Cyberscope

Audit Report

BrushO Network

January 2025

Repository	https://github.com/aitoothbrush/brusho-program-library/tree/main/programs/brusho-nft-manager
Commit	b4b2725ba92dab1b15ff0e3db7441a7d76f5c3fe
Audited by	© cyberscope

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Repository	https://github.com/aitoothbrush/brusho-program-library/tree/main/programs/brusho-nft-manager
Commit	b4b2725ba92dab1b15ff0e3db7441a7d76f5c3fe

Audit Updates

Initial Audit	20 Jan 2025

Source Files

Filename	SHA256
error.rs	4eabf5b1bad9e1ac7a507b800e72cbcda2a124f9df9e3984683fd721d692a861
instructions/update_issuing_authority.rs	0b0eb230f6bad01dda7e00facad135c0f712714cc67fc032358a44ea8f4bc4cd
instructions/set_maker_tree.rs	bd6f95e1f35741851f58a06bb0f812c1d93a770deb64c9f9dc44ea2bbc85383e
instructions/mod.rs	7252d97f881327cdc0647ef6f980bce3a0533d051fffdb842b354fe2418e53d0
instructions/initialize_maker.rs	ebf291b1b43edd56434b96ce72518e1d18a9934ef754c689cf9ee69365449cf3
instructions/issue_brush_nft.rs	1ed67a7944aa328cb7995fb2150b3e44c4bdf329b7c6c74830d0b0c0a1a13439
instructions/update_maker.rs	0e18d8f531caa7673e72601024abbe24fcd0fee39b9946b51feb66a1555619e4

lib.rs	75ee8dda600c3f46df70bdf2feb375c78bd9059b4fb237262d78fb58f1275444
token_metadata.rs	a9ef9e28f21cc97ae454c426d9b14d5dc67eada17863aef2900acc5c3e282c41
state.rs	7d6978fd953bb7a88f12b3aa7b0e95b9a32c2bc5e32eb95f3339408064b0262b

Overview

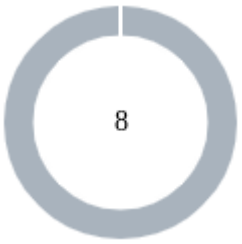
The contract is designed to facilitate the creation, management, and issuance of NFTs within a secure and scalable framework on the Solana blockchain. It provides functionality to initialize NFT creators, referred to as "Makers," who are responsible for managing their unique NFT collections. Each Maker is associated with a governance context and can create and link collections to metadata accounts that define the properties of the NFTs, such as their name, symbol, and metadata URI. These collections can also include master editions to define rarity or scarcity within the collection.

In addition to managing collections, the contract enables the issuance of individual NFTs tied to specific metadata. This includes the ability to generate NFTs with unique identifiers (e.g., `brush_no`) and assign them detailed metadata that links them to the Maker's collection. To support scalability, the contract integrates Merkle tree-based compression, allowing Makers to issue large-scale collections of NFTs efficiently. This compression is achieved through integration with the Bubblegum program, which reduces on-chain storage costs and computational overhead, making the system suitable for applications requiring high-volume NFT issuance.

The contract also provides mechanisms for updating and managing various parameters associated with Makers and collections. Makers can update their issuing authority, manage the Merkle tree configurations linked to their collections, and toggle their active status. By utilizing cross-program invocations (CPIs) to trusted Solana programs like `mpl_token_metadata` and `Bubblegum`, the contract ensures compatibility with the Solana ecosystem while offloading certain responsibilities, such as metadata management and Merkle tree operations, to well-audited external programs.

This implementation combines secure management of NFT collections with the flexibility to issue traditional and compressed NFTs. Its structured functionality allows projects to scale NFT operations seamlessly, whether for art, collectibles, or high-frequency applications like gaming or tokenized assets, while adhering to Solana standards and maintaining metadata integrity.

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	8

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	0	8	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	IEH	Incomplete Error Handling	Acknowledged
	ISU	Inefficient State Updates	Acknowledged
	MDEM	Missing Descriptive Error Messages	Acknowledged
	MEE	Missing Event Emission	Acknowledged
	MCC	Multiple CPI Calls	Acknowledged
	SI	Spelling Inconsistency	Acknowledged
	UBNL	Unchecked Brush Number Length	Acknowledged
	UMUL	Unchecked Metadata URL Length	Acknowledged

IEH - Incomplete Error Handling

Criticality	Minor / Informative
Location	initialize_maker.rs#L135 token_metadata.rs#L63
Status	Acknowledged

Description

The contract invokes external CPI calls such as `create_metadata_accounts_v3` and `create_master_edition_v3` for managing metadata and editions for NFTs. However, these calls lack detailed error handling or logging, making it difficult to debug issues if they fail. Without specific error messages, identifying the root cause of failures becomes challenging, potentially increasing downtime and reducing the maintainability of the contract.

The current implementation does not provide error-specific messages or context, leaving the debugging process reliant on guesswork. This lack of insight into errors can hinder efficient troubleshooting and affect the overall robustness of the contract.

```
create_metadata_accounts_v3(cpi_context, data, true, details)?;
create_master_edition_v3(cpi_context, Some(0))?;
```

```
cpi.invoke_signed(ctx.signer_seeds).map_err(Into::into)
```

Recommendation

It is recommended to implement detailed error handling for all CPI calls by wrapping them in logic that provides specific and descriptive error messages. For example, including contextual messages such as "Error during metadata account creation" or "Error during master edition creation" helps identify the operation that failed. This would facilitate efficient debugging, improve traceability, and enhance the maintainability of the contract.

ISU - Inefficient State Updates

Criticality	Minor / Informative
Location	state.rs#L5
Status	Acknowledged

Description

The `Maker` account contains multiple fields, including critical identifiers such as `merkle_tree`, `realm`, `name`, and `reserved` arrays. While the structure is comprehensive, updating specific fields such as `merkle_tree` requires serializing and deserializing the entire `Maker` account. This process includes handling relatively static or large fields like `name` (a dynamic `String`) and `reserved` arrays, even if they are not being modified. This inefficiency increases computational costs and leads to suboptimal transaction performance, especially in scenarios where frequent updates are required.

```
#[account]
#[derive(Default)]
pub struct Maker {
    pub realm: Pubkey,
    pub realm_authority: Pubkey,
    pub collection: Pubkey,
    pub merkle_tree: Pubkey,
    pub update_authority: Pubkey,
    pub issuing_authority: Pubkey,
    pub name: String,
    pub is_active: bool,
    pub bump: u8,
    pub collection_bump: u8,
    pub reserved1: [u8; 5],
    pub reserved2: [u64; 8],
}
```

Recommendation

It is recommended to optimise the structure of the `Maker` account by:

1. Splitting frequently updated fields (e.g., `merkle_tree`) into a separate account, such as `MakerMetadata`, to reduce the serialization footprint for updates.

2. Replacing dynamic fields like `name` (`String`) with fixed-size arrays (`[u8; N]`) to minimise serialization variability and ensure efficient data handling.

This restructuring will significantly improve transaction efficiency, particularly in high-frequency update scenarios.

Team Update

The team has acknowledged that this is not a security issue and states:

Update operations on 'Maker' are rare.

MDEM - Missing Descriptive Error Messages

Criticality	Minor / Informative
Location	error.rs#L3
Status	Acknowledged

Description

The error codes defined in the `BnmError` enum lack descriptive messages in their `#[msg]` attributes. Error messages are crucial for providing clear and actionable information about why a specific error occurred. Without meaningful messages, it becomes difficult for developers or users interacting with the contract to understand and debug the issues, leading to a poor experience and increased debugging overhead.

```
#[error_code]
pub enum BnmError {
    #[msg("")]
    InvalidMakerNameLength,
    #[msg("")]
    InvaliMetadataUrlLength,
    #[msg("")]
    InvalidRealmAuthority,
    #[msg("")]
    InactiveMaker,
}
```

Recommendation

It is recommended to replace the empty `#[msg]` attributes with clear, descriptive error messages.This improves the usability and maintainability of the contract by making errors more informative and easier to interpret.

MEE - Missing Event Emission

Criticality	Minor / Informative
Location	initialize_maker.rs update_issuing_authority.rs update_maker.rs
Status	Acknowledged

Description

The contract lacks event emissions for various state changes across multiple functions. Significant operations, such as account initialization, updates to key fields like `issuing_authority`, `update_authority`, and `is_active`, as well as other state modifications, are performed without emitting events. Event emissions are essential for maintaining transparency, enabling real-time monitoring, and ensuring that external systems or users can track and respond to these updates programmatically. The absence of such events reduces observability, making it challenging to audit and trace important state changes on-chain.

```
pub fn (ctx: Context<InitializeMaker>, args: InitializeMakerArgs) ->
Result<()> {
    let maker = Maker { ... };
    ctx.accounts.maker.set_inner(maker);
    Ok(())
}
```

```
pub fn update_issuing_authority(ctx: Context, args:
UpdateIssuingAuthorityArgs) -> Result<()> {
    let maker = &mut ctx.accounts.maker;
    maker.issuing_authority = args.issuing_authority;
    Ok(())
}
```

```
pub fn update_maker(ctx: Context, args: UpdateMakerArgs) -> Result<()> {
    if let Some(update_authority) = args.update_authority {
        maker.update_authority = update_authority;
    }
    maker.is_active = args.is_active;
    Ok(())
}
```

Recommendation

It is recommended to incorporate event emissions for all significant state changes in the contract. Each event should capture relevant details, such as the account address, old and new values, and other metadata related to the update. Adding events will enhance transparency, enable real-time tracking, and improve the audibility of on-chain actions, ensuring that critical operations are observable and traceable for external systems and users. This recommendation applies broadly to all areas of the code where significant state changes occur.

MCC - Multiple CPI Calls

Criticality	Minor / Informative
Location	issue_brush_nft.rs#L182
Status	Acknowledged

Description

The contract executes multiple CPI (Cross-Program Invocation) calls to external programs such as Bubblegum and Token Metadata within a single function. These calls include operations like `mint_to_collection_v1`, which require interactions with multiple accounts and signers. While these CPI calls may be necessary for the intended functionality, their repeated execution in a single function increases computational overhead and can result in higher transaction fees. Additionally, these calls may risk exceeding compute limits in high-load scenarios, potentially leading to transaction failures. The computational cost and the potential impact on users' experience make this an area of concern for optimisation.

```
mint_to_collection_v1(
    ctx.accounts
        .mint_to_collection_ctx()
        .with_remaining_accounts(vec![creator,
brush_no_to_asset_creator])
        .with_signer(&[
            maker_signer_seeds,
            top_creator_signer_seeds,
            brush_no_to_asset_signer_seeds,
        ]),
    metadata,
)?;
```

Recommendation

It is recommended to review and optimise the CPI logic to reduce the number of external program interactions within a single function. Where possible, consolidate related CPI calls to minimise redundancy and computational overhead. This approach will help reduce transaction fees, lower the risk of exceeding compute limits, and improve the overall efficiency of the contract. Additionally, consider structuring operations to batch processes

or streamline account interactions to achieve the same functionality with fewer external calls.

SI - Spelling Inconsistency

Criticality	Minor / Informative
Location	error.rs#L8
Status	Acknowledged

Description

The code contains is a typo in the error code name `InvaliMetadataUrlLength`, which should be spelled as `InvalidMetadataUrlLength`. Spelling inconsistencies in error codes can create confusion and reduce the professionalism of the codebase. It can also lead to issues when referencing these error codes in other parts of the contract.

```
#[error_code]
pub enum BnmError {
    #[msg("")]
    InvaliMetadataUrlLength,
}
```

Recommendation

It is recommended to correct the typo by renaming `InvaliMetadataUrlLength` to `InvalidMetadataUrlLength`. Ensure that all references to this error code in the codebase are also updated to maintain consistency and clarity.

UBNL - Unchecked Brush Number Length

Criticality	Minor / Informative
Location	issue_brush_nft.rs#L13
Status	Acknowledged

Description

The contract does not validate the length of the `brush_no` input, potentially leading to storage inefficiencies. Since `brush_no` is stored in accounts as a `String`, excessively long inputs can increase storage costs and lead to potential performance issues. Unrestricted lengths could also introduce vulnerabilities, such as denial-of-service risks due to excessive resource consumption, especially in environments with limited storage capabilities.

```
pub struct IssueBrushNftArgs {
    pub brush_no: String,
    pub metadata_url: String,
}
```

Recommendation

It is recommended to validate the length of `brush_no` during input processing to ensure it remains within a reasonable and practical limit, such as 50 characters. This validation will help optimise storage usage, prevent excessive resource consumption, and maintain the contract's overall efficiency and reliability.

UMUL - Unchecked Metadata URL Length

Criticality	Minor / Informative
Location	issue_brush_nft.rs#L13
Status	Acknowledged

Description

The contract does not validate the length of the `metadata_url` input, which can lead to inefficiencies in storage and increased transaction costs. Since `metadata_url` is a critical field for NFT metadata, excessively long URLs can unnecessarily inflate storage requirements, degrade performance, and raise operational costs. Without proper length restrictions, the contract risks inefficient resource usage and potential vulnerabilities in systems with constrained storage or computational limits.

```
pub struct IssueBrushNftArgs {
    pub brush_no: String,
    pub metadata_url: String,
}
```

Recommendation

It is recommended to validate the length of the `metadata_url` input during processing to ensure it remains within a practical limit, such as 100 bytes. This measure will optimise storage efficiency, reduce transaction costs, and maintain the contract's overall performance and reliability.

Summary

The NFT Manager contract facilitates the creation, management, and issuance of traditional and compressed NFTs on the Solana blockchain. It leverages governance controls, Merkle tree-based compression, and integrations with trusted programs like `mpl_token_metadata` and Bubblegum to efficiently manage metadata and scale NFT collections. This audit evaluates potential security vulnerabilities, business logic issues, and opportunities to enhance scalability, fraud prevention, and operational efficiency. The team has acknowledged the findings.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope'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 regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document 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 represents an extensive assessment 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 Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope'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. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

cyberscope.io