METALEX BORG SECURITY AUDIT REPORT

June 20, 2025

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals

- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

MetaLeX protocol allows for the customization of Gnosis Safe multisigs to be used as a committee for a DAO. This customization involves restricting actions for the specific multisig via a special Guard contract, which serves as a filter for all transactions from the multisig. Apart from the Guard contract, MetaLeX offers a list of different implants that can be installed into the Safe contract to extend its functionality. An example of such an extension is an implant that enables multisig owners to create grants.

1.4 Project Dashboard

Project Summary

Title	Description
Client	MetaLeX
Project name	Borg-core
Timeline	28.05.2024 - 18.06.2025
Number of Auditors	3

Project Log

Date	Commit Hash	Note
28.05.2024	9074503d37cfa1d777ef16f6c88b84c98b4f54eb	Commit for the audit
16.07.2024	b89ff5b7796047bbb6123c97192cfcff33e6b7f5	Commit for the re-audit
22.07.2024	3357105af365fa4a1de24c8a2c44369ccbbde059	Commit for the re-audit 2
02.09.2024	4fe452f5e0f9ffe610840c4d650ca38cbb772f5d	Commit with updates
03.09.2024	32a32c40be50afbfc4dd00d66ff8e8b34a511152	Commit for the re-audit 3
03.10.2024	ea40541219fb6dbedc63e7eb6760ab2059709204	Commit with updates 2
07.10.2024	2cc22ab162299e9678e8eaea01519b53bf650a5f	Commit for the re-audit 4
17.06.2025	b1a796a1da21fb5ecbc58ca84cfa39beb2aa2e21	Commit with updates 3
18.06.2025	b9d43386429e9fdd79fc4982678b88b39e3593fb	Commit for the re-audit 5

Project Scope

The audit covered the following files:

File name	Link
src/libs/conditions/balanceCondition.sol	balanceCondition.sol
src/libs/conditions/timeCondition.sol	timeCondition.sol
src/libs/conditions/baseCondition.sol	baseCondition.sol
src/libs/conditions/chainlinkOracleCondition.sol	chainlinkOracleCondition.sol
src/libs/conditions/signatureCondition.sol	signatureCondition.sol
src/libs/conditions/API3OracleCondition.sol	API3OracleCondition.sol
src/libs/conditions/conditionManager.sol	conditionManager.sol
src/libs/conditions/deadManSwitchCondition.sol	deadManSwitchCondition.sol
src/libs/governance/baseGovernanceAdapater.sol	baseGovernanceAdapater.sol
src/libs/governance/flexGovernanceAdapater.sol	flexGovernanceAdapater.sol
src/libs/auth.sol	auth.sol
src/implants/baseImplant.sol	baseImplant.sol
src/implants/optimisticGrantImplant.sol	optimisticGrantImplant.sol
src/implants/failSafeImplant.sol	failSafeImplant.sol
src/implants/daoVetoGrantImplant.sol	daoVetoGrantImplant.sol
src/implants/daoVoteGrantImplant.sol	daoVoteGrantImplant.sol
src/implants/ejectImplant.sol	ejectImplant.sol
src/baseGuard.sol	baseGuard.sol

File name	Link
src/borgCore.sol	borgCore.sol
src/libs/helpers/signatureHelper.sol	signatureHelper.sol
src/libs/hooks/baseRecoveryHook.sol	baseRecoveryHook.sol
src/libs/hooks/exampleRecoveryHook.sol	exampleRecoveryHook.sol
src/libs/hooks/exampleRecoveryHookRevert.sol	exampleRecoveryHookRevert.sol
src/libs/conditions/multiUseSignCondition.sol	multiUseSignCondition.sol
src/implants/daoVoteImplant.sol	daoVoteImplant.sol
src/implants/daoVetoImplant.sol	daoVetoImplant.sol
src/implants/sudoImplant.sol	sudoImplant.sol
src/libs/governance/snapShotExecutor.sol	snapShotExecutor.sol

Deployments

File name	Contract deployed on mainnet	Comment
borgCore.sol	0x0c05BF...65BEA40C	
ejectImplant.sol	0xb368BD...57D8FE3e	
sudoImplant.sol	0x957774...b4554804	
snapShotExecutor.sol	0xE050BD...Dd4e1bA2	
auth.sol	0x306897...1aBdeB3d	Auth for BorgCore
auth.sol	0xE7cF64...b69bc45D	Auth for snapShotExecutor

File name	Contract deployed on mainnet	Comment
auth.sol	0x2Ff67d...983Ef23B	Auth for ejectImplant and sudoImplant

1.5 Summary of findings

Severity	# of Findings
Critical	4
High	7
Medium	24
Low	72

ID	Name	Severity	Status
C-1	The governance adapter is not protected	Critical	Fixed
C-2	Possible overflow	Critical	Fixed
C-3	Incorrect proposal can be deleted inside `_deleteProposal`	Critical	Fixed
C-4	Modifier `conditionCheck` doesn't revert if no conditions passed	Critical	Fixed
H-1	Calls via `.call()` to internal functions `_deleteProposal`, `executeDirectGrant`, `executeSimpleGrant`, and `executeAdvancedGrant` will always fail	High	Fixed
H-2	Proposal duration is not set	High	Fixed
H-3	DoS of `cooldown`	High	Fixed
H-4	Reentrancy in `optimisticGrantImplant`	High	Fixed
H-5	Incorrect function signature in `daoVoteGrantImplant.proposeAdvancedGrant()` `daoVetoGrantImplant.proposeAdvancedGrant()`	High	Fixed
H-6	Incorrect loop boundary inside `removeContract`	High	Fixed

H-7	Always failing check inside the `checkCondition` for `chainlinkOracleCondition`	High	Fixed
M-1	Updating the policy for the existing method will lead to adding unnecessary `byteOffset` to the `paramOffsets` array	Medium	Fixed
M-2	Missing transferred value check	Medium	Fixed
M-3	Method policies are not cleaned on contract removal	Medium	Fixed
M-4	Missing `conditionCheck` modifiers	Medium	Fixed
M-5	A threshold check can prevent the owner from self-ejecting	Medium	Fixed
M-6	Chainlink returned data is not checked for being stale	Medium	Fixed
M-7	Conditions' order has impact	Medium	Acknowledged
M-8	API3 heartbeat	Medium	Fixed
M-9	The owner can remove themselves	Medium	Fixed
M-10	Tests do not work	Medium	Fixed
M-11	Seconds are used instead of days	Medium	Fixed
M-12	`duration` is not limited	Medium	Fixed
M-13	Proposals do not have an expiration date	Medium	Fixed
M-14	Too flexible logic	Medium	Fixed
M-15	Weak condition	Medium	Fixed
M-16	Backrun of partly set constraints	Medium	Fixed
M-17	Missing boundary check for `duration`	Medium	Fixed
M-18	The `borgMode` variable value should be immutable	Medium	Fixed

M-19	Parameter constraint is rewritten with empty values in some cases	Medium	Fixed
M-20	The native token transfer is not supported by `borgCore.checkTransaction()` in `blacklist` borg mode	Medium	Fixed
M-21	Unreachable Code in `updateThreshold` Function	Medium	Fixed
M-22	`borgCore._checkDirectorsSignatures()` calculates the number of directors' signatures incorrectly if the total number of signatures > threshold	Medium	Fixed
M-23	Missing Update of `lastProposalTime` in `proposeTransaction`	Medium	Fixed
M-24	Expired Proposal Can Still Be Executed	Medium	Fixed
L-1	Missing parameter checks	Low	Fixed
L-2	`matchNum.length` should be checked	Low	Fixed
L-3	`updatePolicy` parameters should be checked	Low	Fixed
L-4	`_recipient` zero address check	Low	Fixed
L-5	`_methodCallData` length check	Low	Fixed
L-6	`_condition` uniqueness check	Low	Fixed
L-7	`_condition` absence check	Low	Fixed
L-8	`RECOVERY_ADDRESS` zero address check	Low	Fixed
L-9	`addToken()` parameters are not checked	Low	Fixed
L-10	`quorum` and `threshold` not checked	Low	Acknowledged
L-11	`_duration` should be limited	Low	Fixed
L-12	`_governanceAdapter` and `_governanceExecutor` not checked	Low	Acknowledged
L-13	Proposal creator validation missing	Low	Fixed

L-14	`quorum` and `threshold` should be <= 100%	Low	Acknowledged
L-15	`_duration` and `_waitingPeriod` should be limited	Low	Acknowledged
L-16	`_governanceAdapter` is not checked	Low	Acknowledged
L-17	Grant amount spending limit check	Low	Acknowledged
L-18	Safe balance check	Low	Fixed
L-19	Proxy address validation	Low	Fixed
L-20	Token balance check	Low	Fixed
L-21	`isSigner` addresses uniqueness check	Low	Fixed
L-22	`_threshold` should be > 0	Low	Fixed
L-23	`BORG_SAFE` is not checked	Low	Fixed
L-24	Role validation check	Low	Acknowledged
L-25	Code improvements	Low	Fixed
L-26	`byteLength` adjustment	Low	Fixed
L-27	Exact match check	Low	Fixed
L-28	Remove unnecessary conversion	Low	Fixed
L-29	Simplify logical conditions	Low	Fixed
L-30	Full access policies in `updatePolicy`	Low	Fixed
L-31	Input parameter requirement removal	Low	Fixed
L-32	Legal agreement condition fix	Low	Fixed
L-33	`_byteOffset` missing check	Low	Fixed
L-34	Remove needless uint8 cast	Low	Fixed

L-35	Protection against underflow in cooldown check	Low	Fixed
L-36	Underflow protection in native cooldown check	Low	Fixed
L-37	Native token recovery in failSafe implant	Low	Fixed
L-38	FundsRecovered event corrections	Low	Fixed
L-39	Optimize `TokenInfo` struct	Low	Fixed
L-40	Unnecessary threshold check in the `ejectOwner`	Low	Fixed
L-41	Native token support in DAO implants	Low	Fixed
L-42	Balance check in `executeAdvancedGrant`	Low	Fixed
L-43	Overwritten initial values	Low	Fixed
L-44	Internal method for grant proposals	Low	Acknowledged
L-45	Separate methods for limit management	Low	Acknowledged
L-46	Remove unused errors	Low	Fixed
L-47	Unnecessary initialization of `newProposalId`	Low	Fixed
L-48	Setter for `metaVesTController`	Low	Fixed
L-49	Rename struct `prop` to `proposalDetail`	Low	Fixed
L-50	Remove unused `governanceExecutor`	Low	Fixed
L-51	`BORG_SAFE` rights extension	Low	Fixed
L-52	Optimization via internal methods	Low	Acknowledged
L-53	Restriction to `view`	Low	Fixed
L-54	Overflow optimization	Low	Fixed
L-55	Unnecessary initialization	Low	Fixed

L-56	`numSigners` optimization	Low	Fixed
L-57	Existing policies removal	Low	Fixed
L-58	Incorrect implementation of ERC165	Low	Fixed
L-59	Parameter constraints in `borgCore` work incorrectly in the case of `borgModes.blacklist`	Low	Acknowledged
L-60	A lack of `metaVesTController` zero checks in `optimisticGrantImplant's`, `daoVetoGrantImplant's`, and `daoVoteGrantImplant's` constructors	Low	Fixed
L-61	Incorrect comment to `borgCore.removePolicyMethod()`	Low	Fixed
L-62	`isMethodCallAllowed` should have `view` visibility	Low	Fixed
L-63	`nonReentrant` modifier	Low	Fixed
L-64	`executeSimpleGrant` has two similar checks	Low	Fixed
L-65	`vetoImplant` and `voteImplant` contain same logic	Low	Acknowledged
L-66	`updatePolicy()` requires `INT` to be greater than zero	Low	Fixed
L-67	Incorrect comment	Low	Fixed
L-68	Unused Modifier `onlyThis`	Low	Fixed
L-69	Incorrect Check for `_duration` in Constructor	Low	Fixed
L-70	Missing Input Validation for `quorum` and `threshold` in Constructor	Low	Acknowledged
L-71	Missing Zero Address Check in Proposal Creation	Low	Fixed
L-72	Typo in Error Name: `SnapShotExeuctor_InvalidParams`	Low	Fixed

1.6 Conclusion

The security audit of the project has been thoroughly conducted, focusing on the contracts installed into a Gnosis SAFE multisig. These contracts are designed to enable members to operate a BORG in accordance with the current BORG RFC Spec. The audit has verified that actions from the SAFE owners are appropriately limited and require approval from the adjacent DAO.

During the audit, key protocol concepts were checked: access control modules, Borg Safe guards and implants, and integrations with other protocols.

Some findings in the report highlight possible erroneous scenarios. Apart from the list of findings, we checked the following attack vectors:

1. **Correctness of access checks in `borgCore`**. It was verified that it is possible to set restrictions for the target addresses of native tokens transfers and the contract calls with specified calldata. Each contract method can be configured with a special policy that limits parameters' values, checking them against a configured range or exact values. It is impossible to pass calldata or transfer value that haven't been approved by the `borgCore` owners.

2. **Correctness of encoding functions signatures called from the implants**. Implants encode function calls such as ERC20, ERC721, and ERC1155 transfers and approve and call to implants themselves to allow them to put a veto on a proposal, execute a proposed grant, or send tokens to the `MetaVesT` controller. All the function calls are encoded correctly along with their parameters. There was an issue discovered where an incorrect function method selector was passed when creating a grant sent to the `MetaVesT` controller.

3. **Correctness of constraints removal**. All the added constraints for the contract method parameters and conditions inside `conditionManager` are removed correctly. The `parameterConstraints` mapping and `paramOffsets` array are cleared from the stored data on some particular methods. It is impossible to have old constraints applied to the newly added contracts in general. However, there was an issue related to the missing constraint removal in some particular cases.

4. **Correctness of integration with the `MetaVesT` controller**. The audit ensured that all the function calls from implants to the `MetaVesT` controller were configured with the correct parameters set. Additionally, it was confirmed that there are necessary setters for the `MetaVesT` controller address, allowing it to be changed if necessary.

5. **Owners ejection from the Borg Safe**. There is a special implant, `ejectImplant`, which allows the removal of owners from the Borg Safe and reduces the signing threshold. The audit confrimed that the `prevOwner` address during owner removal was calculated correctly. The `prevOwner` value is important because `owners` storage in the Borg Safe is organised as a linked list, and there is a requirement to pass the previous value of the element that is being deleted.

6. **Possible reentrancy after making transfers**. It was checked if the contracts follow the checks-effects-interactions pattern or have appropriate restrictions that prevent them from reentering into the important function. It was verified that, in most cases, all the state variables and checks are performed before making transfers. However, there is a finding related to a possible reentrancy attack in the `optimisticGrantImplant` function.

7. **Correct integration with oracle providers**. There are two oracle providers integrated: API3Oracle and Chainlink feed. The audit established that all the integrations follow best-practices approach and account for possibly outdated returned data. There were a few issues discovered which are related to API3Oracle and Chainlink integration.

8. **Bypassing parts of conditions for implant execution**. It was checked that configured conditions are applied correctly, and it is impossible to intentionally skip any of those checks while calling particular contract methods. However, an issue was discovered related to the logical `OR` operation, which is applied when conditions are combined.

9. **DoSing conditions with any external actions**. During the audit, it was verified that most of the conditions are constructed in a way prevents them from getting stuck unexpectedly due to external actions. However, an issue related to the balance check condition was discovered.

10. **Correctness of calldata parsing in the `borgCore`**. The audit ensured that the configured parameter offsets and parameter length in bytes are applied correctly while parsing passed calldata inside the `isMethodCallAllowed` function. However, some recommendations were made for organizing a more secure handling of bytes offsets configuration.

11. **Errors in the separation of rights between Borg and DAO**. It was checked that grants can be proposed only by the Borg Safe or Borg Safe owners and can be executed by the special governance executor. It is also crucial to note that along with proposal data, there is a correctly crafted veto calldata (in the `daoVetoGrantImplant` contract), which allows DAO to reject unwanted proposed grants.

After the concluded audit, it is important to note that the system's security depends on the correct protocol configuration, including initial parameters, roles granted, contract methods, and parameters restrictions. Proper deployment and initialization of all contracts are vital for the system to function properly.

2. FINDINGS REPORT

2.1 Critical

C-1	The governance adapter is not protected
Severity	Critical
Status	Fixed in 3357105a

Description

In the current implementation, `flexGovernanceAdapater` allows any user to call any function in the adapter, which can be exploited by a malicious user to create a proposal to transfer all tokens to an incorrect address. Considering that only implants should be able to call the adapter to pass specific calls to governance, this issue is severe and must be fixed before deployment:

- flexGovernanceAdapater.sol#L34
- flexGovernanceAdapater.sol#L45
- flexGovernanceAdapater.sol#L56
- flexGovernanceAdapater.sol#L64.

Recommendation

We recommend adding a modifier that will restrict method calls to only whitelisted addresses.

Client's commentary

> Client: We implemented the suggested fix by adding the onlyAdmin modifier on createProposal, executeProposal, and cancelProposal out of an abundance of caution. It is important to note that the governance adapter is just a pass-through contract to support multiple on-chain governance contracts. The validation is also expected to be handled on the governance contract that is approved to be used. Open Zepplin's Governor contract has a public function 'propose(..)', with validation internal to the method.
> MixBytes: The `vote` function is still can be called by any user.
> Client: We have removed the pass-thru vote function as voting should be done directly on the public facing governance contract.

C-2	Possible overflow
Severity	Critical
Status	Fixed in b89ff5b7

Description

`signerCount` is a `uint8` variable that can have a maximum value of 255. In there are 256 signers or more, it will overflow, causing the condition to function incorrectly signatureCondition.sol#L52. Borg owners will be able to pass the condition when only 1 or 2 signers have approved it.

Recommendation

We recommend removing the `signerCount` variable and using `_signers.length` to initialize `numSigners`.

Client's commentary

> We implemented the suggested fix, using `_signers.length` to init `numSigners`

C-3	Incorrect proposal can be deleted inside `_deleteProposal`
Severity	Critical
Status	Fixed in b89ff5b7

Description

There is an issue at the line: daoVetoGrantImplant.sol#L199. There can be a case when the `proposalIndex != lastProposalIndex` check doesn't pass, and we remove the last proposal from the `currentProposals` at line daoVetoGrantImplant.sol#203 instead of removing an actual one.

Let's assume there are two proposals in the `currentProposals` array with an ids `1` and `2`. If we try to delete the proposal with `id = 1` (the first element in the array), then `proposalIndex` would be equal to `1` and `lastProposalIndex = currentProposals.length - 1 = 1`. The `proposalIndex != lastProposalIndex` check won't pass and the proposal which is being removed won't be replaced by the one at the end of the proposals array. After the `if` block, the last element is removed from the `currentProposals` array. Here the proposal with `id = 2` was deleted.

This issue can lead to deleting (after vetoing) the incorrect proposal, so that an unwanted one can still be executed.

Recommendation

We recommend changing the check to `proposalIndex - 1 != lastProposalIndex` so that it will account for the difference between proposal index and array index.

Client's commentary

> We implemented the suggested fix changing deletion if statement to `proposalIndex - 1 != lastProposalIndex`.

C-4	Modifier `conditionCheck` doesn't revert if no conditions passed
Severity	Critical
Status	Fixed in b89ff5b7

Description

There is an issue at line conditionManager.sol#L123. Execution breaks when at least one condition succeeds, but in the case when all conditions are configured with a `Logic.OR` type and none return `true`, the modifier execution will still succeed.

Recommendation

We recommend adding a special flag to help track that no conditions with a `Logic.OR` type have passed.

Client's commentary

> We implemented the suggested fix of adding a flag to track the case of no `Logic.OR` conditions passing and returning false in that case.

2.2 High

H-1	Calls via `.call()` to internal functions `deleteProposal`, `executeDirectGrant`, `executeSimpleGrant`, and `executeAdvancedGrant` will always fail
Severity	High
Status	Fixed in b89ff5b7

Description

There is an issue at the lines: daoVetoGrantImplant.sol#L195, daoVetoGrantImplant.sol#L375, daoVetoGrantImplant.sol#L391, daoVetoGrantImplant.sol#L433. These functions are declared internal, but during the creation of proposals, calldata is created which encodes calls to the mentioned functions. During proposal execution, `.call()` is performed to invoke functions execution. Such calls will fail.

Recommendation

We recommend changing the visibility of the mentioned functions to external and restricting their callability to only the same implant (`daoVetoGrantImplant`). It can be done via the `onlyThis` modifier.

Client's commentary

> We implemented the suggested fix adding an onlyThis modifier and changing the function types.

H-2	Proposal duration is not set
Severity	High
Status	Fixed in b89ff5b7

Description

There is an issue at the lines: daoVetoGrantImplant.sol#L236, daoVetoGrantImplant.sol#L288, and daoVetoGrantImplant.sol#L346. New proposals are created and initialized with the id, start time, and calldata. But the proposal duration is not set, which will lead to the passing check at line daoVetoGrantImplant.sol#L174. All created proposals can be executed instantly without waiting for duration time.

Recommendation

We recommend setting the proposal duration at the time when it is created.

Client's commentary

> We implemented the suggested fix, now correctly passing the proposal duration into the governance adapter call.

H-3	DoS of `cooldown`
Severity	High
Status	Fixed in b89ff5b7

Description

The `checkTransaction` function serves as a guard for the BORG_SAFE multisig and checks the call's parameters for it. Currently, this function is publicly accessible, making it possible for any actor to call it. The function resets the last execution timestamp, which affects the cooldown check. Therefore, any actor can front-run any BORG SAFE transaction, causing it to fail and blocking the BORG SAFE:

- borgCore.sol#L153
- borgCore.sol#L166.

Recommendation

We recommend restricting access to this function only from the BORG SAFE contract.

Client's commentary

> We implemented the suggested fix, adding a modifier to restrict this call to only the BORG SAFE contract.

H-4	Reentrancy in `optimisticGrantImplant`
Severity	High
Status	Fixed in b89ff5b7

Description

The `createDirectGrant` function makes external calls (ETH or ERC20 transfer) before updating the `approvedToken.amountSpent`, which allows malicious owners of BORG_SAFE to transfer a much larger part of the funds and break the logic of the optimistic grant restrictions.
optimisticGrantImplant.sol#L128

Recommendation

We recommend using a check-effect-interaction pattern for this function or using a `nonReentrant` modifier.

Client's commentary

> We implemented the suggested fix of adding a nonReentrant modifier for createDirectGrant in the optimisticGrantImplant.

H-5	Incorrect function signature in `daoVoteGrantImplant.proposeAdvancedGrant()` `daoVetoGrantImplant.proposeAdvancedGrant()`
Severity	High
Status	Fixed in b89ff5b7

Description

An incorrect function signature `executeAdvancedGrant(MetaVesT.MetaVesTDetails)` is used for `abi.encodeWithSignature()` in `daoVoteGrantImplant.proposeAdvancedGrant()` and `daoVetoGrantImplant.proposeAdvancedGrant()`. The selector in the `proposalBytecode` will be calculated incorrectly, and the future call to `executeAdvancedGrant()` will be impossible.
daoVoteGrantImplant.sol#L234,
daoVetoGrantImplant.sol#L344

Recommendation

We recommend fixing the signature to

```
executeAdvancedGrant(
    (address,bool,uint8,
    (uint256,uint256,uint256,
    uint256,uint256,uint256,
    uint128,uint128,uint160,
    uint48,uint48,uint160,
    uint48,uint48,address),
    (uint256,uint208,uint48),
    (uint256,uint208,uint48),
    (bool,bool,bool),
    (uint256,bool,address[])[]))
```

Client's commentary

> We implemented the suggested fix of correcting the incorrect message signatures. Test coverage updated.

H-6	Incorrect loop boundary inside `removeContract`
Severity	High
Status	Fixed in 3357105a

Description

There is an issue at borgCore.sol#L269. `policy[_contract].methodSignatures.length` is used as a loop boundary, but the length of the `policy[_contract].methodSignatures` is reduced inside the `_removePolicyMethodSelector` function. It leads to the missing loop iterations, which means that not all `methodSelector` is removed.

Recommendation

We recommend not removing `methodSignatures` inside the `_removePolicyMethodSelector` function.

Client's commentary

> We no longer remove 'methodSignatures' inside the `_removePolicyMethodSelector` function and instead clear the array after all mappings have been cleared.

H-7	Always failing check inside the `checkCondition` for `chainlinkOracleCondition`
Severity	High
Status	Fixed in 3357105a

Description

There is an issue at chainlinkOracleCondition.sol#L59. `updatedAt` value fetched from the Chainlink `latestRoundData()` is the time in the past when the price was reported. `block.timestamp` would always be bigger than that value, which will lead to condition revert.

Recommendation

We recommend changing the mentioned check to `block.timestamp - updatedAt > acceptableDuration`.

Client's commentary

> We have implemented the fix to correct the check to `block.timestamp - updatedAt > acceptableDuration`.

2.3 Medium

M-1	Updating the policy for the existing method will lead to adding unnecessary `byteOffset` to the `paramOffsets` array
Severity	Medium
Status	Fixed in b89ff5b7

Description

There is an issue at lines borgCore.sol#L262 and borgCore.sol#L266. `paramOffsets` is updated even if there is an attempt to change an existing parameter.

Recommendation

We recommend not changing the `paramOffsets` array if a previously existing method parameter constraint is being changed.

Client's commentary

> Implemented the suggested fix to check a paramOffsets uniqueness before adding it to the array.

M-2	Missing transferred value check
Severity	Medium
Status	Fixed in b89ff5b7

Description

There is a check at line borgCore.sol#L154 which passes if there is any calldata specified for the call. It will also pass if there is some value transferred together with the call, which can lead to bypassing the previous check (borgCore.sol#L141) for native token transfers.

Recommendation

We recommend adding a check for the transferred value when the contract method is triggered.

Client's commentary

> We have updated `checkTransaction` to always check the native transfer settings when a tx value is >0 and to always check the policy when data.length is > 0.

M-3	Method policies are not cleaned on contract removal
Severity	Medium
Status	Fixed in b89ff5b7

Description

There is a `removeContract` function defined at line borgCore.sol#L203. It sets the `allowed` and `fullAccess` struct fields to `false`, disallowing the contract from any calls. But it doesn't clean the `policy[_contract].methods` policies mapping. This means it is possible to have that contract allowed in the future with old policies.

Recommendation

We recommend cleaning contract method policies when they are removed from the whitelist.

Client's commentary

> We implemented the suggested fix cleaning the mappings for parameters and methods when the parent policy is removed. We also now store the method mapping keys in an array so we can ensure they are properly cleared when removed.

M-4	Missing `conditionCheck` modifiers
Severity	Medium
Status	Fixed in b89ff5b7

Description

There are two functions: `ejectOwner`, defined at line ejectImplant.sol#L50, and `recoverSafeFunds`, defined at line failSafeImplant.sol#L86. Both of them have a call to the `checkConditions` function inside, but there are no `conditionCheck` modifiers which check conditions particularly for the callable function.

Recommendation

We recommend adding the `conditionCheck` modifier to the mentioned functions.

Client's commentary

> We implemented the suggested fix of adding the conditionCheck modifier to the mentioned functions.

M-5	A threshold check can prevent the owner from self-ejecting
Severity	Medium
Status	Fixed in b89ff5b7

Description

There is a `owners.length > threshold` check at line ejectImplant.sol#L185. There could be some cases when `owners.length == threshold`, and we are attempting to remove one owner and reduce the threshold by `1`. In that case, the mentioned check would always fail and prevent the user from self-ejecting, as the next call to `removeOwner` would fail because there would be an attempt to set a larger threshold than the updated number of owners.

Recommendation

We recommend removing the unnecessary check.

Client's commentary

> We implemented the recommended fix by removing the unnecessary check.

M-6	Chainlink returned data is not checked for being stale
Severity	Medium
Status	Fixed in b89ff5b7

Description

There is a call to the Chainlink price feed `latestRoundData` at line chainlinkOracleCondition.sol#L53. It can return a stale price due to Chainlink lagging in delivering actual data.

Recommendation

We recommend adding a call to the price feed function `updatedAt` to check if the returned data is not stale.

Client's commentary

> We implemented the necessary check, adding a configurable threshold parameter, and checking the time the datafeed was last updated.

M-7	Conditions' order has impact
Severity	Medium
Status	Acknowledged

Description

The current implementation of the conditions check relies on the condition order, which makes the system more vulnerable to owner's faults and makes the protocol less flexible.

- conditionManager.sol#L62-L74
- conditionManager.sol#L120-L126

Recommendation

We recommend redesigning the conditions check architecture to make its order independent.

Client's commentary

> We did not change the condition check order, as it gives the owner more control over condition execution based on order. We will add comments to reflect this as well as update our docs to explain.

M-8	API3 heartbeat
Severity	Medium
Status	Fixed in b89ff5b7

Description

API3 heartbeat is 24 hours, which means that the condition is vulnerable to lags in updates: API3OracleCondition.sol#L46.

Recommendation

We recommend considering that the value update in the API3 oracle can take slightly more than 24 hours.

Client's commentary

> We implemented a configurable threshold parameter, allowing the owner to set the heartbeat threshold to account for update lag.

M-9	The owner can remove themselves
Severity	Medium
Status	Fixed in b89ff5b7

Description

The current implementation of role updates allows the owner to remove themselves, which can lead to the full block of the system: auth.sol#L42-L48.

Recommendation

We recommend restricting the owner's ability to remove themselves and creating a separate function for such actions.

Client's commentary

> We implemented the recommended fix by adding a two-method process to initTransferOwnership, acceptTransfer, and the separate function to zeroOwner if desired, which may be a case for certain BORGs.

M-10	Tests do not work
Severity	Medium
Status	Fixed in 3357105a

Description

Tests do not work, making it impossible to check test coverage and prepare PoCs.

Recommendation

We recommend preparing and setting up all the tests before the protocol deployment.

Client's commentary

> Client: We have added test coverage and ensured the tests all pass. The tests *must* be run as a forked test on the Ethereum Sepolia Test Net. The Private Key must be set to the 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266 address, which is a hard hat test address and the private key can be found readily available online. Please reach out to ensure you can run the tests.
> MixBytes: Currently, `daoVetoGrantImplant` has a `BaseAllocation` type which is not within the scope

M-11	Seconds are used instead of days
Severity	Medium
Status	Fixed in b89ff5b7

Description

`duration` is currently not used, but if it were used, it would incorrectly use seconds instead of days: daoVetoGrantImplant.sol#L28.

Recommendation

We recommend using the correct units for `duration`.

Client's commentary

> We implemented the suggested unit correction fix.

M-12	`duration` is not limited
Severity	Medium
Status	Fixed in b89ff5b7

Description

Currently, `duration` is not saved in proposals (HIGH #2) but is should be saved to proposals, and the value should be limited to avoid breaking the logic: daoVetoGrantImplant.sol#L131.

Recommendation

We recommend limiting `duration` because it will be saved for proposals without the ability to update it.

Client's commentary

> We fixed the error of not passnig duration to the proposal in another fix. We also added a MAX_PROPOSAL_DURATION constant of 30 days to limit duration sets, as well as a grace period to allow the governance the chance to delete the proposal before the BORG members can execute it.

M-13	Proposals do not have an expiration date
Severity	Medium
Status	Fixed in b89ff5b7

Description

In the current architecture, proposals do not have an expiration date so they can be executed after a long period of time:

- daoVetoGrantImplant.sol#L375
- daoVetoGrantImplant.sol#L391
- daoVetoGrantImplant.sol#L433.

Recommendation

We recommend adding an expiration time for the proposals.

Client's commentary

> We have implemented the recommended fix by adding a configurable expiration time for proposals.

M-14	Too flexible logic
Severity	Medium
Status	Fixed in b89ff5b7

Description

`updatePolicy` can be used to update existing policies or add new ones. The chosen architecture is too flexible, making it harder to control the correctness of parameter updates for the owner. The same issue also applies to the `updateMethodCooldown` function:

- borgCore.sol#L257-L267
- borgCore.sol#L383-L395.

Recommendation

We recommend updating the current architecture so that there are two different methods: one to add a new policy/cooldown, which checks that the policy/cooldown didn't previously exist, and the second, which allows updating policy/cooldown with checks that such an element was previously added.

Client's commentary

> We have implemented the suggested fix to only allow the cooldown to be updated if the policy for that method already exists. We have also made several updates to always check if a policy exists for an update, to prevent a policy being updated incorrectly.

M-15	Weak condition
Severity	Medium
Status	Fixed in b89ff5b7

Description

The token's balance can be easily manipulated by any actor, which makes `balanceCondition` vulnerable to DoS attacks:
balanceCondition.sol#L37.

Recommendation

We don't recommend using `Comparison.EQUAL` for token balance checks as it can be easily manipulated.

Client's commentary

> We have implemented the suggested fix by removing the 'Comparison.EQUAL' option in 'balanceCondition'

M-16	Backrun of partly set constraints
Severity	Medium
Status	Fixed in b89ff5b7

Description

The current version of `borgCore.updatePolicy(_contracts, _methodNames...)` doesn't allow setting up all constraints in one call because it doesn't support constraints for `UINT` and some constraints with multiple exact matches for one parameter. So, multiple calls will be required for some methods. However, the method becomes callable after the first method's constraint is set — the first call of `borgCore.updatePolicy(_contracts, _methodNames...)` may be backrun by the Safe, and some partly allowed action will be performed. borgCore.sol#L227
If `borgCore.addContract()` is intended to add some contract to `policy[]` before setting constraints, it may be backrun too. borgCore.sol#L196

Recommendation

We recommend adding `UINT` support to `updatePolicy(_contracts, _methodNames...)` and fixing other limitations of `borgCore.updatePolicy(_contracts, _methodNames...)` from the LOW section. We also recommend renaming `addContract()` function to `addFullAccessContract()` or `addUnrestrictedContract()`.

Client's commentary

> We have updated `updatePolicy(_contracts, _methodNames...)` to support any type of parameter to allow all of the parameters for a method to be set in a single transaction. The risk of multiple BORG members coordinating to backrun with a multisig is low and additional protections exist in the members legal relationship with the BORG entity.

M-17	Missing boundary check for `duration`
Severity	Medium
Status	Fixed in 3357105a

Description

There is an issue at daoVoteGrantImplant.sol#L139. `_duration` value is not checked which may lead to a significant proposal duration being set. It can lead to proposal creation, which will not be limited by the duration time.

Recommendation

We recommend adding `MAX_PROPOSAL_DURATION` variable which will be used to limit the `_duration` being set.

Client's commentary

> Added a MAX_PROPOSAL_DURATION for duration in the constructor and the duration setter.

M-18	The `borgMode` variable value should be immutable
Severity	Medium
Status	Fixed in 3357105a

Description

There is a `borgMode` variable defined at borgCore.sol#L88. It should be impossible to change its value as it will lead to `methodConstraint` checks performed differently inside `isMethodCallAllowed` function.

Recommendation

We recommend fixating the `borgMode` variable value inside the contract constructor.

Client's commentary

> borgMode is now set in the constructor with no ability to change the mode after deployment.

M-19	Parameter constraint is rewritten with empty values in some cases
Severity	Medium
Status	Fixed in 3357105a

Description

There is an issue at borgCore.sol#L424. If there are such `minValue` and `maxValue` which lead to the check failing, the chosen parameter constraint will be rewritten by the empty values at borgCore.sol#L434. The same issue is present at borgCore.sol#L428.

Recommendation

We recommend changing the logic of the check so that the `updatePolicy` function execution reverts if there are incorrect parameters specified. `_addParameterConstraint` with empty `_minValue`, `_maxValue`, `_iminValue` or `_imaxValue` should be called only if there was different than `UINT` or `INT` `ParamType` provided.

Client's commentary

> We have added param validation checks within the call to check for UINT and INT param types so that they won't fall to 0 out. We also revert on incorrect UINT and INT values.

M-20	The native token transfer is not supported by `borgCore.checkTransaction()` in `blacklist` borg mode
Severity	Medium
Status	Fixed in 3357105a

Description

There is a zero check of calldata size in `borgCore.checkTransaction()` in the `blacklist` branch. So, it's impossible to transfer native tokens in a regular way in this mode. There is also no way to restrict call value similar to what's done in the `whitelist` branch.

borgCore.sol#L168
borgCore.sol#L600

Recommendation

We recommend adding in `borgCore.checkTransaction()` support of native tokens to the `blacklist` branch, similar to the `whitelist` branch.

Client's commentary

> We have implemented the suggested change of implementing the check for value > 0 and data.length > 0 for the 'blacklist' branch similarly to the 'whitelist' branch. Native eth destinations are blocked from eth transfer calls if they are added as a receipient in 'blacklist' mode. We also changed the mapping variable name from 'whitelistRecipients' to 'policyRecipient'.

M-21	Unreachable Code in `updateThreshold` Function
Severity	Medium
Status	Fixed in 32a32c40

Description

The issue is identified within the multiUseSignCondition.sol#L78-L85 function of the `MultiUseSignCondition` contract. The function contains a check to ensure that only an owner of the BORG_SAFE can update the threshold. However, this check will always fail because the `onlyOwner` modifier requires `msg.sender` to be the `BORG_SAFE` contract itself, which conflicts with the check inside the function that requires `msg.sender` to be an owner of BORG_SAFE. Since the BORG_SAFE contract cannot be an owner of itself, any call to this function will always revert.

Recommendation

We recommend removing the check `if (!ISafe(BORG_SAFE).isOwner(msg.sender))` within the `updateThreshold` function. Since the `onlyOwner` modifier already restricts access to the BORG_SAFE contract, additional checks inside the function are redundant and cause the function to always revert.

Client's Commentary

> We have corrected this with the recommended fix by removing the `if (!ISafe(BORG_SAFE).isOwner(msg.sender))` check in this function.

M-22	`borgCore._checkDirectorsSignatures()` calculates the number of directors' signatures incorrectly if the total number of signatures > threshold
Severity	Medium
Status	Fixed in 32a32c40

Description

`Safe.execTransaction()` Safe.sol#L111-L192 to pass more than threshold signatures. `SignatureHelper.getSigners()` signatureHelper.sol#L50 only first threshold signers. So if some of the directors' signatures go after the first threshold signatures, borgCore.sol#L756 in `borgCore._checkDirectorsSignatures()` and the total number will be calculated incorrectly.

Recommendation

We recommend adding this information to the documentation.

Client's commentary

> We have addressed this with the recommended fix. We have added comments in borgCore.sol and signatureHelper.sol to reflect this and will plan to add it to our documenation around the Director signer type.

M-23	Missing Update of `lastProposalTime` in `proposeTransaction`
Severity	Medium
Status	Fixed in 2cc22ab1

Description

The issue has been identified within the `proposeTransaction` function of the `daoVetoImplant` contract. The function fails to update the `lastProposalTime` variable to the current `block.timestamp` after a proposal is created. This could cause the proposal cooldown logic to malfunction, as it relies on `lastProposalTime` to enforce proper time intervals between proposals.
The issue is classified as **Medium** severity because it could allow users to bypass the intended proposal cooldown period, potentially leading to system abuse or congestion.

Recommendation

We recommend updating the `lastProposalTime` variable to the current `block.timestamp` immediately after a proposal is successfully created.

Client's Commentary

> We have addresed this with the recommended fix of setting the lastProposalTime in proposeTransaction.

M-24	Expired Proposal Can Still Be Executed
Severity	Medium
Status	Fixed in b9d43386

Description

The `execute` function of the `SnapShotExecutor` contract does not verify whether the proposal has expired based on the `proposalExpirySeconds` parameter. As a result, proposals may be executed long after their intended lifespan, even if they are eligible for cancellation due to expiry. This undermines the expiration logic defined in the `cancel` function and weakens the governance control over outdated or potentially malicious proposals.

Recommendation

We recommend adding an expiration check to the `execute` function using the `proposal.executableAfter` timestamp. A proposal should only be executable if the current timestamp is greater than `executableAfter` and less than `executableAfter + proposalExpirySeconds`.

```
if (p.executableAfter + proposalExpirySeconds <= block.timestamp) {
    revert ProposalExpired();
}
```

Client's Commentary

> We plan to stick with the more flexible approach to proposal deadlines, but recognize that `expiry` is misleading and will rename it to `cancelWaitingPeriod` instead

2.4 Low

L-1	Missing parameter checks
Severity	Low
Status	Fixed in b89ff5b7

Description

In `checkTransaction()`: `operation` should be checked depending on whether `DELEGATECALL` is supported or not.
borgCore.sol#L129

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> We have added bool delegateCallAllowed to the policy struct and a delegateCallAllowed toggle function to disallow/allow delegatecalls based on contract.

L-2	`matchNum.length` should be checked
Severity	Low
Status	Fixed in b89ff5b7

Description

In `updatePolicy(_contracts, _methodNames...)`: `matchNum.length` should be equal to `_contracts.length`.
borgCore.sol#L229

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> We have implemented the suggested fix

L-3	`updatePolicy` parameters should be checked
Severity	Low
Status	Fixed in b89ff5b7

Description

In `updatePolicy(_contracts, _methodNames...)`: should check other parameters based on `_paramTypes`. `_exactMatches` should be empty for `INT` and `UINT` (in case `UINT` support will be added to `updatePolicy(_contracts, _methodNames...)`). `_exactMatches` should be keccak256(true) or keccak256(false) for the `BOOL` type. It's also better to store this bool values as constants. Min and max values should be 0 for types with nonempty `_exactMatches`.
borgCore.sol#L227

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> We have updated with suggested fixes, except storing the bool type/constants suggestions.

L-4	`_recipient` zero address check
Severity	Low
Status	Fixed in b89ff5b7

Description

In `addRecipient()`: a missing `_recipient` zero check.
borgCore.sol#L184

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix.

L-5	`_methodCallData` length check
Severity	Low
Status	Fixed in b89ff5b7

Description

In `isMethodCallAllowed()`: `_methodCallData's` length should be checked.
borgCore.sol#L429

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix.

L-6	`_condition` uniqueness check
Severity	Low
Status	Fixed in 3357105a

Description

In `addCondition()`: `_condition` address should be checked for absence in `conditions`.
conditionManager.sol#L38

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> Client: we have implemented the suggested fix by adding ERC165 interface checks on conditions.
> MixBytes: Condition uniqueness not checked, so owner can add several same conditions.
> Client: We have added a check for ensuring unique conditions.

L-7	`_condition` absence check
Severity	Low
Status	Fixed in 3357105a

Description

In `addConditionToFunction()`: `_condition` address should be checked for absence in `conditionsByFunction[_functionSignature]`. conditionManager.sol#L93

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> Client: we have implemented the suggested fix by adding ERC165 interface checks on conditions.
> MixBytes: `_condition` address not checked, so owner can add several same conditions.
> Client: We have added a check for ensuring unique conditions.

L-8	`RECOVERY_ADDRESS` zero address check
Severity	Low
Status	Fixed in b89ff5b7

Description

Constructor: missing `RECOVERY_ADDRESS` zero check. failSafeImplant.sol#L52

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix.

L-9	`addToken()` parameters are not checked
Severity	Low
Status	Fixed in 3357105a

Description

In `addToken()`: a `_tokenAddress` uniqueness check in `tokenList`. `tokenType` should be 0, 1 or 2. It may be also convinient to use an enum for `tokenType`. `_amount` should be 1 if `tokenType` for ERC721. `_id` should be 0 for ERC20. failSafeImplant.sol#L60

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> Client: we have implemented the suggested fix.
> MixBytes: The parameters are still not checked
> Client: We have implemented the parameter checks in the description.

L-10	`quorum` and `threshold` not checked
Severity	Low
Status	Acknowledged

Description

Constructor, `updateQuorum()` and `updateThreshold()`: `quorum` and `threshold` should be <= 100%. A value corresponding to 100% should be added as a constant: daoVoteGrantImplant.sol#L77, daoVoteGrantImplant.sol#L96, daoVoteGrantImplant.sol#L103.

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> Due to many on chain govenrance systems, the values here may be absolute values OR percentage values, so we did not add a total percentage validation.

L-11	`_duration` should be limited
Severity	Low
Status	Fixed in b89ff5b7

Description

Constructor, `updateDuration()`: `_duration` should be limited by some constant: daoVoteGrantImplant.sol#L77, daoVoteGrantImplant.sol#L89.

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix with a max duration constant.

L-12	`_governanceAdapter` and `_governanceExecutor` not checked
Severity	Low
Status	Acknowledged

Description

Constructor, `setGovernanceAdapter()` and `setGovernanceExecutor()`: missing `_governanceAdapter` and `_governanceExecutor` zero checks: daoVoteGrantImplant.sol#L77, daoVoteGrantImplant.sol#L110, daoVoteGrantImplant.sol#L117.

Recommendation

We recommend adding parameters checks from the description.

L-13	Proposal creator validation missing
Severity	Low
Status	Fixed in b89ff5b7

Description

In `executeDirectGrant()`: add a check that the proposal was created by the implant.
daoVoteGrantImplant.sol#L255-L271

Recommendation

We recommend adding parameters checks from the description.

L-14	`quorum` and `threshold` should be <= 100%
Severity	Low
Status	Acknowledged

Description

Constructor, `updateQuorum()` and `updateThreshold()`: `quorum` and `threshold` should be <= 100%. A value corresponding to 100% should be added as a constant: daoVetoGrantImplant.sol#L94, daoVetoGrantImplant.sol#L137, daoVetoGrantImplant.sol#L144.

Recommendation

We recommend adding parameters checks from the description.

L-15	`_duration` and `_waitingPeriod` should be limited
Severity	Low
Status	Acknowledged

Description

Constructor, `updateWaitingPeriod()`, `updateDuration()`: `_duration` and `_waitingPeriod` should be limited by some constant: daoVetoGrantImplant.sol#L94, daoVetoGrantImplant.sol#L123, daoVetoGrantImplant.sol#L130.

Recommendation

We recommend adding parameters checks from the description.

L-16	`_governanceAdapter` is not checked
Severity	Low
Status	Acknowledged

Description

Constructor, `setGovernanceAdapter()`: missing a `_governanceAdapter` zero check.
daoVetoGrantImplant.sol#L151

Recommendation

We recommend adding parameters checks from the description.

L-17	Grant amount spending limit check
Severity	Low
Status	Acknowledged

Description

In `executeDirectGrant()`, `executeSimpleGrant()`, `executeAdvancedGrant()`: `_amount` should be less than token's `spendingLimit`. This check should be added in case the limit was changed between grant creation and execution: daoVetoGrantImplant.sol#L377, daoVetoGrantImplant.sol#L393, daoVetoGrantImplant.sol#L433.

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> Spending limit should be unchanged in the proposal even if the implant has changed since it needs co-approval via veto vote.

L-18	Safe balance check
Severity	Low
Status	Fixed in b89ff5b7

Description

In `createDirectGrant()`: `_amount` should be less than Safe's balance.
optimisticGrantImplant.sol#L105

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix.

L-19	Proxy address validation
Severity	Low
Status	Fixed in b89ff5b7

Description

Constructor: `_proxyAddres` should be checked by calling `IProxy(_proxyAddress).read()`.
API3OracleCondition.sol#L36

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix.

L-20	Token balance check
Severity	Low
Status	Fixed in b89ff5b7

Description

Constructor: `_token` should be checked by calling `IERC20(_token).balanceOf(_target)`.
balanceCondition.sol#L24

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix.

L-21	`isSigner` addresses uniqueness check
Severity	Low
Status	Fixed in b89ff5b7

Description

Constructor: missing a `isSigner` addresses uniqueness check. Missing a `isSigner` zero check. signatureCondition.sol#L49.

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix for the address(0) check.

L-22	`_threshold` should be > 0
Severity	Low
Status	Fixed in b89ff5b7

Description

Constructor: `_threshold` should be > 0.
signatureCondition.sol#L42

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix.

L-23	`BORG_SAFE` is not checked
Severity	Low
Status	Fixed in b89ff5b7

Description

Constructor: missing a `BORG_SAFE` zero check. deadManSwitchCondition.sol#L27

Recommendation

We recommend adding parameters checks from the description.

Client's commentary

> we have implemented the suggested fix.

L-24	Role validation check
Severity	Low
Status	Acknowledged

Description

`updateRole()` and `setRoleAdapter()`: `_role` is `PRIVILEGED_ROLE`, `ADMIN_ROLE` or `OWNER_ROLE`.
auth.sol#L42, auth.sol#L53

Recommendation

We recommend adding parameters checks from the description.

L-25	Code improvements
Severity	Low
Status	Fixed in b89ff5b7

Description

`exactMatch`: add a comment that this is an array of hashes.
borgCore.sol#L64

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-26	`byteLength` adjustment
Severity	Low
Status	Fixed in b89ff5b7

Description

A constant `byteLength` should be used for `UINT`, `ADDRESS`, `BOOL` and `INT`.
borgCore.sol#L506

`isMethodCallAllowed()` should return false instead of revert in case the exact match wasn't found.
borgCore.sol#L463

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-27	Exact match check
Severity	Low
Status	Fixed in b89ff5b7

Description

`isMethodCallAllowed()`: stop the loop in case the exact match was found.
borgCore.sol#L459

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-28	Remove unnecessary conversion
Severity	Low
Status	Fixed in b89ff5b7

Description

`isMethodCallAllowed()`: remove unneccessary conversion to int.
borgCore.sol#L451

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-29	Simplify logical conditions
Severity	Low
Status	Fixed in b89ff5b7

Description

`checkTransaction()`: simplify logical conditions borgCore.sol#L155 to `if (!policy[to].allowed)`, borgCore.sol#L158 to `if (policy[to].fullAccess)`.

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-30	Full access policies in `updatePolicy`
Severity	Low
Status	Fixed in b89ff5b7

Description

`updatePolicy(_contracts, _methodNames...)`: we recommend removing support of fullAccess-policies from this method. Otherwise, the result of the function call depends on the order of policies if there are method-policies and fullAccess-policies for the same method in the array. borgCore.sol#L227

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix via ensuring that if a contract was already enabled, we don't set it to fullAccess, to prevent unintentionally giving full access once methods have been added.

L-31	Input parameter requirement removal
Severity	Low
Status	Fixed in b89ff5b7

Description

`updatePolicy(_contracts, _methodNames...)`: remove the `_contracts.length > _exactMatches.length` requirement for input parameters as there can be multiple exact matches for one parameter.
borgCore.sol#L232

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-32	Legal agreement condition fix
Severity	Low
Status	Fixed in b89ff5b7

Description

`removeLegalAgreement()`: fix the if-condition to `_index >= legalAgreements.length`.
borgCore.sol#L291

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-33	`_byteOffset` missing check
Severity	Low
Status	Fixed in b89ff5b7

Description

`removeParameterConstraint()`: revert in case `_byteOffset` wasn't found.
borgCore.sol#L411-L417

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-34	Remove needless uint8 cast
Severity	Low
Status	Fixed in b89ff5b7

Description

`removeParameterConstraint()`, `_addParameterConstraint()`: remove the needless uint8 cast.
borgCore.sol#L418 borgCore.sol#L515

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-35	Protection against underflow in cooldown check
Severity	Low
Status	Fixed in b89ff5b7

Description

In `_checkCooldown()`:

```
block.timestamp <
methodConstraint.cooldownPeriod + methodConstraint.lastExecutionTimestamp
```

should be used instead of

```
block.timestamp - methodConstraint.lastExecutionTimestamp <
    methodConstraint.cooldownPeriod
```

for protection against underflow.
borgCore.sol#L527

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-36	Underflow protection in native cooldown check
Severity	Low
Status	Fixed in b89ff5b7

Description

In `_checkNativeCooldown()`: `block.timestamp < nativeCooldown + lastNativeExecutionTimestamp` should be used instead of `block.timestamp - lastNativeExecutionTimestamp < nativeCooldown` for protection against underflow.
borgCore.sol#L539

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-37	Native token recovery in failSafe implant
Severity	Low
Status	Fixed in b89ff5b7

Description

`failSafeImplant` doesn't have functionality for the native token recovery. But according to `borgCore.checkTransaction()`, the native token support is assumed.

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix by adding native token recovery.

L-38	FundsRecovered event corrections
Severity	Low
Status	Fixed in b89ff5b7

Description

Events: `FundsRecovered.id` should be 0 for ERC20 in `recoverSafeFunds()`, failSafeImplant.sol#L102. `FundsRecovered.amount` should be 1 for ERC721 in `recoverSafeFunds()`, failSafeImplant.sol#L107. `FundsRecovered.tokenType` should have a correct value in `recoverSafeFundsERC20()`, `recoverSafeFundsERC721()` and `recoverSafeFundsERC1155()`, failSafeImplant.sol#L133, failSafeImplant.sol#L146 and failSafeImplant.sol#L161.

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-39	Optimize `TokenInfo` struct
Severity	Low
Status	Fixed in b89ff5b7

Description

The `TokenInfo` struct can be optimized. `tokenAddress` and `tokenType` can be placed sequentially to fit into one storage slot:

```
struct TokenInfo {
    uint256 id;
    uint256 amount;
    address tokenAddress;
    uint8 tokenType;
}
```

So the struct would occupy only 3 slots instead of 4.
failSafeImplant.sol#L26

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-40	Unnecessary threshold check in the `ejectOwner`
Severity	Low
Status	Fixed in b89ff5b7

Description

`ejectOwner()`: an unnecessary check for `_threshold` at line ejectImplant.sol#L78. The `_threshold < owners.length` check would be always `true` if the call to the Borg Safe `removeOwner` didn't revert.

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-41	Native token support in DAO implants
Severity	Low
Status	Fixed in b89ff5b7

Description

`daoVetoGrantImplant` and `daoVoteGrantImplant` currently don't work with the native token daoVoteGrantImplant.sol#L145 and daoVetoGrantImplant.sol#L148 will revert in the case of the native token, so the corresponding branch should be removed from `executeDirectGrant()`: daoVoteGrantImplant.sol#L263, daoVetoGrantImplant.sol#L282. For uniformity the native token support should also be removed from `optimisticGrantImplant`, so the grantee will not be dependent on the type of the implant.

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix by supporting native tokens in direct grants of each type.

L-42	Balance check in `executeAdvancedGrant`
Severity	Low
Status	Fixed in b89ff5b7

Description

In `executeAdvancedGrant()`: there should be a check that the `BORG_SAFE's` token balance is not less than `_total`, as it's done in other execution methods.
daoVetoGrantImplant.sol#L433, daoVoteGrantImplant.sol#L324

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-43	Overwritten initial values
Severity	Low
Status	Fixed in b89ff5b7

Description

Initial values for `duration`, `quorum`, `threshold` and `waitingPeriod` (for `daoVetoGrantImplant`) are specified in code, but they are overwritten in the constructor. daoVetoGrantImplant.sol#L28-L31, daoVoteGrantImplant.sol#L28-L30

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-44	Internal method for grant proposals
Severity	Low
Status	Acknowledged

Description

`proposeDirectGrant()`, `proposeSimpleGrant()`, `proposeAdvancedGrant()` can use the same internal method to decrease the size of the contract. daoVetoGrantImplant.sol#L214-L369, daoVoteGrantImplant.sol#L143-L249

Recommendation

We recommend making improvements from the description.

L-45	Separate methods for limit management
Severity	Low
Status	Acknowledged

Description

In `addApprovedGrantToken()`: it's better to add two methods: one to add a limit and another one to update the limit to restrict actions and make the protocol more atomic.
daoVetoGrantImplant.sol#L109-L112, optimisticGrantImplant.sol#L68-L71

Recommendation

We recommend making improvements from the description.

L-46	Remove unused errors
Severity	Low
Status	Fixed in b89ff5b7

Description

Remove unused errors.
daoVetoGrantImplant.sol#L52, daoVetoGrantImplant.sol#L56, daoVetoGrantImplant.sol#L57, daoVetoGrantImplant.sol#L62, daoVetoGrantImplant.sol#L63

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-47	Unnecessary initialization of `newProposalId`
Severity	Low
Status	Fixed in b89ff5b7

Description

Unnecessary initialization of `newProposalId` to zero in `proposeSimpleGrant()`, `proposeDirectGrant()` `proposeAdvancedGrant()`. daoVetoGrantImplant.sol#L217, daoVetoGrantImplant.sol#L269, daoVetoGrantImplant.sol#L320. There is a line daoVetoGrantImplant.sol#L289 where its value is always being overwritten.

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-48	Setter for `metaVesTController`
Severity	Low
Status	Fixed in b89ff5b7

Description

A setter for `metaVesTController` should be added.
daoVetoGrantImplant.sol#L25

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-49	Rename struct `prop` to `proposalDetail`
Severity	Low
Status	Fixed in b89ff5b7

Description

Struct `prop` should be renamed to `proposalDetail`.
daoVetoGrantImplant.sol#L44

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-50	Remove unused `governanceExecutor`
Severity	Low
Status	Fixed in b89ff5b7

Description

`governanceExecutor` is unused and should be removed.
daoVetoGrantImplant.sol#L21

Recommendation

We recommend making improvements from the description.

Client's commentary

> governanceExecutor is now used for methods the exeuctor needs to call.

L-51	`BORG_SAFE` rights extension
Severity	Low
Status	Fixed in b89ff5b7

Description

The `BORG_SAFE` address as a caller should be also allowed in `createDirectGrant()`, `createBasicGrant()` and `createAdvancedGrant()` if `requireBorgVote` is false. optimisticGrantImplant.sol#L115, optimisticGrantImplant.sol#L158 and optimisticGrantImplant.sol#L223

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-52	Optimization via internal methods
Severity	Low
Status	Acknowledged

Description

`createDirectGrant()`, `createBasicGrant()`, `createAdvancedGrant()` can use the same internal method to decrease the size of the contract. optimisticGrantImplant.sol#L101-L255

Recommendation

We recommend making improvements from the description.

L-53	Restriction to `view`
Severity	Low
Status	Fixed in b89ff5b7

Description

If `BaseCondition.checkCondition()` is intended to be used for read-only checks, it should be restricted to `view`. In such case `ConditionManager.checkConditions()` should be also view. Otherwise, callers of `ConditionManager.checkConditions()` should be restricted.
baseCondition.sol#L6, conditionManager.sol#L58

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-54	Overflow optimization
Severity	Low
Status	Fixed in b89ff5b7

Description

`checkCondition()`: `block.timestamp > ONE_DAY + _timestamp` should be used instead of `block.timestamp - _timestamp > ONE_DAY` for protection against underflow.
API3OracleCondition.sol#L46

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-55	Unnecessary initialization
Severity	Low
Status	Fixed in b89ff5b7

Description

`startTime`: unnecessary initialization to zero.
deadManSwitchCondition.sol#L11

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-56	`numSigners` optimization
Severity	Low
Status	Fixed in b89ff5b7

Description

Constructor: `numSigners` can be set to `_signers.length`.
signatureCondition.sol#L55

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix.

L-57	Existing policies removal
Severity	Low
Status	Fixed in b89ff5b7

Description

`removeContract()`, `updatePolicy(_contracts)`: remove existing policies if the method is granted by `fullAccess` or the contract removed from `policy[]`.
borgCore.sol#L203, borgCore.sol#L217

Recommendation

We recommend making improvements from the description.

Client's commentary

> we have implemented the suggested fix for when the contract is removed.

L-58	Incorrect implementation of ERC165
Severity	Low
Status	Fixed in 3357105a

Description

The ERC165 standard is implemented incorrectly by `BaseCondition`. It should return `true` for any supported interface. But it also returns `true` for the interface with id `0x8b94fce4`. It's not clear what the interface with such an ID is because the `ICondition` interface has a different interface ID.

https://eips.ethereum.org/EIPS/eip-165

Recommendation

We recommend adding `ICondition` to `BaseCondition's` implemented interfaces and checking it's id with `type(ICondition).interfaceId`.

Client's commentary

> We have corrected this using the recommended fix, we add ICondition to the BaseCondition class and check the value with `type(ICondition).interfaceId`.

L-59	Parameter constraints in `borgCore` work incorrectly in the case of `borgModes.blacklist`
Severity	Low
Status	Acknowledged

Description

According to the business logic, parameter constraints are not supported by `borgCore` in the case of `borgModes.blacklist`. But there are no restrictions on adding them. If they are added by mistake, `isMethodCallAllowed()` (and therefore `checkTransaction()`) works incorrectly, because returned value is reversed

borgCore.sol#L621
borgCore.sol#L627
borgCore.sol#L639

Recommendation

We recommend prohibiting adding parameter constraints by `updatePolicy()`, `addSignedRangeParameterConstraint()`, `addUnsignedRangeParameterConstraint()`, and `addExactMatchParameterConstraint()` in case of `blacklist` mode

Client's commentary

> Blacklist mode should have parameters constraints that work exactly like the whitelist mode. If FullAccessOrBlock is not set to true, in isMethodCallAllowed(), if paramOffsets are > 0 for that method, there will constraint checks that work the same as whitelist mode. This is to still give the option of granularity at the parameter/method level with with blacklist mode.

L-60	A lack of `metaVesTController` zero checks in `optimisticGrantImplant's`, `daoVetoGrantImplant's`, and `daoVoteGrantImplant's` constructors
Severity	Low
Status	Fixed in 3357105a

Description

There are no checks that `metaVesTController` is not equal to `address(0)` in `optimisticGrantImplant's`, `daoVetoGrantImplant's`, and `daoVoteGrantImplant's` constructors.

optimisticGrantImplant.sol#L62
daoVetoGrantImplant.sol#L126
daoVoteGrantImplant.sol#L134

Recommendation

We recommend adding these checks.

Client's commentary

> We have added the zero address checks for MetaVest in the constructor of these 3 Implants.

L-61	Incorrect comment to `borgCore.removePolicyMethod()`
Severity	Low
Status	Fixed in 3357105a

Description

The comment to `borgCore.removePolicyMethod()` should be fixed.

borgCore.sol#L319

Recommendation

We recommend changing the comment to

```
/// @notice Function to remove a method constraint
/// @notice with all parameter constraints
/// @dev contract and method must be enabled
```

Client's commentary

> We have corrected the comment for removePolicyMethod()

L-62	`isMethodCallAllowed` should have `view` visibility
Severity	Low
Status	Fixed in 3357105a

Description

`isMethodCallAllowed` can be marked as `view` method: borgCore.sol#L599

Recommendation

We recommend changing the visibility of the function.

Client's commentary

> We have updated isMethodCallAllowed visibility to a view method.

L-63	`nonReentrant` modifier
Severity	Low
Status	Fixed in 3357105a

Description

It is better to add a `nonReentrant` modifier to all functions that can send ETH: daoVetoGrantImplant.sol#L226, daoVoteGrantImplant.sol#L329

Recommendation

We recommend adding a `nonReentrant` modifier for functions from the description.

Client's commentary

> We have added the nonReentrant modifier for the highlighted functions.

L-64	`executeSimpleGrant` has two similar checks
Severity	Low
Status	Fixed in 3357105a

Description

`executeSimpleGrant` has two similar checks, and one of them can be removed:
daoVoteGrantImplant.sol#L397-L401

Recommendation

We recommend removing one of the checks.

Client's commentary

> We have removed the duplicate check.

L-65	`vetoImplant` and `voteImplant` contain same logic
Severity	Low
Status	Acknowledged

Description

`vetoImplant` and `voteImplant` contain precisely the same logic, so one of the contracts can removed.

Recommendation

We recommend removing one of the contracts.

Client's commentary

> We did not address this item.

L-66	`updatePolicy()` requires `INT` to be greater than zero
Severity	Low
Status	Fixed in 3357105a

Description

`updatePolicy()` requires `INT` type to be greater than zero, but there could be a situation which requires upper bound to be negative: borgCore.sol#L428

Recommendation

We recommend allowing the upper bound to be negative for the `INT` type.

Client's commentary

> We have changed the validation for INT to not be bound by UINT number range.

L-67	Incorrect comment
Severity	Low
Status	Fixed in 3357105a

Description

This comment should include information about the blacklist: borgCore.sol#L266

Recommendation

We recommend changing comment to `/// @dev remove contract address from the whitelist or blacklist`.

Client's commentary

> We have updated the comment on this function to the suggested.

L-68	Unused Modifier `onlyThis`
Severity	Low
Status	Fixed in 2cc22ab1

Description

This issue has been identified within the `daoVetoImplant` and `daoVoteImplant` contracts.
The `onlyThis` modifier is declared but never used in the contracts.

Recommendation

We recommend removing the `onlyThis` modifier.

Client's Commentary

> We have addressed this by removing the onlyThis modifier on the Implants that do not need it.

L-69	Incorrect Check for `_duration` in Constructor
Severity	Low
Status	Fixed in 2cc22ab1

Description

This issue has been identified in the constructor of the `daoVoteImplant` contract.
The constructor checks the value of `duration` instead of the `_duration` parameter to ensure it does not exceed `MAX_PROPOSAL_DURATION`.
The issue is classified as **Low** severity because it can lead to incorrect initialization of the `duration` value, though it does not pose immediate security risks.

Recommendation

We recommend updating the conditional check in the constructor to verify the `_duration` parameter instead of the `duration` state variable.

Client's Commentary

> We have fixed the conditional check for _duration in the constructor.

L-70	Missing Input Validation for `quorum` and `threshold` in Constructor
Severity	Low
Status	Acknowledged

Description

This issue has been identified within the constructor of the `daoVetoImplant` and `daoVoteImplant` contracts.
The constructor does not validate the input parameters `_quorum` and `_threshold`. Lack of validation could lead to invalid or nonsensical values (e.g., a quorum or threshold exceeding 100%) being set, which could disrupt the governance process or make proposal execution impossible.

Recommendation

We recommend adding input validation checks to ensure that `_quorum` and `_threshold` are within appropriate ranges (e.g., between 0 and 100).

Client's Commentary

> We did not add validation checks for the suggested parameters as we may support many governance types that require # of votes instead of percentages or other formats.

L-71	Missing Zero Address Check in Proposal Creation
Severity	Low
Status	Fixed in b9d43386

Description

This issue has been identified within the `propose` function of the `SnapShotExecutor` contract.
The function does not validate that the `target` address is not the zero address (`address(0)`). As a result, it is possible to create a proposal with a zero address as the target, which cannot be subsequently removed or managed properly.
The issue is classified as **Low** severity because, while it does not directly compromise security, it can lead to proposals being stuck and unremovable. This, in turn, reduces the available limit for creating valid proposals.

Recommendation

We recommend adding a check to ensure that the `target` address is not equal to `address(0)` when creating a new proposal.

L-72	Typo in Error Name: `SnapShotExeuctor_InvalidParams`
Severity	Low
Status	Fixed in b9d43386

Description

There is a typo in the name of the custom error `SnapShotExeuctor_InvalidParams` defined in the `SnapShotExecutor` contract. The correct spelling of the contract name is `SnapShotExecutor`, but the error name mistakenly uses `SnapShotExeuctor` (note the transposition of 'e' and 'c').

Recommendation

We recommend renaming the mentioned error.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes