Accretion

SECURITY ASSESSMENT

Provided by Accretion Labs Pte Ltd. for realms
September 10, 2025
A25REA2

realms

AUDITORS

Role	Name
Lead Auditor	Robert Reith (robert@accretion.xyz)
Auditor	0xbountyhunt3r (contact@accretion.xyz)

CLIENT

realms (https://realms.today) is the primary DAO governance platorm in the Solana ecosystem. It is used by most Solana DAOs to manage their decision process and provides a complete solution for Solana DAOs including DAO token management, council tokens, different voting configurations, treasury management, and more. Realms engaged Accretion to conduct a security assessment of a PR which implements support for Metaplex Core NFTs as voter authentication.

ENGAGEMENT TIMELINE

16 Jul — Project Kickoff
Initial planning and scope definition

16 Jul — Assessment Begins
Security review and testing phase

30 Jul — Review Fixes
Security recommendations are given and implemented

10 Sep — Project Completion
Report delivery and on-chain confirmation

AUDITED CODE

Program 1

ProgramID: n/a

Repository:

https://github.com/Mythic-Project/governance-program-library/pull/2

ASSESSMENT

The security assessment was conducted on an update of Realms's governance program, implementing a new feature allowing NFT holders to participate in governance, treating NFT assets similarly to tokens. Users can vote on proposals by proving ownership of NFTs within an approved collection. This feature integrates seamlessly into the existing governance program as a separate plugin.

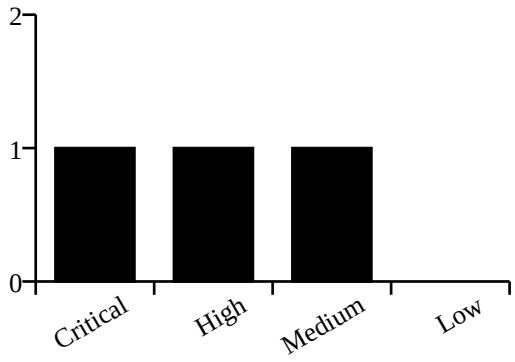
CODE ASSESSMENT

The code was added to the existing program using the native solana SDK. They followed a consistent coding style and the code is well structured, using exising governance utility methods for authentication and validation. In addition to that, basic tests were implemented.

KEY FINDINGS

Our main findings included one critical issue from a missing program ownership check allowing users to supply fake NFTs and vote using them. Another issue was found that allowed users to pre-vote before creating a proposal, which potentially could have enabled sophisticated governance attacks. All issues were fixed by the Realms team.

SEVERITY DISTRIBUTION



ENGAGEMENT SCOPE

The scope of this security assessment was a full review of the following items:

Item 1: Metaplex NFT Core Plugin for Realms

Link: https://github.com/Mythic-Project/governance-program-library/pull/2
Commit: 1a81ca8f80bd9a41c4467de573ed801d54eb5da1
Program ID: n/a
Audit Result:

- **Audited Commit:** 1a81ca8f80bd9a41c4467de573ed801d54eb5da1
- **Status:** unverified
- **Comment:** Not verified

ISSUES SUMMARY

ID	TITLE	SEVERITY	STATUS
ACC-C1	Missing Program Ownership Check on Assets	critical	fixed
ACC-H1	Can Vote on Proposals That Don't Exist Yet	high	fixed
ACC-M1	Voter Weight Record Weight Can Be Set to 0 By Third Parties	medium	fixed

DETAILED ISSUES

ID	ACC-C1
Title	Missing Program Ownership Check on Assets
Severity	critical
Status	fixed

Description

We found that the instruction `update_voter_weight_record` iterates over the remaining accounts, deserializing each account and passing it to `resolve_nft_vote_weight_and_mint` without checking the program ownership of the asset account. This means that an attacker can provide a fake account with data that indicates that this account is an NFT belonging to the correct NFT collection, even when this is not the case. This results in potentially being able to steal votes without owning an NFT.

Location

https://github.com/Mythic-Project/governance-program-library/blob/848ce7e5e9de239ff2fce8626737b02e8477cec9/programs/core-voter/src/instructions/update_voter_weight_record.rs#L52-L62

https://github.com/Mythic-Project/governance-program-library/blob/848ce7e5e9de239ff2fce8626737b02e8477cec9/programs/core-voter/src/instructions/cast_nft_vote.rs#L75-L84

Relevant Code

```
/// update_voter_weight_record.rs L52-L62
    for asset in ctx.remaining_accounts.iter() {
        let (nft_vote_weight, _) = resolve_nft_vote_weight_and_mint(
            registrar,
            governing_token_owner,
            asset.key.clone(),
            &BaseAssetV1::from_bytes(&asset.data.borrow()).unwrap(),
            &mut unique_nft_mints,
        )?;

        voter_weight = voter_weight.checked_add(nft_vote_weight as u64).unwrap();
    }
```

```
/// cast_nft_vote.rs L75-L84
    for (asset, asset_vote_record_info) in
        ctx.remaining_accounts.iter().tuples()
    {
        let (asset_vote_weight, asset_mint) = resolve_nft_vote_weight_and_mint(
            registrar,
            &governing_token_owner,
            asset.key.clone(),
            &BaseAssetV1::from_bytes(&asset.data.borrow()).unwrap(),
            &mut unique_asset_mints,
        )?;
```

Mitigation Suggestion

Add a program ownership check when iterating over the remaining accounts. Each asset should belong to the MPL program.

Remediation

Fixed in commit `35d06d8da110104cf10a995cc95de619402119b5`.

ID	ACC-H1
Title	Can Vote on Proposals That Don't Exist Yet
Severity	high
Status	fixed

Description

We found that the program allows to cast and NFT vote for a Proposal that does not exist yet, as long as we have the Proposal Pubkey. Furthermore votes don't expire, and can only be relinquished when the proposal exists. This opens the door to an interesting governance attack where over a long timeframe an attacker buys NFTs, casts a vote on a non-existant proposal, and then sells the NFT again to gain funding to buy the next NFT and repeat the process. This way, an attacker could accumulate votes of a large share of NFTs in circulation with a fraction of the cost. At some point, the attacker then creates a malicious proposal at the proposal address where all the votes have been cast for, gaining a significant advantage in the vote.

Location

https://github.com/Mythic-Project/governance-program-library/blob/848ce7e5e9de239ff2fce8626737b02e8477cec9/programs/core-voter/src/instructions/cast_nft_vote.rs#L54-L57

Relevant Code

```
/// cast_nft_vote.rs L54-L57
pub fn cast_nft_vote<'a, 'b, 'c, 'info>(
    ctx: Context<'a, 'b, 'c, 'info, CastNftVote<'info>>,
    proposal: Pubkey,
) -> Result<()> {
```

Mitigation Suggestion

When casting an NFT vote, ensure that the Proposal exists and is in Voting state.

Remediation

Fixed in commit `97997faea513ce2e1c5c52709dedd49964f5fc70`.

ID	ACC-M1
Title	Voter Weight Record Weight Can Be Set to 0 By Third Parties
Severity	medium
Status	fixed

Description

We found that as the update voter weight record instruction is unauthenticated, an attacker could always call this instruction with no remaining accounts, which will lead to this account having a voter_weight of 0. This should work as long as the caller uses a VoterWeightAction that is not `CastVote`. In practice, one could disrupt a user who is in the process of accumulating their NFT votes to cast an NFT vote across multiple transactions:

1. User has 20 NFTs in a collection that can vote 2. User calls cast_nft_vote with 5 NFTs (1-5) 3. User calls cast_nft_vote with 5 NFTs (6-10) 4. User calls cast_nft_vote with 5 NFTs (11-15) 4a. <----- ATTACKER calls update_voter_weight_record, resetting weight to 0 5. User calls cast_nft_vote with 5 NFTs (16-20) and uses vote weight record

Now the user will only have the votes of 5 NFTs, while all the asset votes will have been created for all 20 NFTs. They will have to reset their vote to relinquish their nft votes and try again.

Location

https://github.com/Mythic-Project/governance-program-library/blob/848ce7e5e9de239ff2fce8626737b02e8477cec9/programs/core-voter/src/instructions/update_voter_weight_record.rs#L52-L62

Relevant Code

```
/// update_voter_weight_record.rs L52-L62
    for asset in ctx.remaining_accounts.iter() {
        let (nft_vote_weight, _) = resolve_nft_vote_weight_and_mint(
            registrar,
            governing_token_owner,
            asset.key.clone(),
            &BaseAssetV1::from_bytes(&asset.data.borrow()).unwrap(),
            &mut unique_nft_mints,
        )?;

        voter_weight = voter_weight.checked_add(nft_vote_weight as u64).unwrap();
    }
```

Mitigation Suggestion

Add an authentication mechanism to the update functions.

Remediation

Fixed in commit `1a81ca8f80bd9a41c4467de573ed801d54eb5da1`.

APPENDIX

Vulnerability Classification

We rate our issues according to the following scale. Informational issues are reported informally to the developers and are not included in this report.

Severity	Description
Critical	Vulnerabilities that can be easily exploited and result in loss of user funds, or directly violate the protocol's integrity. Immediate action is required.
High	Vulnerabilities that can lead to loss of user funds under non-trivial preconditions, loss of fees, or permanent denial of service that requires a program upgrade. These issues require attention and should be resolved in the short term.
Medium	Vulnerabilities that may be more difficult to exploit but could still lead to some compromise of the system's functionality. For example, partial denial of service attacks, or such attacks that do not require a program upgrade to resolve, but may require manual intervention. These issues should be addressed as part of the normal development cycle.
Low	Vulnerabilities that have a minimal impact on the system's operations and can be fixed over time. These issues may include inconsistencies in state, or require such high capital investments that they are not exploitable profitably.
Informational	Findings that do not pose an immediate risk but could affect the system's efficiency, maintainability, or best practices.

Audit Methodology

Accretion is a boutique security auditor specializing in Solana's ecosystem. We employ a customized approach for each client, strategically allocating our resources to maximize code review effectiveness. Our auditors dedicate substantial time to developing a comprehensive understanding of each program under review, examining design decisions, expected and edge-case behaviors, invariants, optimizations, and data structures, while meticulously verifying mathematical correctness—all within the context of the developers' intentions.

Our audit scope extends beyond on-chain components to include associated infrastructure, such as user interfaces and supporting systems. Every audit encompasses both a holistic protocol design review and detailed line-by-line code analysis.

During our assessment, we focus on identifying:
• Solana-specific vulnerabilities
• Access control issues
• Arithmetic errors and precision loss
• Race conditions and MEV opportunities
• Logic errors and edge cases
• Performance optimization opportunities
• Invariant violations
• Account confusion vulnerabilities
• Authority check omissions
• Token22 implementation risks and SPL-related pitfalls
• Deviations from best practices

Our approach transcends conventional vulnerability classifications. We continuously conduct ecosystem-wide security research to identify and mitigate emerging threat vectors, ensuring our audits remain at the forefront of Solana security practices.