GA GUARDIAN

M0

EVM-M Extensions Review

Security Assessment

December 12th, 2025

Summary

Audit Firm Guardian

Prepared By Cosine, Mark Jonathas, Zdravko Hristov

Client Firm M0

Final Report Date December 12, 2025

Audit Summary

M0 engaged Guardian to review the security of their EVM-M Extensions. From the 10th of November to the 12th of November, a team of 3 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking

Guardian assigns a Confidence Ranking of 5 to the protocol. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

Note: Fixes to the findings uncovered in the remediation review have not been reviewed by Guardian.

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. **Recommendation:** Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	Code is clean, well-structured, and adheres to best practices. Only 1 Significant issue was uncovered per week. Design patterns are sound, and test coverage is strong. **Recommendation:** Suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity issues. Varied Low severity findings.
3: Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases. **Recommendation:** Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1-2 High/Critical findings per engagement week.
2: Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities. Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. **Recommendation:** Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week. Or additional High/Critical findings uncovered in remediation review which have not been resolved and confirmed by Guardian.
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. **Recommendation:** Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Overview

Project Summary

Project Name	M0
Language	Solidity
Codebase	https://github.com/m0-foundation/evm-m-extensions
Commit(s)	Main Review commit: 49cc37acbb21d414dd64a8116504f4f747cc1c84 Remediation Review commit: fcaaee0d107f61dd2e8b445d83d89716f87c23e9

Audit Summary

Delivery Date	December 12, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	0	0	0	0	0	0
Medium	1	0	0	1	0	0
Low	5	0	0	2	0	3
Info	10	0	0	7	0	3

Audit Scope & Methodology

contract,source,total,comment
common/src/libs/TransferHelper.sol,32,58,20
evm-m-extensions/src/components/pausable/Pausable.sol,17,44,17
evm-m-extensions/src/projects/jmi/JMIExtension.sol,142,343,134
source count: {
  total: 445,
  source: 191,
  comment: 171,
  single: 56,
  block: 115,
  mixed: 0,
  empty: 83,
  todo: 3,
  blockEmpty: 0,
  commentToSourceRatio: 0.8952879581151832
}

As well as the changes to the following files from the previous audit:
Freezable.sol
MYieldToOne.sol
SwapFacility.sol

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of M0, fuzz-testing was performed on the protocol's main functionalities. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 10,000,000+ runs with a prepared fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
GLOB-01	JMI.totalAssets ≤ JMI.totalSupply	✅	✅	✅	10M+
GLOB-02	JMI.totalSupply ≤ JMI.totalAssets + M.balanceOf(JMI)	✅	✅	✅	10M+

Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	Trusted Routers Cannot Execute Actions	DoS	Medium	Acknowledged
L-01	Missing _beforeApprove Overwrite	Validation	Low	Acknowledged
L-02	Asset Donations Can Lead To Underflow	DoS	Low	Resolved
L-03	_replaceAssetWithM And _wrapCalc Can Round To 0	Validation	Low	Resolved
L-04	No Extension Validation	Unexpected Behavior	Low	Acknowledged
L-05	MToken Can Be Wrapped	Unexpected Behavior	Low	Resolved
I-01	Arbitrage Opportunity On Depeg	Warning	Info	Acknowledged
I-02	replaceAssetWithM Can Be Misused	Warning	Info	Resolved
I-03	Precision Loss On Transfers Can Be Improved	Rounding	Info	Acknowledged
I-04	Approvals For Frozen Account Cannot Be Cleared	DoS	Info	Acknowledged
I-05	Recipient Of Replaced Assets May Be Frozen	Validation	Info	Acknowledged

M-01 | Trusted Routers Cannot Execute Actions

Category	Severity	Location	Status
DoS	Medium	SwapFacility.sol	Acknowledged

Description

Whenever a function guarded with the `isNotLocked` modifier is called, `Locker.set(caller_)` is executed. If `msg.sender` is a trusted router, `caller_` is set to whatever address the router returns. Otherwise, the real `msg.sender` is used.

However, `SwapFacility` uses `msg.sender` in all of its functions. For example, it will try transferring assets out of `msg.sender`, etc... This will result in reverts when the function is called by a trusted router since it's expected to only forward the call, not pay for the transactions.

Furthermore, `msg.sender` is passed to the `permit()` calls as well, meaning it has to be the signer of the message, which is impossible for the router.

There is also asymmetry existing when calling `JMI.wrap()`, because it uses `ISwapFacility(msg.sender).msgSender()`, which will be the correct forwarded sender.

Recommendation

Replace the `msg.sender` usage in `SwapFacility` with `msgSender()`.

Resolution

M0 Team: Acknowledged.

L-01 | Missing _beforeApprove Overwrite

Category	Severity	Location	Status
Validation	Low	JMIExtension.sol	Acknowledged

Description

The `JMIExtension` overwrites all the `MYieldToOne` hooks which reverts if a given account is frozen to also add the `_requireNotPaused` check, except for the `_beforeApprove` hook.

Recommendation

Consider to also overwrite and add the `_requireNotPaused` to the `_beforeApprove` hook.

Resolution

M0 Team: Acknowledged.

L-02 | Asset Donations Can Lead To Underflow

Category	Severity	Location	Status
DoS	Low	JMIExtension.sol: 295-298	Resolved

Description

The `JMIExtension` contract uses `balanceOf` to perform checks related to the asset amounts deposited instead of a internal mapping. On the other hand it uses the variable `totalAssets` to store the total amount of all deposited non-M-token assets.

This can lead to DoS and stuck funds if someone would transfer assets into the contract directly. For example:
• `totalAssets` = 0
• Two assets are whitelisted (Asset1 & Asset2)
• Bob deposits 100 Asset1 and 100 Asset2
• `totalAssets` = 200
• Eve transfers 50 Asset1 into the contract
• Eve calls `replaceAssetWithM` to get 150 Asset1 out of the contract
• `totalAssets` = 200 - 150 = 50
• Now it's not possible to get the 100 Asset2 out of the contract as `totalAssets` would underflow in that case

Another point is that asset caps can be consumed without wrapping.

For a wrap to succeed, the asset cap of the asset being wrapped should not be exceeded.

```
assetCap(asset) >= (IERC20(asset).balanceOf(address(this)) + amount)
```

Because `balanceOf()` is used, anyone can transfer asset tokens to the contract and consume the cap. The tokens cannot be utilized since `totalAssets` won't be modified.

Recommendation

Consider to implement a internal mapping instead to track asset amounts.

Resolution

M0 Team: The issue was resolved in PR#15.

L-03 | _replaceAssetWithM And _wrapCalc Can Round To 0

Category	Severity	Location	Status
Validation	Low	JMIExtension.sol: 293	Resolved

Description

The `_fromExtensionToAssetAmount` calculation in the `_replaceAssetWithM` function could round to 0 in the edge case that the given asset has less decimals than the $M token and a dust amount is given.

This would result in the user losing $M tokens and decreasing the `totalAssets` without receiving any assets in return.

The same behavior is also present in the `_wrap()` flow, where users can send assets, but have 0 extension tokens minted for them.

```
uint256 jmiAmount_ = _fromAssetToExtensionAmount(asset, amount);
...
_mint(recipient, jmiAmount_);
```

Recommendation

Consider reverting in both cases if the resulting amount is 0.

Resolution

M0 Team: The issue was resolved in PR#10.

L-04 | No Extension Validation

Category	Severity	Location	Status
Unexpected Behavior	Low	SwapFacility.sol: 499	Acknowledged

Description

Before interacting with the system, an extension has to be either approved by an admin or be an approved earner.

```
function isApprovedExtension(address extension) public view returns (bool) {
return _isApprovedEarner(extension) || isAdminApprovedExtension(extension);
}
```

The `_isApprovedEarner()` function will return `true` for any address if the earners list is being ignored.

```
function _isApprovedEarner(address extension) private view returns (bool) {
return
IRegistrarLike(registrar).get(EARNERS_LIST_IGNORED_KEY) != bytes32(0) ||
IRegistrarLike(registrar).listContains(EARNERS_LIST_NAME, extension);
}
```

Therefore, whenever the earners list is ignored, any arbitrary address can be used for an extension. This allows emitting `Swapped`, `SwappedOutM` and `JMIAssetReplaced()` events with arbitrary addresses and poses risks if the code changes in future.

Recommendation

Consider excluding the `EARNERS_LIST_IGNORED_KEY` part from `_isApprovedEarner()` and managing the swappers via approvals.

Resolution

M0 Team: Acknowledged.

L-05 | MToken Can Be Wrapped

Category	Severity	Location	Status
Unexpected Behavior	Low	SwapFacility.sol: 285	Resolved

Description PoC

To wrap an mToken into JMIExtension users must call SwapFacility.swap() and specify mToken as tokenIn and JMIExtension as tokenOut. This will execute the first branch in _swap():

```
if (tokenIn == mToken) return _swapInM(tokenOut, amount, recipient);
```

The _swapInM() function is not adjusted to support JMIExtension, instead it used the generic MExtension interface to call wrap().

```
IMExtension(extensionOut).wrap(recipient, amount);
```

Because JMIExtension inherits from MExtension, the call will succeed, but MExtension.wrap() will be executed instead of JMIExtension.wrap() There the internal _wrap(address,address,uint256) function is executed.

This function is different than the _wrap(address,address,address,uint256) function defined in JMIExtension . From there on, the whole wrap is executed by the logic defined in MExtension instead of JMIExtension, which means the empty _beforeWrap() function of MExtension is executed and the following validation, including the pause check, is not performed:

```
function _beforeWrap(address asset, address account, address recipient, uint256 amount) internal view
virtual {
_requireNotPaused();
if (!isAllowedAsset(asset)) revert AssetNotAllowed(asset);
if (!isAllowedToWrap(asset, amount)) revert AssetCapReached(asset);
super._beforeWrap(account, recipient, amount);
}
```

Since asset here is mToken, skipping the two if statements is fine, but not calling _requireNotPaused() means the code doesn't respect the paused flag at all.

In result, wrapping mTokens into the JMIExtension will be possible even when it's in a paused state.

Recommendation

Call the correct function If tokenOut in SwapFacility.swapInM() is the JMIExtension .

Resolution

M0 Team: The issue was resolved in PR#9.

I-01 | Arbitrage Opportunity On Depeg

Category	Severity	Location	Status
Warning	Info	JMIExtension.sol	Acknowledged

Description

If one of the allowed assets (stablecoins) in the `JMIExtension` would lose in value an arbitrage opportunity arises:

• Traders buys assets for $0.9

• Trader deposit the assets into the `JMIExtension`

• Trader swaps the `JMIExtension` into a safe one

• Repeat

Recommendation

Be aware of this risk and react as quickly as possible in that case by pausing the contracts and setting the cap of the given asset to zero.

Resolution

M0 Team: Acknowledged.

I-02 | replaceAssetWithM Can Be Misused

Category	Severity	Location	Status
Warning	Info	SwapFacility.sol: 123-131	Resolved

Description

The `replaceAssetWithM` flow does not check if the given asset was whitelisted (cap > 0).

This function can therefore be used to swap any M0 extension to any token in the `JMIExtension`.

If by accident a token with higher value would land inside the contract any user could claim it by for example performing a $1 M0 Extension against 1 WETH swap.

Recommendation

Be aware that this function can be used in that way and document this behavior.

Or consider to only allow the withdraw of whitelisted tokens to decrease potential attack surface. In this case however a new whitelist would make more sense than the current cap logic as otherwise it would no longer be possible to prevent further deposits while still allowing withdraws.

Resolution

M0 Team: The issue was resolved in PR#15.

I-03 | Precision Loss On Transfers Can Be Improved

Category	Severity	Location	Status
Rounding	Info	Global	Acknowledged

Description

Whenever an extension is wrapping `mTokens`, it pulls `amount` of these `mTokens` from the sender and mints the exact same amount of extension tokens. However, due to the index tracking behavior of the `MToken` , transferring `X` amount can result in less than `X` tokens received.

```
IMTokenLike(mToken).transferFrom(msg.sender, address(this), amount);
_mint(recipient, amount);
```

In the case of `wrap()`, the system will mint more extension tokens that the received `mTokens`, which means some of the extension tokens will not be redeemable. The same is true for `_replaceAssetWithM()` .

The opposite rounding problem exists in `_unwrap()` , there the amount being subtracted is rounded up and because of that the system will be potentially losing value with each transfer.

Recommendation

There are already comments in the `MExtension` acknowledging the problem. While it's hard to be solved for the `_unwrap()` case, you can add balance delta tracking for `wrap()` and `_replaceAssetWithM()` to avoid some of the losses.

Resolution

M0 Team: Acknowledged.

I-04 | Approvals For Frozen Account Cannot Be Cleared

Category	Severity	Location	Status
DoS	Info	MYieldToOne.sol: 168	Acknowledged

Description

The approve hook reverts if the spender is frozen, regardless of the allowance amount. This prevents users from setting allowance to zero for spenders that later became frozen, effectively trapping their prior approvals until the spender is unfrozen.

Recommendation

Allow `approve(owner, spender, 0)` to succeed even if the spender is frozen.

Resolution

M0 Team: Acknowledged.

I-05 | Recipient Of Replaced Assets May Be Frozen

Category	Severity	Location	Status
Validation	Info	JMIExtension.sol: 285-307	Acknowledged

Description

The `JMIExtension` enforces freeze checks for wrap/unwrap/transfer via `MYieldToOne` hooks, but `_replaceAssetWithM()` does not validate whether the recipient is frozen. A frozen account can therefore receive allowed assets directly through `replaceAssetsWithM()`

Recommendation

If this is not expected, revert if the recipient is frozen.

Resolution

M0 Team: Acknowledged.

Remediation Findings & Resolutions

ID	Title	Category	Severity	Status
I-01	Decimals Can't Be Refetched	Warning	Info	Acknowledged
I-02	Accidental Asset Transfers Are Lost Now	Warning	Info	Acknowledged
I-03	canSwapViaPath() May Revert	Warning	Info	Acknowledged
I-04	Unnecessary Assignments	Best Practices	Info	Resolved
I-05	Unsafe Uint240 Casting	Informational	Info	Resolved

I-01 | Decimals Can't Be Refetched

Category	Severity	Location	Status
Warning	Info	JMIExtension.sol: 180	Acknowledged

Description

The `setAssetCap` saves the given assets decimals and it is not possible to update them in case the decimals of the given `ERC20` token would ever change.

Recommendation

Be aware of that and consider to refetch the decimals every time the `setAssetCap` function is called to be able to react on such a change.

Resolution

M0 Team: Acknowledged.

I-02 | Accidental Asset Transfers Are Lost Now

Category	Severity	Location	Status
Warning	Info	JMIExtension	Acknowledged

Description

Now asset balances are tracked with a internal balance. Therefore if now non $M token assets are accidentally transferred into the `JMIExtension` contract they are lost.

Recommendation

Consider to add a sweep function to be able to withdraw these assets.

Resolution

M0 Team: Acknowledged.

I-03 | canSwapViaPath() May Revert

Category	Severity	Location	Status
Warning	Info	SwapFacility.sol: 263-268	Acknowledged

Description

A `try/catch` was added to `SwapFacility.canSwapViaPath()` to fetch the paused status of the two tokens.

```
// If contracts are paused, return false
try Pausable(tokenIn).paused() returns (bool tokenInPaused) {
isTokenInPaused = tokenInPaused;
} catch {}
try Pausable(tokenOut).paused() returns (bool tokenOutPaused) {
isTokenOutPaused = tokenOutPaused;
} catch {}
```

Because the addresses are arbitrary, any contract (or EOA with code) can be passed as a token, return a data that cannot be decoded to a `bool` and make the call to `canSwapViaPath()` to revert. This can lead to DOS for external integrators that expect the function to never revert.

Recommendation

Document the risk of reverting.

Resolution

M0 Team: Acknowledged.

I-04 | Unnecessary Assignments

Category	Severity	Location	Status
Best Practices	Info	SwapFacility.sol: 256-257	Resolved

Description

The following two variables in the SwapFacility.canSwapViaPath() are assigned their default values (false for bool), which is unnecessary.

```
bool isTokenInPaused = false;
bool isTokenOutPaused = false;
```

Recommendation

Consider removing the assignments.

Resolution

M0 Team: The issue was resolved in PR#26.

I-05 | Unsafe Uint240 Casting

Category	Severity	Location	Status
Informational	Info	JMIExtension.sol	Resolved

Description

The balance of the tokens in the `Asset` struct is stored as `uint240`. The comment states that M token's supply won't exceed `uint240`, so balance should be safe.

```
// M token's supply can't exceed uint240, so uint240 is safe to use.
uint240 balance;
```

However, this balance doesn't track M assets, but every other enabled asset. In the `_wrap()` function the `amount` of tokens transferred is unsafely cast to `uint240`.

If the token is non-standard , i.e has very big precision, and the asset cap allows it, the amount used may exceed `uint240` and result in wrong accounting.

Recommendation

1. Fix the comment to explain the right reason why using `uint240` is safe

2. Make sure to not use assets which can have such big supplies

3. Configure asset caps with this in mind

Resolution

M0 Team: The issue was resolved in PR#25.

Disclaimer

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 Guardian 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. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’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 Guardian is subject to dependencies and 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.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits