Chainsulting

DUST Protocol

PROGRAM AUDIT

14.10.2022

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of DUST Protocol. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (09.09.2022)	Layout
0.4 (13.09.2022)	Automated Security Testing Manual Security Testing
0.5 (14.09.2022)	Verify Claims and Test Deployment
0.9 (17.09.2022)	Summary and Recommendation
1.0 (19.09.2022)	Final document
1.1 (29.09.2022)	Final re-check

2. Project Overview

DUST protocol is a decentralized protocol and SPL (Solana Program Library) token created on the Solana blockchain with a starting supply of zero, and a maximum supply of 33,300,000. DUST has an emission schedule with multiple halvenings and mining rewards that are earned via staking NFTs. Countless projects have independently adopted DUST within their own ecosystems making it the most used SPL token on the Solana blockchain.

Website: https://www.dustprotocol.com

Documentation: https://docs.dustprotocol.com

Dust Protocol.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the program functioning in a number of scenarios, or creates a risk that the program may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a program, or provides the opportunity to use a program in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the program in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the program and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and rust program developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the program
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the programs to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your programs.

5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Program Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
./gem_bank/src/instructions/init_bank.rs	044c2f674af15d6b561f496ccf2ff863
./gem_bank/src/instructions/set_vault_lock.rs	2c1d221c8c05fe34b0b96e996755cb00
./gem_bank/src/instructions/add_to_whitelist.rs	919d9a95c08ae50adf826d9353e4c95d
./gem_bank/src/instructions/withdraw_gem.rs	beab1bea9cfd7f62a4431827c8479c2b
./gem_bank/src/instructions/remove_from_whitelist.rs	dea3b762baa1c492d0a8c88dfe936ba7
./gem_bank/src/instructions/mod.rs	d2e98dbba14cd3c1e16541530f8397c8
./gem_bank/src/instructions/update_vault_owner.rs	ae23ba338525dec20dfb8bc28562c53a
./gem_bank/src/instructions/update_bank_manager.rs	aa690d1590e27a279e1cee11a4ed1766
./gem_bank/src/instructions/deposit_gem.rs	ca027e0eb2aeea39f371650dcd345037
./gem_bank/src/instructions/init_vault.rs	83f1aa785d9bd56d480f411cff6cd101
./gem_bank/src/instructions/set_bank_flags.rs	6785d223df5df1950abc9f2631eae885
./gem_bank/src/lib.rs	c7edec3164ceab050f8d48c84e9f17b6
./gem_bank/src/state/bank.rs	aec1a65350dde2cdcf14349fd6f7036d
./gem_bank/src/state/gem_deposit_receipt.rs	f2f33346edef7cfa41cef3d471ab5a49
./gem_bank/src/state/whitelist_proof.rs	47dd5e54ac22cb7f2c4af3e0d81d2340
./gem_bank/src/state/mod.rs	2e7256f665f4b9844cdf604107302bbe
./gem_bank/src/state/vault.rs	bd5474f4986bb5ac1edfdb1bf4c87c02

./gem_farm/src/instructions/fund_reward.rs	739071f76c99a9cee9f0dc7d12d6d647
./gem_farm/src/instructions/init_farm.rs	3cd03c30a7626c1bc04630933e5f6f7b
./gem_farm/src/instructions/stake.rs	c56257a9c9d3dbf94927d8d27f94f6df
./gem_farm/src/instructions/refresh_farmer_signed.rs	108f4b83a5313b0a631ff268b53c71e6
./gem_farm/src/instructions/mod.rs	f0b161f2b2cd0793707eb74dd87d2dd1
./gem_farm/src/instructions/authorize_funder.rs	6fb871994f273e582f75770be156a5af
./gem_farm/src/instructions/unstake.rs	b866eb570a2011a5f9921a2849891369
./gem_farm/src/instructions/treasury_payout.rs	46b59d94090b9e9b7c50539880cd82a6
./gem_farm/src/instructions/deauthorize_funder.rs	08aea6ab67136d36fb7e5a18033f9fcc
./gem_farm/src/instructions/remove_from_bank_whitelist.rs	680baa91d9624c75d83304e83c0bb147
./gem_farm/src/instructions/lock_reward.rs	cdf924d3650344a807da531633788df7
./gem_farm/src/instructions/cancel_reward.rs	f2ec35a5a38e893a6b514f8528471d48
./gem_farm/src/instructions/flash_deposit.rs	843dbdb6f27242443392e0a86847aa8b
./gem_farm/src/instructions/refresh_farmer.rs	a6d5f430cbc97c6f060245fe0749144d
./gem_farm/src/instructions/init_farmer.rs	4a6e40b7e10c553710b4f2d76e126556
./gem_farm/src/instructions/claim.rs	4a52dbf7093b0778c23f4a066ac5716b
./gem_farm/src/instructions/update_farm.rs	762203b315f85657a203d334f624b32a
./gem_farm/src/instructions/add_to_bank_whitelist.rs	349d7194bd7a0c2b57b84cbb0556f3d1
./gem_farm/src/number128.rs	6f997108f5826165c852b0e082d3a59d
./gem_farm/src/lib.rs	f2d7eafad82d399ebc8f990e6579ca26
./gem_farm/src/state/farm.rs	2ba703e2aa4b4bb130d5ffc567f6b68f
./gem_farm/src/state/fixed_rewards.rs	b83d5b605c205cbb07739216814beed5
./gem_farm/src/state/variable_rewards.rs	d467a34d09ab2d79e2753a777c4306f1
./gem_farm/src/state/mod.rs	987672ba9c03e963e89f6da733214057
./gem_farm/src/state/farmer.rs	229800f61361d18ebc4ec776f5eb69dd
./gem_farm/src/state/authorization_proof.rs	0396bee90da92ea5e55a0749c1c592e0

5.2 Used Code from other Frameworks/Dependencies

Dependency / Import Path	Source
anchor-lang	https://docs.rs/anchor-lang/0.18.2/anchor_lang/
anchor-spl	https://docs.rs/anchor-spl/0.18.2/anchor_spl/
bitflags	https://docs.rs/bitflags/1.3.2/bitflags/
bytemuck	https://docs.rs/bytemuck/1.7.2/bytemuck/
static_assertions	https://docs.rs/static_assertions/1.1.0/static_assertions
thiserror	https://docs.rs/thiserror/1.0.30/thiserror/
arrayref	https://docs.rs/arrayref/0.3.6/arrayref/
metaplex-token-metadata	https://docs.rs/metaplex-token-metadata/0.0.1/metaplex_token_metadata/
uint	https://docs.rs/uint/0.9.1/uint/
spl-math	https://docs.rs/spl_math/0.1.0/spl_math/

Dependency / Import Path	Source
quote	https://docs.rs/quote/1.0.8/quote/
syn	https://docs.rs/syn/1.0.57/syn/
proc-macro2	https://docs.rs/proc-macro2/1.0/proc-macro2/

6. Scope of Work

The DUST Protocol developer team provided us with the files that needs to be tested. The scope of the audit is the Gem Bank and Farm program.

The team put forward the following assumptions regarding the security, usage of the program:

- The program is coded according to the newest standards and in a secure way.

Gem Bank

- Anyone with a wallet can create a bank
- Each bank has an effectively unlimited number of vaults, each created by a vault owner
- Each vault can store an effectively unlimited number of NFTs (different mints, yes)
  Dirty little secret - it can store normal tokens too, not just NFTs
- You as the bank manager can decide which NFTs are / aren't allowed in the vaults. That's done through whitelisting
- You as the bank manager can lock / unlock specific vaults, controlling whether vault owners are allowed to withdraw / deposit gems
- You also have the option of freezing ALL vaults at the same time
- You as the bank manager can also configure extra points for rarities for your collection

Gem Farm

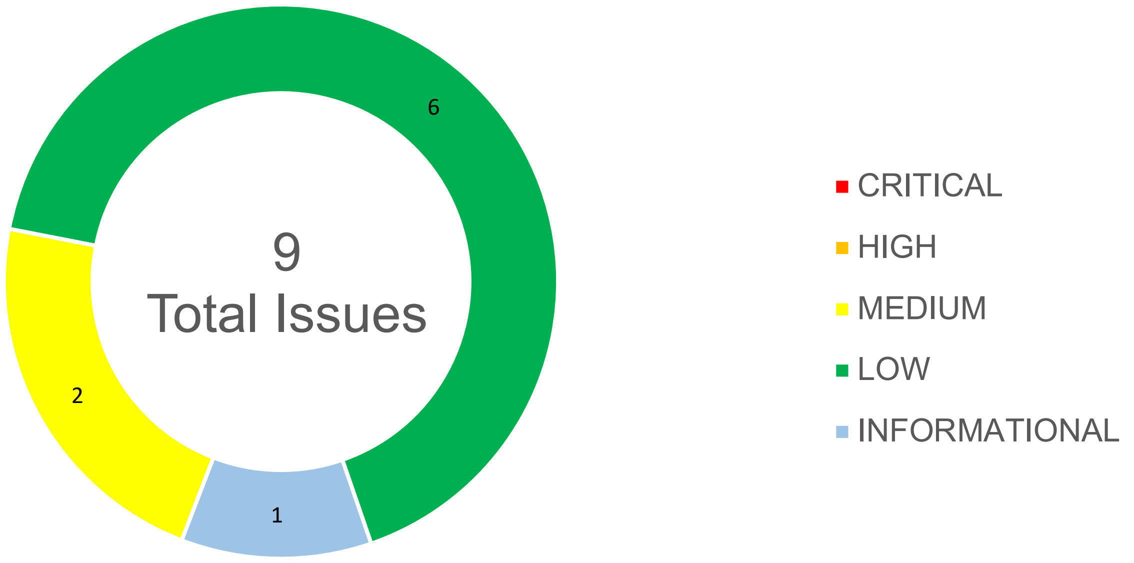
- Anyone with a wallet can start a farm
- Each new farm automatically starts an instance of Gem Bank, used for storing the staked NFTs
- Each farm comes with a bunch of config knobs you can turn, such as minimum staking period, and cooldown period
- When starting a farm you can decide on up to 2 rewards. For each one you choose:
    1. Reward mint
    2. Reward type - fixed or variable

3. As of right now, after the farm has been started, the above can't be changed. This is done for security reasons
    1. Although the reward mint / type can't be changed - the actual schedules / amounts absolutely can!
    2. Worst case you start a new farm with new mints / types

3. After the farm is ready, you as the farm manager authorize one or multiple funders (could be just you), who fund the rewards pots with $$$ and configure the reward schedule
3. Users show up and register as farmers. Once registered they can stake / unstake & claim rewards
    1. Claiming can be done at any point without them having to unstake
3. Any unused rewards can be cancelled
3. There is also an option to lock rewards in place, to effectively "guarantee" them to users
3. Reward distribution depends on both gems staked, and gem rarity

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

6.1 Findings Overview



No	Title	Severity	Status
6.2.1	The Farm Manager Can Lock The Farmer's Stakes	MEDIUM	ACKNOWLEDGED
6.2.2	The Bank Manager Can Lock The Vault's Funds	MEDIUM	ACKNOWLEDGED
6.2.3	The Farm Can End Up Without A Manager	LOW	ACKNOWLEDGED
6.2.4	The authorized_funder_count Attribute Can Get Desynchronized	LOW	ACKNOWLEDGED
6.2.5	The Bank Can End Up Without A Manager	LOW	ACKNOWLEDGED
6.2.6	The Vault Can End Up Without An Owner	LOW	ACKNOWLEDGED
6.2.7	The whitelisted_creators And The whitelisted_mints Attributes Can Get Desynchronized	LOW	ACKNOWLEDGED

6.2.8	The gem_box_count Attribute Can Get Desynchronized	LOW	ACKNOWLEDGED
6.2.9	The Bank Access Is Not Verified Before Creating A Vault	INFORMATIONAL	ACKNOWLEDGED

6.2 Manual Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **0 Critical issues** in the code of the program.

HIGH ISSUES

During the audit, Chainsulting's experts found **0 High issues** in the code of the program.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **2 Medium issues** in the code of the program.

6.2.1 The Farm Manager Can Lock The Farmer's Stakes

Severity: MEDIUM
Status: ACKNOWLEDGED
Code: NA
File(s) affected: update_farm.rs

Attack / Description	The update_farm instruction is used to update the farm's config. The farm manager is able to update min_staking_period_sec, cooldown_period_sec and unstaking_fee_lamp to any value without any restrictions. Therefore, the farm manager can prevent the farmer from unstaking by applying a long min staking period, cooldown duration or a high unstaking fee. This represents a

	significant centralization risk where the farm manager has too much control over the farmer’s gems.
Code	Line 13 – 30 (update_farm.rs)

```rust
pub fn handler(
    ctx: Context<UpdateFarm>,
    config: Option<FarmConfig>,
    manager: Option<Pubkey>,
) -> ProgramResult {
    let farm = &mut ctx.accounts.farm;

    if let Some(config) = config {
        farm.config = config;
    }

    if let Some(manager) = manager {
        farm.farm_manager = manager;
    }

    msg!("updated farm");
    Ok(())
}
```

Result/Recommendation	Consider requiring the farm to be empty of gems before modifying its configuration, or adding restrictions that will prevent the farm manager from inserting big values in FarmConfig attributes.

6.2.2 The Bank Manager Can Lock The Vault’s Funds

Severity: MEDIUM
Status: ACKNOWLEDGED
Code: NA
File(s) affected: set_vault_lock.rs, set_vault_lock.rs

Attack / Description	The set_vault_lock and the set_bank_flags instructions allow the bank manager to lock any vault or freeze them at once, preventing the vault owner from depositing/ withdrawing their gems. This represents a significant centralization risk where the bank manager has too much control over the vaults.

Code	Line 22 – 34 (set_vault_lock.rs)
	`pub fn handler(ctx: Context<SetVaultLock>, vault_locked: bool) -> ProgramResult {`
	`    let bank = &ctx.accounts.bank;`
	`    let vault = &mut ctx.accounts.vault;`
	
	`    if Bank::read_flags(bank.flags)?.contains(BankFlags::FREEZE_VAULTS) {`
	`        return Err(ErrorCode::VaultAccessSuspended.into());`
	`    }`
	
	`    vault.locked = vault_locked;`
	
	`    // msg!("vault {} lock set to {}", vault.key(), vault_locked);`
	`    Ok(())`
	`}`
	
	Line 13 – 21 (set_vault_lock.rs)
	`pub fn handler(ctx: Context<SetBankFlags>, flags: u32) -> ProgramResult {`
	`    let bank = &mut ctx.accounts.bank;`
	
	`    let flags = Bank::read_flags(flags)?;`
	`    bank.reset_flags(flags);`
	
	`    msg!("flags set: {:?}", flags);`
	`    Ok(())`
	`}`

Result/Recommendation	It is recommended to restrict the locking/freezing functionality to only prevent depositing gems to avoid locking the vault owner's gems. According to your documentation this function might be intended: https://docs.gemworks.gg/gem-bank/overview - You as the bank manager can lock / unlock specific vaults, controlling whether vault owners are allowed to withdraw / deposit gems - You also have the option of freezing ALL vaults at the same time

LOW ISSUES

During the audit, Chainsulting's experts found **6 Low issues** in the code of the program.

6.2.3 The Farm Can End Up Without A Manager

Severity: LOW
Status: ACKNOWLEDGED
Code: NA
File(s) affected: update_farm.rs

Attack / Description	When calling the update_farm instruction, there is a possibility that the farm manager sets a wrong address as a new manager, which can result in the farm having no manager.

Code	Line 13 – 30 (update_farm.rs)
	
	`pub fn handler(`
	`    ctx: Context<UpdateFarm>,`
	`    config: Option<FarmConfig>,`

	```
	    manager: Option<Pubkey>,
	) -> ProgramResult {
	    let farm = &mut ctx.accounts.farm;

	    if let Some(config) = config {
	        farm.config = config;
	    }

	    if let Some(manager) = manager {
	        farm.farm_manager = manager;
	    }

	    msg!("updated farm");
	    Ok(())
	}
	```
Result/Recommendation	It is recommended to implement a process where the farm manager sets an address as a farm manager candidate, then this address can only be the farm manager, if calling the program to accept the role.

6.2.4 The authorized_funder_count Attribute Can Get Desynchronized

Severity: LOW
Status: ACKNOWLEDGED
Code: NA
File(s) affected: authorize_funder.rs

Attack / Description	The authorize_funder instruction makes use of init_if_needed in the authorization_proof account. If the account is already initialized, the authorized_funder_count attribute will be incremented without adding a funder.
Code	Line 17 – 27 (authorize_funder.rs)

	```
	// funder
	    pub funder_to_authorize: AccountInfo<'info>,
	    #[account(init_if_needed, seeds = [
	            b"authorization".as_ref(),
	            farm.key().as_ref(),
	            funder_to_authorize.key().as_ref(),
	        ],
	        bump = bump,
	        payer = farm_manager,
	        space = 8 + std::mem::size_of::<AuthorizationProof>())]
	    authorization_proof: Box<Account<'info, AuthorizationProof>>,
	```
	 Line 51 (authorize_funder.rs) <pre>farm.authorized_funder_count.try_add_assign(1)?;</pre>

Result/Recommendation	It is recommended to use init in order to prevent the authorization proofs' reinitialization and attributes desynchronization.

6.2.5 The Bank Can End Up Without A Manager

Severity: LOW
Status: ACKNOWLEDGED
Code: NA
File(s) affected: update_bank_manager.rs

Attack / Description	When calling the update_bank_manager instruction, there is a possibility that the bank manager sets a wrong address as a new_manager, leaving the bank without a manager, which can disable some functionalities.

Code	Line 13 – 20 (update_bank_manager.rs)
	
	`pub fn handler(ctx: Context<UpdateBankManager>, new_manager: Pubkey) -> ProgramResult {`
	`    let bank = &mut ctx.accounts.bank;`
	
	`    bank.bank_manager = new_manager;`
	
	`    msg!("bank manager updated to: {}", new_manager);`
	`    Ok(())`
	`}`

Result/Recommendation	It is recommended to implement a process where the bank manager sets an address as a bank manager candidate, then this address can only be the bank manager if calling the program to accept the role.

6.2.6 The Vault Can End Up Without An Owner

Severity: LOW
Status: ACKNOWLEDGED
Code: NA
File(s) affected: update_vault_owner.rs

Attack / Description	When calling the update_vault_owner instruction, there is a possibility that the vault owner sets a wrong address as a new_owner, which can result in the vault having no owner, and the funds getting locked.

Code	Line 18 – 32 (update_vault_owner.rs)
	
	`pub fn handler(ctx: Context<UpdateVaultOwner>, new_owner: Pubkey) -> ProgramResult {`
	`    let bank = &ctx.accounts.bank;`
	`    let vault = &mut ctx.accounts.vault;`

	```
	    if Bank::read_flags(bank.flags)?.contains(BankFlags::FREEZE_VAULTS) {
	        return Err(ErrorCode::VaultAccessSuspended.into());
	    }

	    // todo is it wise that we're letting them set the owner w/o checking signature?
	    //  what if they accidentally set the wrong one? The vault will be frozen forever.
	    vault.owner = new_owner;

	    msg!("owner updated to: {}", new_owner);
	    Ok(())
	}
	```
Result/Recommendation	It is recommended to implement a process where the vault owner sets an address as a vault owner candidate, then this address can only be the vault owner if calling the program to accept the role.

6.2.7 The whitelisted_creators And The whitelisted_mints Attributes Can Get Desynchronized

Severity: LOW
Status: ACKNOWLEDGED
Code: NA
File(s) affected: add_to_whitelist.rs

Attack / Description	The add_to_whitelist instruction makes use of init_if_needed in the whitelist_proof account. If the account is already initialized, the whitelisted_creators or the whitelisted_mints attribute will be incremented without adding a new whitelist proof, since the proof already exists in the past account.

Code	Line 18 – 27 (update_vault_owner.rs)
	```
	  #[account(init_if_needed,
	      seeds = [
	```

	```
	            b"whitelist".as_ref(),
	            bank.key().as_ref(),
	            address_to_whitelist.key().as_ref(),
	        ],
	        bump = bump,
	        payer = payer,
	        space = 8 + std::mem::size_of::<WhitelistProof>())]
	    pub whitelist_proof: Box<Account<'info, WhitelistProof>>,
	```
	 Line 55 – 60 (update_vault_owner.rs)
	```
	   if whitelist_type.contains(WhitelistType::CREATOR) {
	        bank.whitelisted_creators.try_add_assign(1)?;
	    }
	    if whitelist_type.contains(WhitelistType::MINT) {
	        bank.whitelisted_mints.try_add_assign(1)?;
	    }
	```

Result/Recommendation	It is recommended to use init in order to prevent the whitelist proofs' reinitialization and attributes desynchronization.

6.2.8 The gem_box_count Attribute Can Get Desynchronized

Severity: LOW
Status: ACKNOWLEDGED
Code: NA
File(s) affected: deposit_gem.rs

Attack / Description	The deposit_gem instruction makes use of init_if_needed in the gem_box account. If the account is already initialized, the gem_box_count attribute will be incremented without adding a new gem box.

Code	Line 31 – 40 (deposit_gem.rs)
	
	`#[account(init_if_needed, seeds = [`
	`            b"gem_box".as_ref(),`
	`            vault.key().as_ref(),`
	`            gem_mint.key().as_ref(),`
	`        ],`
	`        bump = bump_gem_box,`
	`        token::mint = gem_mint,`
	`        token::authority = authority,`
	`        payer = owner)]`
	`    pub gem_box: Box<Account<'info, TokenAccount>>,`
	
	Line 216 (deposit_gem.rs)
	
	`    vault.gem_box_count.try_add_assign(1)?;`

Result/Recommendation	It is recommended to use init in order to prevent the gem boxes' reinitialization and attributes desynchronization.

INFORMATIONAL ISSUES

During the audit, Chainsulting's experts found **1 Informational issue** in the code of the program.

6.2.9 The Bank Access Is Not Verified Before Creating A Vault

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: NA
File(s) affected: init_vault.rs

Attack / Description	The bank manager can lock/unlock specific vaults, controlling whether the vault owners are allowed to withdraw/deposit gems. Has also has the option of freezing all vaults at the same time using the bank flags. In the init_vault instruction, the bank flags are not verified before creating a new vault.
Code	Line 33 - 57 (init_vault.rs)

```rust
pub fn handler(ctx: Context<InitVault>, owner: Pubkey, name: String) -> ProgramResult {
    // record total number of vaults in bank's state
    let bank = &mut ctx.accounts.bank;
    let vault = &mut ctx.accounts.vault;

    bank.vault_count.try_add_assign(1)?;

    // derive the authority responsible for all token transfers within the new vault
    let vault_address = vault.key();
    let authority_seed = &[vault_address.as_ref()];
    let (authority, bump) = Pubkey::find_program_address(authority_seed, ctx.program_id);

    // record vault's state
    vault.bank = bank.key();
    vault.owner = owner;
    vault.creator = ctx.accounts.creator.key();
    vault.authority = authority;
    vault.authority_seed = vault_address;
    vault.authority_bump_seed = [bump];
    vault.locked = false;
    (&mut vault.name[..]).write_all(name.as_bytes())?;

    msg!("new vault founded by {}", &ctx.accounts.creator.key());
    Ok(())
}
```

Result/Recommendation	Consider verifying if initiating a new vault in a frozen bank is allowed by the business logic. According to your documentation this function might be intended: https://docs.gemworks.gg/gem-bank/overview - You as the bank manager can lock / unlock specific vaults, controlling whether vault owners are allowed to withdraw / deposit gems - You also have the option of freezing ALL vaults at the same time

6.3 Automated Vulnerability Test

6.3.1 **Cargo**

The cargo test command passes successfully, these unit tests assure that the functions are correct and return   expected results.

6.3.2 **Anchor**

The Anchor test command passes successfully

6.3.3 **Coverage (Cargo Tarpaulin)**

The test coverage is 41,34%, it is recommended to improve the test coverage in order to assure the program's functionality.

6.3.4 **Soteria**

Soteria has detected an issue concerning the init_if_needed keyword, which contains a security bug that makes the instruction susceptible to an account type cosplay attack. How- ever, this bug was already fixed in the audited program by checking the first 8 bytes of the instruction data to be either zero or the expected discriminator.

6.3.5 **Cargo Audit**

No major results were found using cargo audit, it is recommended to upgrade the regex version to one greater than 1.5.5.

6.3.6 **Cargo Geiger**

No major issues related to the usage of unsafe Rust code were found using cargo geiger.

6.4 Verify Claims

- The program is coded according to the newest standards and in a secure way.

Gem Bank

- Anyone with a wallet can create a bank
- Each bank has an effectively unlimited number of vaults, each created by a vault owner
- Each vault can store an effectively unlimited number of NFTs (different mints, yes)
  Dirty little secret - it can store normal tokens too, not just NFTs
- You as the bank manager can decide which NFTs are / aren't allowed in the vaults. That's done through whitelisting
- You as the bank manager can lock / unlock specific vaults, controlling whether vault owners are allowed to withdraw / deposit gems
- You also have the option of freezing ALL vaults at the same time
- You as the bank manager can also configure extra points for rarities for your collection

Gem Farm

- Anyone with a wallet can start a farm
- Each new farm automatically starts an instance of Gem Bank, used for storing the staked NFTs
- Each farm comes with a bunch of config knobs you can turn, such as minimum staking period, and cooldown period
- When starting a farm you can decide on up to 2 rewards. For each one you choose:
    1. Reward mint
    2. Reward type - fixed or variable

3. As of right now, after the farm has been started, the above can't be changed. This is done for security reasons
    1. Although the reward mint / type can't be changed - the actual schedules / amounts absolutely can!
    2. Worst case you start a new farm with new mints / types

3. After the farm is ready, you as the farm manager authorize one or multiple funders (could be just you), who fund the rewards pots with $$$ and configure the reward schedule

3. Users show up and register as farmers. Once registered they can stake / unstake & claim rewards
    1. Claiming can be done at any point without them having to unstake

3. Any unused rewards can be cancelled

3. There is also an option to lock rewards in place, to effectively "guarantee" them to users
3. Reward distribution depends on both gems staked, and gem rarity

Status: tested and verified ✅

7. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the solana program.

The main goal of the audit was to verify the claims regarding the security and functions of the program. During the audit, no critical, no high, 2 medium, 6 low and 1 informational issue have been found, after the manual and automated security testing. We advise the

DUST Protocol team to implement the recommendations contained in all our findings, to further enhance the code's security and readability. It’s as well recommended to improve the test coverage (41,34%) in order to assure the program’s functionality

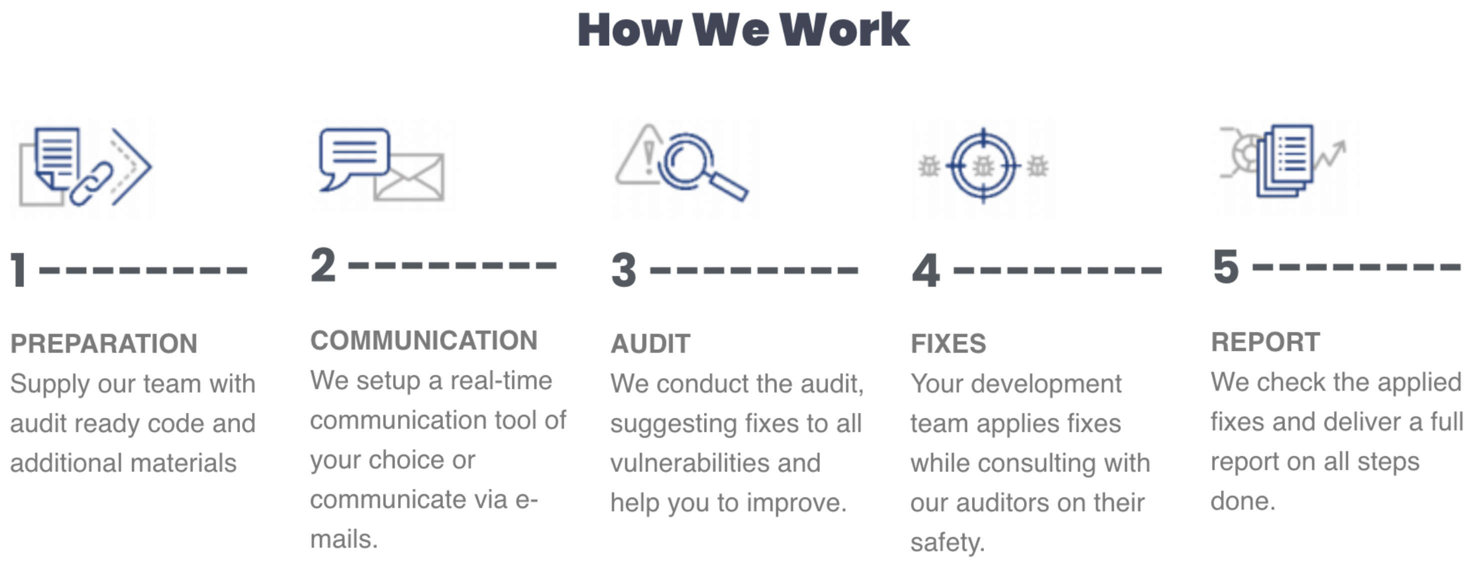
8. About the Auditor

Chainsulting is a professional software development firm, founded in 2017 and based in Germany. They show ways, opportunities, risks and offer comprehensive web3 solutions. Their services include web3 development, security and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Solana, Tezos, Ethereum, Binance Smart Chain, and Polygon to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secure the smart contracts of 1Inch, POA Network, Unicrypt, LUKSO among numerous other top DeFi projects.

Chainsulting currently secures $100 billion in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the web3 sector to deliver top-notch smart contract audit solutions, tailored to the clients' evolving business needs.

Check our website for further information: https://chainsulting.de



How We Work

1 --------

PREPARATION
Supply our team with audit ready code and additional materials

2 --------

COMMUNICATION
We setup a real-time communication tool of your choice or communicate via e-mails.

3 --------

AUDIT
We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.

4 --------

FIXES
Your development team applies fixes while consulting with our auditors on their safety.

5 --------

REPORT
We check the applied fixes and deliver a full report on all steps done.