Matrix Security Audit

Report Version 1.1

December 10, 2024

Conducted by **Hunter Security**

Table of Contents

1 About Hunter Security

Hunter Security is an industry-leading smart contract security auditing firm. Having conducted over 100 security audits protecting over $1B of TVL, our team always strives to deliver top-notch security services to the best DeFi protocols. For security audit inquiries, you can reach out on Telegram or Twitter at *@georgehntr*.

2 Disclaimer

Audits are a time-, resource-, and expertise-bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can reveal the presence of vulnerabilities, but cannot guarantee their absence.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - involves a small loss of funds or affects a core functionality of the protocol.
- **Low** - encompasses any unexpected behavior that is non-critical.

3.2 Likelihood

- **High** - a direct attack vector; the cost is relatively low compared to the potential loss of funds.
- **Medium** - only a conditionally incentivized attack vector, with a moderate likelihood.
- **Low** - involves too many or unlikely assumptions; offers little to no incentive.

3.3 Actions required by severity level

- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	Matrix
Commit hash	ade88834fbb98d75634ded2bdb3ec77617a31bee
Resolution	39e0cbc9e9385da4571725c94114ca821e197553
Methods	Manual review & testing

Scope

src/actions/SwapActions.sol
src/staking/HyperStaking.sol
src/staking/MatrixVault.sol
src/libs/GrowthMathLib.sol
src/Auction.sol
src/BuyAndBurn.sol
src/MatriX.sol

Issues Found

High risk	0
Medium risk	1
Low risk	3

5 Findings

5.1 Medium

5.1.1 Incorrect growth curve after the 88th day

Severity: Medium

Context: Auction.sol

Description: The unlocking of the patience pool occurs gradually and is divided into three distinct periods:

- Day 0 to Day 28: The unlocked percentage increases linearly from 0 to 1%.
- Day 28 to Day 88: The unlocked percentage increases linearly from 1 to 15%.
- Day 88 to Day 369: The unlocked percentage increases exponentially from 15% to 100%.

The issue in the code arises when calculating the percentage for the third period:

```
percentageUnlocked =
    GrowthMathLib.exponentialGrowth(timeElapsed, 369 days, 0.15e18, 1e18);
```

The second argument passed to the *exponentialGrowth* function is expected to represent the duration of the third period during which the percentage grows exponentially. Since the total duration of all three periods is intended to be 369 days, the third period's duration should be calculated as 369 - 88 = 281 days.

Additionally, the first argument is expected to reflect the time elapsed since the start of the third period. However, the variable *timeElapsed* represents the time elapsed since the beginning of the first period (Day 0).

As a result, instead of the patience pool unlock percentage beginning to grow exponentially from 15% after Day 88, it incorrectly starts at approximately 23.58%.

Recommendation: To resolve this issue, consider making the following adjustment and improving test coverage:

```
percentageUnlocked =
    GrowthMathLib.exponentialGrowth(timeElapsed - 88 days, 281 days, 0.15e18, 1e18);
```

Resolution: Resolved. Worth noting: an additional adjustment was made to update the percentage from 15% to 28%.

5.2 Low

5.2.1 Overflow may cause unexpected behavior

Severity: Low

Context: Auction.sol

Description: When determining what auction cycle we are currently at, the following calculation is made:

```
currentAuction = uint8(timeElapsedSince / Constants.GAP_BETWEEN_AUCTIONS) + 1;
```

The problem is that once *timeElapsedSince / Constants.GAP_BETWEEN_AUCTIONS* becomes > 255, the downcast will silently overflow and restart the counting. Therefore, the returned *currentAuction*, *startTime* and *endTime* will be incorrect at the given timestamp.

Recommendation: Consider removing the downcast.

Resolution: Resolved.

5.2.2 Leftover tokens when providing liquidity due to slippage

Severity: Low

Context: Auction.sol

Description: In *addLiquidityToHyperMatrixPool*, if any matrix amount was not taken by the pool to add as liquidity, the leftover would be locked forever in the Auction contract.

Recommendation: Consider burning the remaining matrix amount if there's any leftover due to slippage.

Resolution: Acknowledged.

5.2.3 Insufficient input validation in privileged setter methods

Severity: Low

Context: MatrixVault.sol

Description: The following method need to have an upper cap value enforced to ensure no malicious setting is possible:

- *changeBuyActionState* - prevent too high *_s.intervalBetween* (such that causes underflow)

Recommendation: Consider implementing stricter input validation.

Resolution: Acknowledged.

5.3 Informational

5.3.1 Typographical mistakes, non-critical issues and code-style suggestions

Severity: Informational

Context: src/*

Description: The contracts contains one or more typographical mistakes, non-critical issues and code-style suggestions. In an effort to keep the report size reasonable, we enumerate these below:

1. Do not allow 0-amount deposits in the MatrixAuction.
2. The Matrix *TOTAL_SUPPLY* does not consider the amount that is minted when providing liquidity to the Uniswap pool.
3. Missing *notExpired(_deadline)* modified on *buyTitanX* and *buyHyper*.

4. The *FullMath* library and *pool* constant in *MatriX* are never used.
5. *getAuctionAt* will return first auction if *block.timestamp* < *startTimestamp* instead of null.
6. *_duration* > *_upperBoundDays* case not handled in *linearInterpolation*.
7. Misleading comments for *exponentialGrowth* - *20e17* should be *20e16*, *~44.074e17* should be *~44.231e16*.

Recommendation: Consider fixing the above typographical mistakes, non-critical issues and code-style suggestions.

Resolution: Partially resolved.