GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

Abracadabra Money

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, 0xKato

Client Firm Abracadabra Money

Final Report Date November 14, 2023

Audit Summary

Abracadabra Money engaged Guardian to review the security of its GMX V2 Market Integration. From the 26th of October to the 7th of November, a team of 3 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Arbitrum**

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

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/AbracadabraTests

Table of Contents

Project Overview

Project Summary

Project Name	Abracadabra Money
Language	Solidity
Codebase	https://github.com/Abracadabra-money/abracadabra-money-contracts
Commit(s)	Initial: f257c00412cb762a4de4c9ec195fe7241006d31d Final: 46dbe2efd236de23b02e4ef8d0723633cf291700

Audit Summary

Delivery Date	November 14, 2023
Audit Methodology	Static Analysis, Manual Review, Test Suite

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	2	0	0	0	0	2
High	2	0	0	1	0	1
Medium	10	0	0	6	0	4
Low	4	0	0	2	0	2

Audit Scope & Methodology

Vulnerability Classifications

Vulnerability Level	Classification
🔴 Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
🟠 High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
🟡 Medium	Inherent risk of future exploits that may or may not impact the smart contract execution.
🟢 Low	Minor deviation from best practices.

Methodology

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.

Findings & Resolutions

ID	Title	Category	Severity	Status
ORDA-1	Wrong amount deposited into the DegenBox	Logical Error	Critical	Resolved
GMXC-1	Liquidations Prevented By Request Expiration	Protocol Manipulation	Critical	Resolved
GMXC-2	amountToAdd Not Valued At The Collaterization Rate	Logical Error	High	Resolved
GMOCL-1	longToken Assumed To Be The indexToken	Logical Error	High	Acknowledged
GMXC-4	Insolvent Liquidations Revert	Liquidations	Medium	Acknowledged
ORDA-2	Markets With ETH As The shortToken Are Gameable	Protocol Manipulation	Medium	Resolved
GMXC-5	Blacklisted Callee Incorrectly Reset	Logical Error	Medium	Resolved
ORDA-3	Anyone May Withdraw Funds From Order After Close	Access Control	Medium	Resolved
GMOCL-2	Use Of Deprecated latestAnswer Function	Deprecation	Medium	Acknowledged
GMOCL-3	More Restrictive PnL Type Used	Protocol Risk	Medium	Acknowledged
ORDA-4	minOut Applies To Terminal Orders	Logical Error	Medium	Acknowledged
GMXC-6	Unwieldy Collateral	Warning	Medium	Acknowledged
GMXC-7	Share Amount Provided As amountMarketToken	Logical Error	Medium	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
CAUL-1	Errant maxRate Validation	Validation	Medium	Resolved
ORDA-5	Init Function Called Multiple Times	Validation	Low	Resolved
ORDA-6	Hardcoded Gas Limit	Warning	Low	Resolved
CAUL-2	Lacking Caps On Init	Validation	Low	Acknowledged
GMXC-8	Pending Orders Can Be Closed	Validation	Low	Acknowledged

ORDA-1 | Wrong amount deposited into the DegenBox

Category	Severity	Location	Status
Logical Error	● Critical	GmxV2CauldronOrderAgent.sol: 197	Resolved

Description

In the `sendValueInCollateral` function the amount provided as a parameter is a `collateralShare` amount (GM shares in the degen box), that amount is then converted to a `shortToken` amount via an exchange rate.

The `amountShortToken` is then sent to the `degenBox`, however the amount (GM shares) is what is deposited.

Therefore a GM token Shares amount will be treated as a short token amount, which may have severely different valuations depending on the exchange rate. Additionally this can often cause the liquidation to revert if the GM token is a lower value than the short token.

Recommendation

In the `sendValueInCollateral` function, line 197:

`degenBox.deposit(IERC20(shortToken), address(degenBox), recipient, amount, 0);`

Should be replaced with:

`degenBox.deposit(IERC20(shortToken), address(degenBox), recipient, amountShortToken, 0);`

Resolution

Abracadabra Team: Remediated with commit: bda0214.

GMXC-1 | Liquidations Prevented By Request Expiration

Category	Severity	Location	Status
Protocol Manipulation	● Critical	GmxV2CauldronV4.sol: 135	Resolved

Description

During liquidation, if a user has an open order, the order is cancelled in order to retrieve the underlying tokens. However orders cannot be cancelled while within the `REQUEST_EXPIRATION_BLOCK_AGE`.

Therefore a malicious user can front-run liquidations and create an order to prevent liquidations. The `REQUEST_EXPIRATION_BLOCK_AGE` is currently configured to 1200 blocks on Arbitrum. In practice, the time to execute an order in GMX should be ~2 seconds, however when the deposit or withdrawal feature is disabled on GMX the keeper will not execute or cancel orders.

Therefore a malicious user can gain a ~5 minute grace period where they cannot be liquidated due to the `REQUEST_EXPIRATION_BLOCK_AGE` when the deposit or withdrawal feature is disabled as the order will not be executed or cancelled by the keeper.

Recommendation

Do not allow the creation of orders when an account is liquidatable.

Additionally, consider validating that the deposit and withdrawal feature are enabled before users are able to use the `ACTION_CREATE_ORDER`.

Resolution

Abracadabra Team: An insolvency check upon order creation was added in commit debb03.

Validation that deposit and withdrawal execution is not disabled was added in commit 9f2aa9.

GMXC-2 | amountToAdd Not Valued At The Collaterization Rate

Category	Severity	Location	Status
Logical Error	High	GmxV2CauldronV4.sol: 62-70	Resolved

Description

When validating whether a borrow position is solvent, the `COLLATERIZATION_RATE` is used to verify that the amount borrowed does not exceed the maximum percentage borrowable with the current collateral.

However, the `COLLATERIZATION_RATE` is not applied to `amountToAdd`, which is the amount of GM tokens expected to be obtained from a pending deposit.

For example, assume MIM and GM have a 1-1 price and the `COLLATERIZATION_RATE` is 75%. 10 GM tokens would allow borrowing 10 MIM rather than 7.5.

As a result, the amount of MIM able to be borrowed is calculated to be larger than allowed, and the position is deemed solvent.

Recommendation

Value the `amountToAdd` at the `COLLATERIZATION_RATE`.

Resolution

Abracadabra Team: Resolved in commit 2c2d6c0.

GMOCL-1 | longToken Assumed To Be The indexToken

Category	Severity	Location	Status
Logical Error	High	GmOracleWithAggregator.sol: 61	Acknowledged

Description

In the `_get` function the price of the long token provided is the price of the index token, however not all GM markets have the long token as the index token.

For example, the DOGE/USD GM market has Ether as its long token. In this case the price of ETH is provided as the price of DOGE which will be dramatically inaccurate.

Recommendation

Treat the long token separately from the index token, as these two are not guaranteed to be the same.

Resolution

Abracadabra Team: With the planned supported GM markets this oracle configuration is expected, however if we choose to support the DOGE or other similar gm markets in the future we will make a change.

GMXC-4 | Insolvent Liquidations Revert

Category	Severity	Location	Status
Liquidations	Medium	GmxV2CauldronV4.sol: 152	Acknowledged

Description

Insolvent liquidations cannot occur if the Order contract does not have enough to cover the outstanding amount.

It is unlikely that positions are able to become insolvent as this would require unexpected immediate price action or inaction from liquidators, however it would be prudent to be able to handle such a case.

Recommendation

Consider implementing logic such that insolvent positions may be liquidated successfully.

Resolution

Abracadabra Team: We can do partial liquidations to avoid this.

ORDA-2 | Markets With ETH As The shortToken Are Gameable

Category	Severity	Location	Status
Protocol Manipulation	Medium	GmxV2CauldronOrderAgent.sol: 330	Resolved

Description

In the event where a homogenous market or any GMX market had WETH as a short token, it would be possible for the user to withdraw their backing tokens through the `refundWETH` function, without going through the expected `ACTION_WITHDRAW_FROM_ORDER` cook action, therefore avoiding the solvency check and allowing for a user to cause their position to go insolvent.

Recommendation

Such a market is unlikely to exist, however it should be explicitly stated that markets with Ether as the short token are incompatible with the system.

Resolution

Abracadabra Team: Fixed with commit e265a7.

GMXC-5 | Blacklisted Callee Incorrectly Reset

Category	Severity	Location	Status
Logical Error	Medium	GmxV2CauldronV4.sol: 201	Resolved

Description

In the `closeOrder` function the `blacklistedCallees` mapping entry is set to `false` for the user's order, however the user's order was just set to the zero address and so therefore the previous order address entry was not correctly set to `false`.

Recommendation

Update the `blacklistedCallees` mapping before zeroing out the user's order.

Resolution

Abracadabra Team: Fixed with commit 623f7d.

ORDA-3 | Anyone May Withdraw From Order After Close

Category	Severity	Location	Status
Access Control	Medium	GMXV2CauldronOrderAgent.sol: 186	Resolved

Description

If a user were to ever close their Order but still have some funds remaining in the Order, anyone would be able to `ACTION_CALL` the `withdrawFromOrder` function from the Cauldron and take those funds. This can occur if a user were to call `withdrawFromOrder` without their entire amount and set `closeOrder = True`.

Recommendation

Consider forcing a user to retrieve their entire `shortToken` and `WETH` balance prior to closing an order with the `withdrawFromOrder` function.

Otherwise, clearly document that users should not leave their funds in an Order after closing it, as it becomes unblacklisted and accessible for all.

Resolution

Abracadabra Team: Fixed with commit c1fdaf.

GMOCL-2 | Use Of Deprecated latestAnswer Function

Category	Severity	Location	Status
Deprecation	Medium	GmOracleWithAggregator.sol: 54	Acknowledged

Description

In the `_get` function, the `latestAnswer` function is used to read the latest price from the Chainlink `indexAggregator` and `shortAggregator`.

However the `latestAnswer` function is deprecated and should be replaced by a `latestRoundData` call with heartbeat validation as well as a sequencer uptime check for Arbitrum.

Recommendation

Use the latestRoundData function to fetch the latest price from Chainlink and implement the necessary heartbeat and sequencer uptime validations.
Blockchain Oracles for Connected Smart Contracts | Chainlink Documentation
Chainlink Data Feeds Documentation | Chainlink Documentation

Resolution

Abracadabra Team: Acknowledged.

GMOCL-3 | More Restrictive PnL Type Used

Category	Severity	Location	Status
Protocol Risk	Medium	GmOracleWithAggregator.sol: 12	Acknowledged

Description

In the `GmOracleWithAggregator` contract the reported price uses the `MAX_PNL_FACTOR_FOR_TRADERS` `PNL_TYPE` to read the market token price from GMX. However this PnL type is less constrictive on the trader PnL than the `MAX_PNL_FACTOR_FOR_DEPOSITS`.

Therefore when the market PnL is in between these two factors like so: `MAX_PNL_FACTOR_FOR_TRADERS < PnL in market < MAX_PNL_FACTOR_FOR_DEPOSITS`

Then the resulting price of the market token will be higher when measured using the more constrictive PnL factor for traders. Therefore the user's collateral will be valued higher using the PnL factor for traders than compared to the PnL factor for deposits.

Out of an abundance of caution, it may be preferable to use the less constrictive max PnL for deposits, so that the collateral is not optimistically valued by capping the PnL to a lower amount. This is somewhat arbitrary as GM tokens cannot be redeemed as long as the pnl to pool ratio exceeds the `MAX_PNL_FACTOR_FOR_WITHDRAWALS`, which is the lowest of these PnL factors.

Recommendation

Consider using the less constrictive `MAX_PNL_FACTOR_FOR_DEPOSITS` to read the price of the GM token.

Resolution

Abracadabra Team: Acknowledged.

ORDA-4 | minOut Applies To Terminal Orders

Category	Severity	Location	Status
Logical Error	Medium	GmxV2CauldronOrderAgent.sol: 213	Acknowledged

Description

In the `orderValueInCollateral` function the `minOut` and `minOutLong` values are considered regardless of if the order is pending or if it has reached a terminal status.

A deposit may have been cancelled or a withdrawal may have been executed, and therefore the Order contract has a distinct amount of short tokens, however the collateral value is still based on the minimum output that was configured for the order.

This directly discounts a user's collateral as they will often receive more than the configured minimum amount.

Recommendation

If the order has reached a terminal status, return the balance of short tokens converted to market tokens as the `orderValueInCollateral`.

Resolution

Abracadabra Team: We want to incentivize users to close orders if they are completed and wanted to limit complexity.

GMXC-6 | Unwieldy Collateral

Category	Severity	Location	Status
Warning	Medium	GmxV2CauldronV4.sol: 187	Acknowledged

Description

There are several characteristics of GM tokens that make them less than ideal as collateral.

Firstly, liquidations may be somewhat unwieldy for the liquidator as there is currently no market to swap GM tokens and unwrapping the GM tokens cannot occur in a single liquidation transaction.

Additionally, if the `pnlToPoolFactor` in the GMX system is above the `MAX_PNL_FACTOR_FOR_WITHDRAWALS` GM tokens cannot be redeemed for their backing `longTokens` or `shortTokens`.

Recommendation

No changes may be necessary, simply be aware of this unwieldiness for liquidators, ensure that liquidators are still properly incentivized and consider this when determining the `COLLATERIZATION_RATE` for GM tokens.

Resolution

Abracadabra Team: Acknowledged.

GMXC-7 | Share Amount As amountMarketToken

Category	Severity	Location	Status
Logical Error	Medium	GmxV2CauldronV4.sol: 152	Resolved

Description

In the `liquidate` function, when additional value is necessary to cover the borrowed amount and the `Order.sendValueInCollateral` function is called, the provided `amountMarketToken` is computed from `collateralShare - userCollateralShare[user]`. However this is a share amount rather than an elastic market token amount.

Currently there is no strategy for GM tokens in the `DegenBox`, however if there were to be a strategy that increased the elastic supply of GM tokens relative to the base shares, then the share value provided to the `Order.sendValueInCollateral` would be inaccurate.

Recommendation

Consider converting the outstanding `collateralShare` amount to a market token amount with the `DegenBox.toAmount` function before providing this amount as the `amountMarketToken` for the `sendValueInCollateral` function.

Resolution

Abracadabra Team: Fixed with commit 431592.

CAUL-1 | Errant maxRate Validation

Category	Severity	Location	Status
Validation	Medium	CauldronV4.sol: 490	Resolved

Description

In the `cook` function the `ACTION_UPDATE_EXCHANGE_RATE` action includes a `minRate` and `maxRate` to bound the allowed updated `rate`.

The `rate` is validated to be `> minRate` as well as `> maxRate` if a `maxRate` is set. However this misconstrues the meaning of a `maxRate` as the `rate` should be validated to be `< maxRate`.

Recommendation

Validate that the `rate` is `< maxRate` when the `maxRate ≠ 0`.

Resolution

Abracadabra Team: Fixed with commit 55c602.

ORDA-5 | Init Function Called Multiple Times

Category	Severity	Location	Status
Validation	Low	GmxV2CauldronOrderAgent.sol: 133	Resolved

Description

The `init` function may not be called again if the cauldron address has been assigned a non-zero value, however there is no restriction that the `_cauldron` parameter is not the zero address.

Within the context of the `GmxV2CauldronOrderAgent` this is not an issue as the `_cauldron` address provided to the `init` function is always the `msg.sender` which cannot be the zero address.

However it may be prudent to add validation that the `_cauldron` parameter value is not the zero address to prevent vulnerabilities if the `GmxV2CauldronRouterOrder` contract were to be used in a different context in the future.

Recommendation

Consider adding validation such that the `_cauldron` address cannot be the zero address in the `init` function.

Resolution

Abracadabra Team: Fixed with commit b03d57.

ORDA-6 | Hardcoded Gas Limit

Category	Severity	Location	Status
Warning	Low	GmxV2CauldronOrderAgent.sol: 72	Resolved

Description

In the `GmvV2CauldronOrderAgent` contract, the `CALLBACK_GAS_LIMIT` is hardcoded to `1_000_000`. However in some cases the GMX team may reduce the maximum allowed gas limit such that `1_000_000` exceeds the maximum cap.

In this case it may be better to allow a reconfiguration of the `CALLBACK_GAS_LIMIT` than to deploy a new `GmxV2CauldronRouterOrder` implementation.

Recommendation

Consider making the `CALLBACK_GAS_LIMIT` a configurable value by a trusted address. Otherwise be aware that the GMX team could reduce the maximum callback gas limit to below `1_000_000` which would prevent any deposits or withdrawals from being created.

Resolution

Abracadabra Team: Fixed with commit 965af4.

CAUL-2 | Lacking Caps On Init

Category	Severity	Location	Status
Validation	Low	CauldronV4.sol: 141	Acknowledged

Description

In the `init` function, there are no requirements that the assigned `INTEREST_PER_SECOND`, `COLLATERIZATION_RATE`, `LIQUIDATION_MULTIPLIER`, and `BORROW_OPENING_FEE` are within a reasonable range.

Recommendation

Consider adding validation in the future to restrict the possible values for these critical values.

Resolution

Abracadabra Team: This is a design choice.

GMXC-8 | Pending Orders Can Be Closed

Category	Severity	Location	Status
Validation	Low	GmxV2CauldronV4.sol: 80	Acknowledged

Description

With the `ACTION_WITHDRAW_FROM_ORDER` action it is possible to close an order in the Abracadabra system while the order in GMX is still pending.

If the order is already closed in the Abracadabra system, then during the `afterDepositExecution` or `afterWithdrawalCancellation` callbacks, the call to the `closeOrder` function will revert and prevent any logic executing on the Abracadabra side, yet the order will still be executed.

The `ACTION_WITHDRAW_FROM_ORDER` action must leave the position in a solvent state in Abracadabra so there is no imminent risk to the Abracadabra system. However this is an unusual edge case and can result in users losing funds, therefore it may make sense to explicitly disallow it.

Recommendation

Consider disallowing the `ACTION_WITHDRAW_FROM_ORDER` action when an order is active.

Resolution

Abracadabra Team: This is an edge case we expect only to be possible to custom frontends or direct contract interaction.

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 Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits 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