Trim can only remove the spaces before and after. It cannot guarantee that the entire domain name does not contain spaces.

```
3   pub fn check_name(cvt_name: String) -> Result<()> {
4       if cvt_name != cvt_name.trim().to_lowercase() {
5           msg!("Domain names must be lower case and have no space");
6           return Err(ProgramError::InvalidArgument.into());
7       }
8
9       if cvt_name.contains('.') {
10          return Err(ProgramError::InvalidArgument.into());
11      }
12
13      Ok(())
14  }
```

Recommendations:

Exvul Web3 Security recommends adding domain name restriction verification.

Result: Confirmed

Fix Result: Fixed

Add judgment:

```
pub fn check_name(cvt_name: String) -> Result<()> {
    if cvt_name != cvt_name.trim().to_lowercase() {
        msg!("Domain names must be lower case and have no space");
        return Err(ProgramError::InvalidArgument.into());
    }

    if cvt_name.contains(' ') {
        return Err(ProgramError::InvalidArgument.into());
    }

    if cvt_name.contains('.') {
        return Err(ProgramError::InvalidArgument.into());
    }

    Ok(())
}
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.15 Problem with arbitrary user calling the initialize method

ID:	NVE-015	**Location:**	cvt/src/lib.rs
Severity:	Medium	**Category:**	Verification Control
Likelihood:	Medium	**Impact:**	Medium

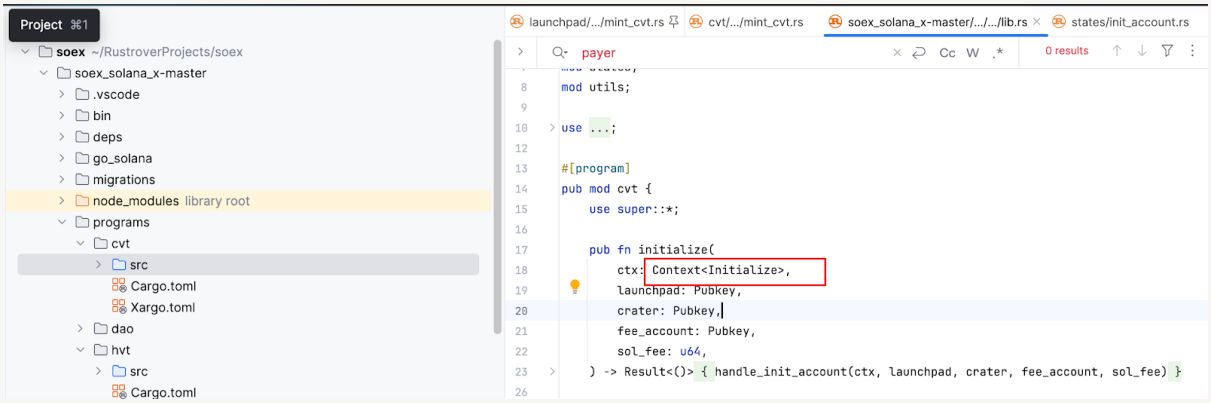
Description:

In the code, there is no logic for permission control, that is, any user can call the initialize method. This is very important because initialization usually involves setting global parameters (such as

administrators, fee accounts, etc.). Without proper permission control, anyone can change these important parameters at will, which may lead to the following problems:

Malicious users may reinitialize the contract and change key data such as administrators and fee accounts.

This will undermine the security of the entire contract, especially if the initialization is related to fund management, which may lead to the transfer of assets or hijacking of control.



Recommendations:

Exvul Web3 Security recommends that in initialize, you should ensure that only deployers or specific administrator roles can call this method.

Result: Confirmed

Fix Result: Fixed

Privileged role requirements have been added:

```
pub fn handle_init_account(
    ctx: Context<Initialize>,
    launchpad: Pubkey,
    creater: Pubkey,
    fee_account: Pubkey,
    sol_fee: u64,
) -> Result<()> {
    require!(ctx.accounts.payer.key() == ADMIN_PUBKEY, CVTErrorCode::AdminAccessRequired);
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.16 Multiplication before division causes precision problems

ID:	NVE-016	Location:	staking_hvt/src/instructions/round.rs
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	Medium

Description:

In the handle_round_create method and calculate_releaseable_amount method, the amount is calculated by multiplication first and then division, which will cause precision problems.

```
pub fn handle_round_create(ctx: Context<RoundCreate>, cvt_name: String, round: u64) -> Result<()> {
    let pool :&mut RefMut<Pool> = &mut ctx.accounts.pool.load_mut()?;
    let current_slot :Slot = ctx.accounts.clock.slot;

    let current_round :&mut RefMut<RewardRound> = &mut ctx.accounts.current_round.load_mut()?;
    let oracle_account :&Box<Account<OracleAccount>> = &ctx.accounts.oracle_account;

    if !ctx.accounts.payer.is_signer {
        return Err(StakeErrorCode::InvalidSigner.into());
    }

    //alweays check the round index
    if pool.current_round_index != round {
        return Err(StakeErrorCode::InvalidRound.into());
    }

    // check if round is already created
    if pool.current_round_index != oracle_account.round_index {
        return Err(StakeErrorCode::InvalidRound.into());
    }

    let amount :u64 = oracle_account.round_reward;
    //:qax
    let release_amount :u64 = amount / 100 * 80;
    let treasury_amount :u64 = amount / 100 * 4;
    let owner_amount :u64 = amount - release_amount - treasury_amount;
```

```
fn calculate_releaseable_amount(
    last_claim_block: u64,
    total_claim: u64,
    release_block: u64,
    total_staked: u64,
    started_block: u64,
) -> u64 {
    let block :Slot = Clock::get().unwrap().slot;
    if block >= release_block {
        return total_staked - total_claim;
    } else {
        let total_block :u64 = release_block - started_block;
        let release_block :Slot = block - last_claim_block;
        let release :u64 = total_staked / total_block * release_block;
        return release;
    }
}
```

The following two locations have the same problem and have been modified:

launchpad/src/instructions/mint_pre_cvt.rs

launchpad/src/instructions/mint_cvt.rs

Recommendations:

Exvul Web3 Security recommends multiplying first and then dividing to enhance the accuracy of calculation.

Result: Confirmed

Fix Result: Fixed

Calculation adjustment is to multiply first and then divide:

```
let release_amount = (amount * 80) / 100 ;
let treasury_amount = (amount * 4) / 100;
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.17 Cancelling the pledge does not update the total amount of pledged

ID:	NVE-017	Location:	staking_hvt/src/instructions/unstake.rs
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	Medium

Description:

Staking_hvt module, canceling staking does not update the total staking amount.

```
pub fn handle_unstake(
    ctx: Context<Unstake>,
    cvt_name: String,
    hvt_name: String,
    round: u64,
) -> Result<()> {
    let mut pool = ctx.accounts.pool.load_mut()?;
    let current_slot = ctx.accounts.clock.slot;
    assert_eq!(pool.current_round_index, round);

    let stake = &mut ctx.accounts.stake;
    //check my stake account amount > 0
    if !stake.is_stake {
        return Err(StakeErrorCode::StakeAmountIsZero.into());
    }

    let mut round = ctx.accounts.current_round.load_mut()?;
    let maybe_roundup = round.accum_reward(current_slot);
    let current_round = pool.current_round_index;

    let my_stake = &mut ctx.accounts.stake;
    assert_eq!(my_stake.round_index, current_round);
    // update user earn
    my_stake.reward += my_stake.stake * (round.accum_reward - my_stake.accum_reward);
    if current_slot > round.end_slot {
        my_stake.last_update_slot = round.end_slot;
    } else {
        my_stake.last_update_slot = current_round;
    }

    my_stake.accum_reward = round.accum_reward;
```

Recommendations:

Exvul Web3 Security recommends updating the total amount of stake in time after canceling the stake.

Result: Confirmed

Fix Result: Fixed

Added unstaking function to update the total amount of stake:

```
pool.total_staked -= 1;
round.total_staked -= 1;
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.18 The status of the unstaking was not updated in time

ID:	NVE-018	**Location:**	staking_lp/src/instructions/unstake.rs
Severity:	Medium	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Medium

Description:

The staking_lp module updates its staking status to is_stake = true when staking, but after canceling the pledge, the staking status is not updated to is_stake = false, so the staking status has always been is_stake = true.

```rust
pub fn handle_unstake(ctx: Context<Unstake>, round: u64, amount: u64) -> Result<()> {
    let pool = &mut ctx.accounts.pool.load_mut()?;
    let current_slot = ctx.accounts.clock.slot;
    msg!(
        "handle_unstake current_round_index {} round {}",
        pool.current_round_index,
        round
    );
    assert_eq!(pool.current_round_index, round);

    let round = &mut ctx.accounts.current_round.load_mut()?;
    let maybe_roundup = round.accum_reward(current_slot);
    let current_round = pool.current_round_index;
    msg!("handle_unstake is_stake {}", ctx.accounts.my_stake.is_stake);
    assert!(ctx.accounts.my_stake.is_stake);

    let my_stake = &mut ctx.accounts.my_stake;
    msg!(
        "handle_unstake stake {} round accum_reward {} my_stake accum_reward {} {}",
        my_stake.stake,
        round.accum_reward,
        my_stake.accum_reward,
        my_stake.stake as u128 * (round.accum_reward - my_stake.accum_reward) / 1e9 as u128
    );
    assert_eq!(my_stake.round_index, current_round);
    assert_eq!(my_stake.round_index, round.round_index);
```

Recommendations:

Exvul Web3 Security recommends updating the status after canceling the stake.

Result: Confirmed

Fix Result: Fixed

Added status updates after unstaking:

```rust
if my_stake.stake == 0 {
    //when withdraw all also claim reward
    my_stake.is_stake = false;
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.19 There is no fixed logic of charging 0.01 sol when claiming LP rewards

ID:	NVE-019	Location:	staking_lp/src/instructions/stake.rs
Severity:	Medium	Category:	Business Issues
Likelihood:	Medium	Impact:	Medium

Description:

According to the documentation, a fixed amount of 0.01 sol is charged when claiming LP rewards, but there is no logic in the code to charge 0.01 sol.

> Each claim requires a 0.01 SOL protocol fee.

Recommendations:

Exvul Web3 Security recommends adding fee logic.

Result: Confirmed

Fix Result: Fixed

Fee logic has been added:

```
let claim_fee = 1e7 as u64;
msg!("Claimed fee: {}", claim_fee);
let cpi_context = CpiContext::new(
    ctx.accounts.system_program.to_account_info(),
    system_program::Transfer {
        from: ctx.accounts.payer.to_account_info().clone(),
        to:ctx.accounts.fee_account.to_account_info(),
    },
);
system_program::transfer(cpi_context,claim_fee)?;
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.20 Stake and unstake cannot be frozen

ID:	NVE-020	Location:	staking_lp/src/instructions/unstake.rs
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Medium

Description:

```
pub fn switch_freezed(ctx: Context<SwitchFreezed>, freezed: bool) -> Result<()> {
    handle_switch_freezed(ctx, freezed)
}
```

Freeze is set but no check is done during actual staking and unstaking, which results in the inability to pause staking and unstaking in the contract.

```rust
pub fn handle_stake(ctx: Context<Stake>, round: u64, amount: u64) -> Result<()> {
    let mut pool = ctx.accounts.pool.load_mut()?;
    let current_slot = ctx.accounts.clock.slot;
    assert_eq!(pool.current_round_index, round);

    let mut round = ctx.accounts.current_round.load_mut()?;
    let maybe_roundup = round.accum_reward(current_slot);

    transfer_token_to_pool(
        &ctx.accounts.user_lp_account,
        ctx.accounts.my_lp_valut.to_account_info(),
        ctx.accounts.lp_mint.to_account_info(),
        amount,
        &ctx.accounts.token_program,
        ctx.accounts.payer.to_account_info(),
        ctx.accounts.lp_mint.decimals,
    )?;

    let current_round = pool.current_round_index;
    if ctx.accounts.my_stake.is_stake {
        let my_stake = &mut ctx.accounts.my_stake;
        assert_eq!(my_stake.round_index, current_round);

        // update user earn
        my_stake.reward += ((my_stake.stake as u128)
            * (round.accum_reward - my_stake.accum_reward) as u128
            / 1e9 as u128) as u64;
        my_stake.last_update_slot = current_slot;
        my_stake.accum_reward = round.accum_reward;

        //add stake
        my_stake.stake += amount;
    } else {
        let my_stake = &mut ctx.accounts.my_stake;
        //First time stake
        my_stake.round_index = current_round;
        my_stake.is_stake = true;
        my_stake.last_update_slot = current_slot;
        my_stake.accum_reward = round.accum_reward;

        my_stake.stake = amount;
    }

    pool.total_staked += amount;
    round.total_staked += amount;
```

```rust
pub fn handle_unstake(ctx: Context<Unstake>, round: u64, amount: u64) -> Result<()> {
    let pool = &mut ctx.accounts.pool.load_mut()?;
    let current_slot = ctx.accounts.clock.slot;
    msg!(
        "handle_unstake current_round_index {} round {}",
        pool.current_round_index,
        round
    );
    assert_eq!(pool.current_round_index, round);

    let round = &mut ctx.accounts.current_round.load_mut()?;
    let maybe_roundup = round.accum_reward(current_slot);
    let current_round = pool.current_round_index;
    msg!("handle_unstake is_stake {}", ctx.accounts.my_stake.is_stake);
    assert!(ctx.accounts.my_stake.is_stake);

    let my_stake = &mut ctx.accounts.my_stake;
    msg!(
        "handle_unstake stake {} round accum_reward {} my_stake accum_reward {} {}",
        my_stake.stake,
        round.accum_reward,
        my_stake.accum_reward,
        my_stake.stake as u128 * (round.accum_reward - my_stake.accum_reward) / 1e9 as u128
    );
    assert_eq!(my_stake.round_index, current_round);
    assert_eq!(my_stake.round_index, round.round_index);

    // update user earn
    my_stake.reward += ((my_stake.stake as u128)
        * (round.accum_reward - my_stake.accum_reward) as u128
        / 1e9 as u128) as u64;

    if current_slot > round.end_slot {
        my_stake.last_update_slot = round.end_slot;
    } else {
        my_stake.last_update_slot = current_round;
    }

    my_stake.accum_reward = round.accum_reward;
    my_stake.stake -= amount;

    transfer_token_from_pool(
        &ctx.accounts.my_lp_valut,
        ctx.accounts.user_lp_account.to_account_info(),
        ctx.accounts.lp_mint.to_account_info(),
        amount,
        &ctx.accounts.token_program,
        &ctx.accounts.global_account.to_account_info(),
        ctx.accounts.lp_mint.decimals,
        ctx.bumps.global_account,
    )?;

    if maybe_roundup.is_some() && (current_round == round.round_index) {
        pool.current_round_index += 1;
```

Recommendations:

Exvul Web3 Security recommends adding a pause mechanism.

Result: Confirmed

Fix Result: Fixed

Added freezed judgment:

```
pub fn handle_unstake(ctx: Context<Unstake>, round: u64, amount: u64) -> Result<()> {
    let pool = &mut ctx.accounts.pool.load_mut()?;
    assert_eq!(pool.freezed, 0);
    let current_slot = ctx.accounts.clock.slot;
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.21 Stake and unstake do not check if amount is not 0

ID:	NVE-021	Location:	staking_lp/src/instructions/stake.rs
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

Stake and unstake do not determine whether the amount is not 0, which means that the amount is 0 and can be staked and unstaked.

```rust
pub fn handle_stake(ctx: Context<Stake>, round: u64, amount: u64) -> Result<()> {
    let mut pool = ctx.accounts.pool.load_mut()?;
    let current_slot = ctx.accounts.clock.slot;
    assert_eq!(pool.current_round_index, round);

    let mut round = ctx.accounts.current_round.load_mut()?;
    let maybe_roundup = round.accum_reward(current_slot);

    transfer_token_to_pool(
        &ctx.accounts.user_lp_account,
        ctx.accounts.my_lp_valut.to_account_info(),
        ctx.accounts.lp_mint.to_account_info(),
        amount,
        &ctx.accounts.token_program,
        ctx.accounts.payer.to_account_info(),
        ctx.accounts.lp_mint.decimals,
    )?;

    let current_round = pool.current_round_index;
    if ctx.accounts.my_stake.is_stake {
        let my_stake = &mut ctx.accounts.my_stake;
        assert_eq!(my_stake.round_index, current_round);

        // update user earn
        my_stake.reward += ((my_stake.stake as u128)
            * (round.accum_reward - my_stake.accum_reward) as u128
            / 1e9 as u128) as u64;
        my_stake.last_update_slot = current_slot;
        my_stake.accum_reward = round.accum_reward;

        //add stake
        my_stake.stake += amount;
    } else {
        let my_stake = &mut ctx.accounts.my_stake;
        //First time stake
        my_stake.round_index = current_round;
        my_stake.is_stake = true;
        my_stake.last_update_slot = current_slot;
        my_stake.accum_reward = round.accum_reward;

        my_stake.stake = amount;
    }

    pool.total_staked += amount;
    round.total_staked += amount;
```

Recommendations:

Exvul Web3 Security recommends that the amount is not 0.

Result: Confirmed

Fix Result: Ignored

3.22 Any caller can preemptively modify is_open_sol_reward_pool

ID:	NVE-022	Location:	staking_lp/src/instructions/handle_open_sol_reward.rs
Severity:	Low	Category:	Verification Control
Likelihood:	Low	Impact:	Medium

Description:

Since there is no permission check, any caller can preemptively modify is_open_sol_reward_pool.

```
use crate::states::*;
use anchor_lang::prelude::*;

pub fn handle_open_sol_reward(ctx: Context<OpenSolReward>, is_open_sol_reward_pool:u64) ->Result<()> {
    let pool = &mut ctx.accounts.pool.load_init()?;
    pool.is_open_sol_reward_pool = is_open_sol_reward_pool;

    Ok(())
}
```

Recommendations:

Exvul Web3 Security recommends adding permission verification.

Result: Confirmed

Fix Result: Fixed

Added role judgment:

require!(ctx.accounts.payer.key().to_string() ==

ADMIN_PUBKEY, StakeErrorCode::AdminAccessRequired);

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.23 Test environment configuration

ID:	NVE-023	Location:	hvt/src/config.rs
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Medium

Description:

The current test environment configuration is recommended to be changed to the formal environment.

config.rs

programs > hvt > src > config.rs

```
1  pub const HVT_METADATA_URI: &str = "https://dev-cvt-nft.soex.io/hvt/";
2
```

Recommendations:

Exvul Web3 Security recommends changing to a formal environment for online use.

Result: Confirmed

Fix Result: Fixed

Modified to the official environment

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.24 Spelling Errors

ID:	NVE-024	Location:	cvt/src/lib.rs
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

There are multiple spelling errors in the contract.

Other

```
#[derive(Default, Debug)]
pub struct CollectionAccount {
    pub cvt_mint_key: Pubkey,
    pub token_mint: Pubkey,

    pub burn_idx: u64,
    pub burn_hash: [u8; 32],

    pub price: u64,
    pub tvl: u64,

    pub last_mint_price: u64,
    pub last_mint_time: u64,
    pub current_supply: u64,
    pub max_supply: u64,
    pub fee_mode: u64,
    pub when_price_go_up: u64,
    pub ohther_fee: u64,
}
```

Wsols

```
msg!("Transfer WSOL to contract account");
transfer_token_to_pool(
    &ctx.accounts.user_sol_account,
    ctx.accounts.sol_account.to_account_info(),
    ctx.accounts.sol_token.to_account_info(),
    mint_fee,
    &ctx.accounts.token_program,
    ctx.accounts.signer.to_account_info(),
    ctx.accounts.sol_token.decimals,
)?;
```

There are typos in both check_owner and startd_block.

Recommendations:

Exvul Web3 Security recommends correcting the spelling errors.

Result: Confirmed

Fix Result: Fixed

All typos have been corrected.

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.25 Transfer unverified user balance

ID:	NVE-025	Location:	staking_lp/src/instructions/stake.rs
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

The staking_lp module does not verify whether the user has sufficient funds. If the amount is greater than the user's actual balance, the transfer may fail, which is not handled in the code.

```
pub fn handle_stake(ctx: Context<Stake>, round: u64, amount: u64) -> Result<()> {
    let mut pool = ctx.accounts.pool.load_mut()?;
    let current_slot = ctx.accounts.clock.slot;
    assert_eq!(pool.current_round_index, round);

    let mut round = ctx.accounts.current_round.load_mut()?;
    let maybe_roundup = round.accum_reward(current_slot);

    transfer_token_to_pool(
        &ctx.accounts.user_lp_account,
        ctx.accounts.my_lp_valut.to_account_info(),
        amount,
        &ctx.accounts.token_program,
        ctx.accounts.payer.to_account_info(),
    )?;
```

Recommendations:

Exvul Web3 Security recommends adding a check if the amount is greater than the user's actual balance.

Result: Confirmed

Fix Result: Ignored

3.26 Unfollow uses price buckets for all fee modes

ID:	NVE-026	Location:	hvt_launchpad/src/instructions/unfollow.rs
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

Price buckets are for price changes, unfollow prices are different. Fixed price mode is unnecessary and wastes gas.

Recommendations:

Exvul Web3 Security recommends changing the fixed-price model.

Result: Confirmed

Fix Result: Ignored

3.27 Any user preemptively initializes

ID:	NVE-027	Location:	staking_lp/src/instructions/init_pool.rs
Severity:	Low	Category:	Verification Control
Likelihood:	Low	Impact:	Low

Description:

In the staking_lp module, any user can call handle_init_pool or a similar initialization function and create global_account and pool. Once the accounts are initialized, other users can no longer initialize these accounts, but key configurations of the initial accounts may have been set by accidental or malicious users.

```
pub fn handle_init_pool(
    ctx: Context<InitPool>,
    oracle: Pubkey,
    soex: Pubkey,
    lp: Pubkey,
    freezed: bool,
) -> Result<()> {
    let pool = &mut ctx.accounts.pool.load_init()?;

    pool.manager = *ctx.accounts.payer.key;
    pool.freezed = if freezed { 1 } else { 0 };
    pool.round_slot = STAKING_LP_ROUND_SLOT;
    pool.current_round_index = 0;
    pool.total_staked = 0;
    pool.oracle = oracle;
    pool.lp_mint = lp;
    pool.soex_mint = soex;

    Ok(())
}
```

Recommendations:

Exvul Web3 Security recommends setting privileged roles to call the initialization method..

Result: Confirmed

Fix Result: Fixed

Added privileged role restrictions:

```
pub fn handle_init_pool(
    ctx: Context<InitPool>,
    oracle: Pubkey,
    soex: Pubkey,
    lp: Pubkey,
    freezed: bool,
) -> Result<()> {
    require!(ctx.accounts.payer.key() == ADMIN_PUBKEY, StakeErrorCode::AdminAccessRequired);
    let pool = &mut ctx.accounts.pool.load_init()?;
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.28 Calculation accuracy issues

ID:	NVE-028	**Location:**	staking_lp/src/instructions/round_check.rs
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

Description:

staking_lp module, handle_round_create method,

current_round.reward_per_slot = release_amount / pool.round_slot;

There is a potential calculation accuracy problem here. When release_amount or pool.round_slot is not an integer multiple, rounding errors will occur.

```
let release_amount = ctx.accounts.soex_account.amount / 100;

current_round.round_index = round;
current_round.end_slot = current_slot + pool.round_slot;
current_round.start_slot = current_slot;
current_round.accum_reward = 0;
current_round.last_update_slot = current_slot;
current_round.release = release_amount;
current_round.total_staked = pool.total_staked;
current_round.reward_per_slot = release_amount / pool.round_slot;
```

Recommendations:

Exvul Web3 Security recommends ensuring that the precision error is within an acceptable range or increasing the calculation precision.

Result: Confirmed

Fix Result: Ignored

The customer replied that this problem would not occur in real scenarios.

3.29 Check owner does not work

ID:	NVE-029	Location:	hvt/src/lib.rs
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

This function only checks the validity of the name, and does not check the subordination and existence of cvt and hvt.

```
54      pub fn burn(
55          ctx: Context<BurnHvt>,
56          cvt_name: String,
57          hvt_name: String,
58          price: u64,
59          last_mint_price: u64,
60          burn_idx: u64,
61          burn_hash: [u8; 32],
62      ) -> Result<()> {
63          handle_burn(ctx, cvt_name, hvt_name, price, last_mint_price, burn_idx,
64      }
65
66      pub fn check_oner(
67          ctx: Context<CheckOwner>,
68          cvt_name: String,
69          hvt_name: String,
70          payload: Vec<u8>,
71      ) -> Result<()> {
72          handle_check_owner(ctx, cvt_name, hvt_name, payload)
73      }
74  }
```

Recommendations:

Exvul Web3 Security recommends checking the affiliation and existence of cvt and hvt.

Result: Confirmed

Fix Result: Ignored

3.30 The handle_init method has a preemptive initialization problem

ID:	NVE-030	Location:	hvt/src/config.rs
Severity:	Low	Category:	Verification Control
Likelihood:	Low	Impact:	Low

Description:

The handle_init method has no restrictions on calls, so there is a possibility of preemptive initialization after deployment.

```
use anchor_lang::prelude::*;
use crate::errors::HvtErrorCode;
use crate::states::*;

pub fn handle_init(ctx: Context<Init>, launchpad: Pubkey) -> Result<()> {
    require!((ctx.accounts.global_config.inited & 1) == 0, HvtErrorCode::AlreadyInited);
    ctx.accounts.global_config.inited |= 1;

    ctx.accounts.global_config.admin = *ctx.accounts.payer.key;
    ctx.accounts.global_config.launchpad = launchpad;
    Ok(())
}
```

Recommendations:

Exvul Web3 Security recommends adding permission control.

Result: Confirmed

Fix Result: Fixed

Added privileged role restrictions:

```
pub fn handle_init(ctx: Context<Init>, launchpad: Pubkey) -> Result<()> {
    require!(ctx.accounts.payer.key() == ADMIN_PUBKEY,HvtErrorCode::AdminAccessRequired);
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.31 No judgment admin privilege role is not a zero address

ID:	NVE-031	Location:	premium_cvt/src/instructions/handle_set_manager.rs
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

It is not judged that the admin privilege role is not a zero address. If admin authority is unexpectedly set as a zero address by an administrator, it will cause Admin's privilege role to fail.

```
pub fn handle_set_manager(ctx: Context<SetManager>, manager: Pubkey) -> Result<()>
    if ctx.accounts.signer.is_signer == false {
        return Err(ErrorCode::InvalidSigner.into());
    }

    let global_config = &mut ctx.accounts.global_config.load_mut()?;
    global_config.admin = manager;

    emit!(NewManagerEvent {
        old_manager: ctx.accounts.signer.key(),
        new_manager: manager
    });

    Ok(())
}
```

The same problem exists in the following locations, which have been modified:

og/src/instructions/handle_set_manager.rs
treasury/src/instructions/handle_set_manager.rs

Recommendations:

EXVUL Web3 Security Recomons adds a judgment that the admin privilege role is not zero -address in handle_set_manager.

Result: Confirmed

Fix Result: Fixed

Add event:

require!(Pubkey::default() != manager,ErrorCode::ZeroAddressError);

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.32 cvt_sol_account setting error will not be executed later

ID:	NVE-032	Location:	premium_cvt/src/instructions/handle_set_cvt_sol_account.rs
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

The cvt_sol_account account can only be set once. If the cvt_sol_account is set incorrectly, subsequent transfers will no longer be safe.

```
21  pub fn handle_set_cvt_sol_account(ctx: Context<SetCvtSolAccount>, cvt_sol_account: Pubkey) -> Result<()> {
22      require!(Pubkey::default() != cvt_sol_account,ErrorCode::ZeroAddressError);
23
24      if ctx.accounts.signer.is_signer == false {
25          return Err(ErrorCode::InvalidSigner.into());
26      }
27
28      let global_config = &mut ctx.accounts.global_config.load_mut()?;
29      require!(global_config.cvt_sol_account==Pubkey::default(), ErrorCode::AlreadySet);
30
31      global_config.cvt_sol_account = cvt_sol_account;
32      global_config.authority &= !(0x01 << 2);
33
34      msg!("set_cvt_sol_account {} {}", global_config.authority, cvt_sol_account);
35
36      Ok(())
37  }
```

Recommendations:

Exvul Web3 Security recommends adding dynamic settings in handle_set_cvt_sol_account. If cvt_sol_account is set incorrectly, it can be updated. In order to avoid callers abusing the handle_set_cvt_sol_account method, it is recommended to add a pause mechanism or a cvt_sol_account whitelist mechanism in the handle_set_cvt_sol_account method; without modifying the code, the caller is required to set the account accurately.

Result: Confirmed

Fix Result: Ignored

Customer response: You can consider reserving an interface set by the owner.

3.33 Check if the claim has been opened, otherwise the start slot time may be reset

ID:	NVE-033	Location:	og/src/instructions/handle_open_claim.rs
Severity:	Low	Category:	Verification Control
Likelihood:	Low	Impact:	Low

Description:

When opening a claim, it is recommended to add an on/off switch to check whether the claim has been opened, otherwise the start slot time may be reset.

```
5   pub fn handle_set_open_claim(
6       ctx: Context<OpenClaim>,
7   ) -> Result<()> {
8       if ctx.accounts.signer.is_signer == false {
9           return Err(ErrorCode::InvalidSigner.into());
10      }
11
12      let global_config = &mut ctx.accounts.global_config.load_mut()?;
13
14      global_config.claim_start_slot = ctx.accounts.clock.slot;
15      global_config.authority |= 0x01;
16
17      msg!("set_claim_config {}",ctx.accounts.clock.slot);
18
19      Ok(())
20  }
```

Recommendations:

Exvul Web3 Security recommends adding on/off when opening claim.

Result: Confirmed

Fix Result: Fixed

The judgment has been added and can only be modified once:

require!(global_config.claim_start_slot==0, ErrorCode::AlreadySet);

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.34 Computational issues

ID:	NVE-034	Location:	hvt_launchpad/src/instructions/follow.rs
Severity:	Informational	Category:	Business Issues
Likelihood:	Low	Impact:	Informational

Description:

There are two types of fees when following. The first is sol and soex. If the price is less than 20, there is no fee. If the calculated result is 98, there is no fee. The second is that if the multiplication result of price and other_fee is less than 10000, there is no fee. It is recommended to improve the calculation accuracy for both calculations.

```
92      let fee = price * 5 / 100;
93      // 50 / 30 / 20
94      let (creator_fee, burn_fee, hvt_fee, treasury_fee) = calculate_hvt_fee(fee);
95      hvt_share_fee = hvt_fee;
96      msg!("follow {} {} {} {}", creator_fee, burn_fee, hvt_fee, treasury_fee);
97      assert_eq!(creator_fee + burn_fee + hvt_fee + treasury_fee, fee);
```

```
133     transfer_token_to_pool(
134         &ctx.accounts.token_account,
135         ctx.accounts.cvt_vault_account.to_account_info(),
136         ctx.accounts.mint.to_account_info(),
137         price,
138         &ctx.accounts.token_program,
139         ctx.accounts.payer.to_account_info(),
140         ctx.accounts.mint.decimals,
141     )?;
142
143     let fee = ((price as u128 * other_fee as u128) / 10000) as u64;
144
```

Recommendations:

Exvul Web3 Security recommends improving calculation accuracy.

Result: Confirmed

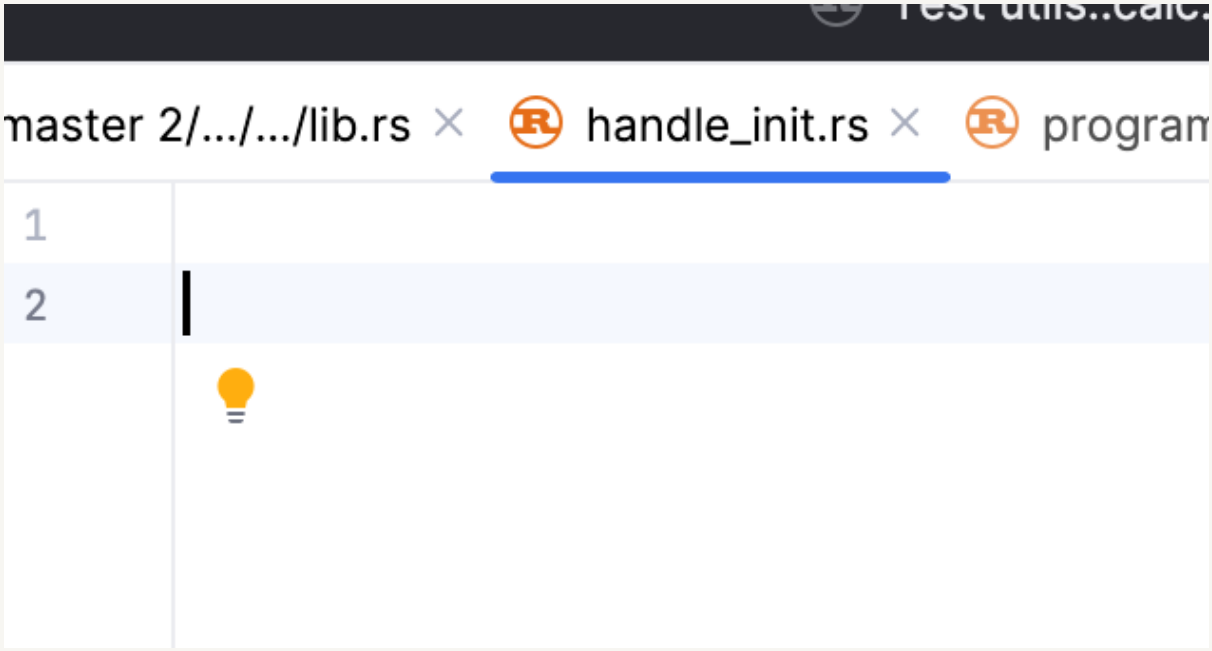
Fix Result: Ignored

3.35 handle_init is empty

ID:	NVE-035	Location:	og/src/lib.rs
Severity:	Informational	Category:	Business Issues
Likelihood:	Low	Impact:	Informational

Description:

In OG, handle_init is empty and is not actually called. Consider deleting it.



The same problem exists in the following locations, which have been modified:

premium_cvt/src/lib.rs

Recommendations:

Exvul Web3 Security recommends removing meaningless code.

Result: Confirmed

Fix Result: Fixed

Code has been deleted.

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.36 cvt mint no events

ID:	NVE-036	Location:	cvt/src/instructions/mint_cvt.rs
Severity:	Informational	Category:	Business Issues
Likelihood:	Low	Impact:	Informational

Description:

As shown in the figure, compared with OG's mint, cvt mint does not have the corresponding emit event. If you interact by listening to events, it may cause some problems.

```
83
84      create_master_edition_v3(
85          CpiContext::new_with_signer(
86              ctx.accounts.metadata_program.to_account_info(),
87              CreateMasterEditionV3 {
88                  edition: ctx.accounts.master_edition_account.to_account_info(),
89                  payer: ctx.accounts.payer.to_account_info(),
90                  mint: ctx.accounts.mint.to_account_info(),
91                  metadata: ctx.accounts.nft_metadata.to_account_info(),
92                  mint_authority: ctx.accounts.payer.to_account_info(),
93                  update_authority: ctx.accounts.payer.to_account_info(),
94                  system_program: ctx.accounts.system_program.to_account_info(),
95                  token_program: ctx.accounts.token_program.to_account_info(),
96                  rent: ctx.accounts.rent.to_account_info(),
97              },
98              &[&seeds[..]],
99          ),
100         Some(1),
101     )?;
102     msg!("Minted NFT successfully");
103
104     ctx.accounts.collection.current_supply += 1;
105
106     Ok(())
107 }
```

Recommendations:

Exvul Web3 Security recommends adding event logging for important information.

Result: Confirmed

Fix Result: Fixed

Added nft emit event:

```
106
107     emit!(MintCvtEvent {
108         who: ctx.accounts.payer.key(),
109         mint: ctx.accounts.mint.key(),
110         cvt_name,
111     });
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.37 hvt mint no events

ID:	NVE-037	Location:	hvt/src/instructions/mint.rs
Severity:	Informational	Category:	Business Issues
Likelihood:	Low	Impact:	Informational

Description:

As shown in the figure, compared with OG's mint, hvt mint does not have the corresponding emit event. If you interact by listening to events, it may cause some problems.

```
98      create_master_edition_v3(
99          CpiContext::new_with_signer(
100             ctx.accounts.metadata_program.to_account_info(),
101             CreateMasterEditionV3 {
102                 edition: ctx.accounts.master_edition_account.to_account_info(),
103                 payer: ctx.accounts.payer.to_account_info(),
104                 mint: ctx.accounts.hvt_mint.to_account_info(),
105                 metadata: ctx.accounts.nft_metadata.to_account_info(),
106                 mint_authority: ctx.accounts.payer.to_account_info(),
107                 update_authority: ctx.accounts.payer.to_account_info(),
108                 system_program: ctx.accounts.system_program.to_account_info(),
109                 token_program: ctx.accounts.token_program.to_account_info(),
110                 rent: ctx.accounts.rent.to_account_info(),
111             },
112             &[&seeds[..]],
113         ),
114         Some(1),
115     )?;
```

Recommendations:

Exvul Web3 Security recommends adding event logging for important information.

Result: Confirmed

Fix Result: Fixed

Added nft emit event:

```
emit!(MintHvtEvent {
    who: ctx.accounts.payer.key(),
    mint: ctx.accounts.hvt_mint.key(),
    cvt_name,
    hvt_name,
    nft_id,
});
```

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.38 Reward calculation accuracy issue

ID:	NVE-038	Location:	staking_lp/src/instructions/stake.rs
Severity:	Informational	Category:	Business Issues
Likelihood:	Low	Impact:	Informational

Description:

The staking_lp module may have rounding and decimal truncation issues due to the use of u128 and division operations, resulting in inaccurate rewards. If the reward value is too small, it may be discarded.

```
21      let current_round = pool.current_round_index;
22      if ctx.accounts.my_stake.is_stake {
23          let my_stake = &mut ctx.accounts.my_stake;
24          assert_eq!(my_stake.round_index, current_round);
25
26          // update user earn
27          my_stake.reward += ((my_stake.stake as u128)
28              * (round.accum_reward - my_stake.accum_reward) as u128
29              / 1e9 as u128) as u64;
30          my_stake.last_update_slot = current_slot;
31          my_stake.accum_reward = round.accum_reward;
32
33          //add stake
34          my_stake.stake += amount;
```

Recommendations:

Exvul Web3 Security recommends confirming the calculation accuracy or enhancing the accuracy of the calculation.

Result: Confirmed

Fix Result: Ignored

3.39 If closed is equal to collection.closed, an error is returned

ID:	NVE-039	Location:	premium_cvt/src/lib.rs
Severity:	Informational	Category:	Business Issues
Likelihood:	Low	Impact:	Informational

Description:

Should test closed to see if it is equal to collection.closed, and return an error if so.

```
pub fn set_closed(ctx: Context<Config>, closed: u32) -> Result<()> {
    let collection = &mut ctx.accounts.collection.load_mut()?;
    require!(
        collection.admin == ctx.accounts.payer.key(),
        errors::ErrorCode::InvalidAdmin
    );
    collection.closed = closed;
    msg!("set_closed {}", closed);
    emit!(ClosedEvent { closed });
    Ok(())
}
```

Recommendations:

Exvul Web3 Security recommends checking if closed is equal to collection.closed and returning an error if they are equal.

Result: Confirmed

Fix Result: Fixed

Code has been deleted.

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

3.40 should have msg when set max supply

ID:	NVE-040	Location:	premium_cvt/src/lib.rs
Severity:	Informational	Category:	Business Issues
Likelihood:	Low	Impact:	Informational

Description:

Currently there is no event logged when setting the max supply in the contract.

```
pub fn set_max_supply(ctx: Context<Config>, new_max_supply: u32) -> Result<()> {
    let collection = &mut ctx.accounts.collection.load_mut()?;
    require!(
        collection.admin == ctx.accounts.payer.key(),
        errors::ErrorCode::InvalidAdmin
    );
    assert!(new_max_supply >= collection.current_supply);
    collection.max_supply = new_max_supply;

    Ok(())
}
```

Recommendations:

Exvul Web3 Security recommends adding event logging.

Result: Confirmed

Fix Result: Fixed

Add event:

msg!("Max supply set to {}", new_max_supply);

Fixed version:

https://github.com/soexdev/soex-protocol

041a25073b115d2922536d428ae5590e9a78ffb9

4. CONCLUSION

In this audit, we thoroughly analyzed **Soex** smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been communicated to the project leader. We therefore consider the audit result to be **PASSED**. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5. APPENDIX

5.1 Basic Coding Assessment

5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: Critical

5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found
- Severity: Critical

5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the contract
- Result: Not found
- Severity: Critical

5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.6 Soft Fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.7 Hard Fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: Critical

5.1.8 Abnormal Memo Assessment

- Description: Assess whether there is abnormal memo vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in contract processing.
- Result: Not found
- Severity: Critical

5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: Critical

5.2 Advanced Code Scrutiny

5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: High

5.2.2 Account Permission Control

- Description: Examine permission control issue in the contract
- Results: Not Found
- Severity: Medium

5.2.3 Malicious Code Behavior

- Description: Examine whether sensitive behavior present in the code
- Results: Not found
- Severity: Medium

5.2.4 Sensitive Information Disclosure

- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: Medium

5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: Low

6. DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without ExVul's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ExVul to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. ExVul's position is that each company and individual are responsible for their own due diligence and continuous security. ExVul's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

7. REFERENCES

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound).

https://cwe.mitre.org/data/ definitions/191.html.

[2] MITRE. CWE- 197: Numeric Truncation Error.

https://cwe.mitre.org/data/definitions/197. html.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption.

https://cwe.mitre.org/data/ definitions/400.html.

[4] MITRE. CWE-440: Expected Behavior Violation.

https://cwe.mitre.org/data/definitions/440. html.

[5] MITRE. CWE-684: Protection Mechanism Failure.

https://cwe.mitre.org/data/definitions/ 693.html.

[6] MITRE. CWE CATEGORY: 7PK - Security Features.

https://cwe.mitre.org/data/definitions/ 254.html.

[7] MITRE. CWE CATEGORY: Behavioral Problems.

https://cwe.mitre.org/data/definitions/438. html.

[8] MITRE. CWE CATEGORY: Numeric Errors.

https://cwe.mitre.org/data/definitions/189.html.

[9] MITRE. CWE CATEGORY: Resource Management Errors.

https://cwe.mitre.org/data/ definitions/399.html.

[10] OWASP. Risk Rating Methodology.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

www.exvul.com

contact@exvul.com

@EXVULSEC

github.com/EXVUL-Sec

ExVul