Numerai

Security Assessment

July 2nd, 2019

Prepared For:
Stephane Gosselin | *Numerai*
stephane@numer.ai

Prepared By:
Robert Tonic | *Trail of Bits*
robert.tonic@trailofbits.com

Michael Colburn | *Trail of Bits*
michael.colburn@trailofbits.com

Executive Summary

Trail of Bits to review the security of Numerai Token from May 20th through May 24th, 2019. We conducted this assessment over the course of two person-weeks with two engineers, reviewing the Solidity smart contracts and migration scripts at commit hash `374de2e9500cef332487a3f92b1959c41b4fd9b8` from the `NMR_monorepo` repository. An additional one-person week was reserved to perform reporting and follow-up review as specified by Numerai.

The Numerai review was oriented towards best-effort identification of potential issues with the migration scripts and process, as well as the implementation of the Numerai smart contracts, and their compliance with the ERC20 specification. When reviewing the migration scripts and processes, a manual approach was used. This review focused on potential failures which could occur both on- and off-chain, as well as implementation details which could cause problems during a migration. When reviewing the Solidity smart contracts, manual review was guided by Slither's static analysis results for a blended approach. This helped to ensure breadth of coverage while also ensuring that each issue reported by Slither and related tools was accurate or without other mitigating circumstances.

The review across both the migrations and smart contracts produced a total of 10 findings, ranging in severity from Medium to Informational. In regard to the Solidity smart contracts, two issues, TOB-NMR-001: createRound in UpgradeDelegate calls delete on a struct containing a mapping and TOB-NMR-002: Missing return on disableContractUpgradability were detected by Slither and manually reviewed for accuracy. These findings were documented as known by Numerai, and appear to have sufficient mitigations. Two issues in the Gnosis wallet, discovered during a prior Trail of Bits assessment, were also identified as affecting Numerai: TOB-NMR-003: The Gnosis wallet has incompatible behaviors between wallet implementations regarding data padding and TOB-NMR-004: The Gnosis wallet does not check for contract existence and may mislead the user about transaction's result detail these findings. Finally, TOB-NMR-010: Solidity compiler optimizations can be dangerous details how the use of the Solidity optimization flag could result in erroneous behavior. The remaining findings detail potential issues which could occur during a migration, both on- and off-chain, and risks related to the project dependencies. Most notably, TOB-NMR-007: Transactions can occur during the upgrade process details how transactions could occur during a migration.

While performing the smart contract review, the Numerai team requested Trail of Bits review the ERC20 compliance of their upgraded Numeraire token. Appendix D details the result of this compliance review, as well as functionality relevant to the ERC20 interface provided by Numerai.

The Numerai system as a whole represents a complex on-chain system, with intricate migration operations. The multi-step migration process requires human interaction for certain validations. Scripted portions of the migration process are also complex, and use hardcoded values and addresses, intricate helpers, and complex success logic which could be prone to developer error. Despite the complexity of the migration process, the Solidity components of the Numerai system are well organized and commented. By continuing efforts to refactor the automated migration process and continuing to assess the Solidity components of the codebase, Numerai is positioned to continue improving their security posture. Appendix C further details general code-quality recommendations Numerai can follow to help continue these efforts.

At the time of assessment, Numerai was in the process of refactoring the migration scripts and process to help address the complexity and findings noted in this report. With this in mind, Numerai should consider further review of the migration process, as well as a more in-depth review on the smart contract components of their system. Due to the brevity of the assessment, the findings in this report represent a limited review of the system as a whole.

Project Dashboard

Application Summary

Name	Numerai
Version	`374de2e9500cef332487a3f92b1959c41b4fd9b8`
Type	Solidity, JavaScript
Platforms	Ethereum

Engagement Summary

Dates	May 20th - May 24th, 2019
Method	Whitebox
Consultants Engaged	2
Level of Effort	3 person-weeks

Vulnerability Summary

Total Medium-Severity Issues	5	■■■■■
Total Low-Severity Issues	3	■■■
Total Informational-Severity Issues	2	■■
Total	10	■■■■■■■■■■

Category Breakdown

Patching	1	■
Access Controls	1	■
Data Validation	2	■■
Error Reporting	3	■■■
Undefined Behavior	2	■■
Configuration	1	■
Total	10	■■■■■■■■■■

Engagement Goals

The engagement was scoped to provide a security assessment of the Numerai Solidity smart contracts and their upgrade scripts and process.

Specifically, we sought to answer the following questions:

- Are there any immediate security concerns in the Solidity smart contracts?
- Are there issues with the repurposing of functions in the delegate contract?
- How could contract migration fail?
- Are the transports used to deploy the migrations secure?
- Do the project dependencies pose a threat to the system?
- Does Numerai properly implement the ERC20 interface?
- What special privileges does Numerai have over the system?

Coverage

Due to the brevity of the assessment, the use of tools such as Echidna and Manticore would have taken too long to configure. This is due to the extensive initialization required to emulate the Numerai system, which requires the loading and configuration of on-chain data to correctly test against. As a result, coverage achieved over the Numerai codebase consisted of manual review and Slither's static analysis.

Manual migration process. The manual portions of the migration process were reviewed for potential errors and concerning operations. The review accounted for manual operations which augment scripted components of the migration.

Endpoint HTTPS usage. The migration scripts have various node endpoints defined within them. Due to the nature of the migration scripts, the defined endpoints were reviewed to ensure connections were established over TLS.

Numeric value handling during migrations. Due to differences in numeric handling between JavaScript and Solidity, there is a possibility where values could overflow or underflow. Furthermore, the parsing of these values must be accounted for. These scenarios, and potentially erroneous operations, were reviewed for correctness.

Error logging and process exits. In many cases the migration script will produce errors to the console, then exit. The appropriate uses of exit codes and error logging were reviewed.

Project dependencies. A cursory review of the project's dependency was performed using `retire.js` and `yarn audit`.

Yarn commands. The Yarn commands defined within the project's `package.json` were reviewed in the context of performing migrations.

Solidity optimizations and version usage. Numerai uses multiple versions of Solidity between their first and second contract versions. They also enable optimizations for both versions of their contracts. The dangers of optimizations and versions were reviewed with the Numerai team.

Gnosis MultiSig wallets. Numerai uses the Gnosis MultiSig wallets in their production deployments. During a previous assessment, Trail of Bits identified two concerns in the Gnosis wallet. These concerns were reviewed in the context of Numerai's deployment.

Slither static analysis false-positive review. Slither was applied to the Numerai codebase. The resulting findings were reviewed to ensure findings were false-positive, and that no true-positive findings surfaced. This review helped guide manual code review of the codebase.

Slither upgradability review of the proxy contracts. Slither's upgradability check was applied to the Numerai proxy contracts. Manual review of the proxy contracts was also performed, guided by the results of Slither.

SafeMath-like library review. Numerai has several libraries with functionality similar to the OpenZeppelin SafeMath library. These libraries were reviewed to ensure mathematical correctness.

Repurposed UpgradeDelegate contract functions review. This contract, which is used to execute the on-chain migration logic, has several functions repurposed in a way that diverges from their naming. These functions were reviewed for correctness.

ERC20 race condition mitigation review. Since Numerai's token must follow the ERC20 interface, the race condition inherent to the ERC20 standard is present. The race condition mitigation implemented by Numerai was reviewed for correctness.

Recommendations Summary

This section aggregates all the recommendations made during the engagement. Short-term recommendations address the immediate causes of issues. Long-term recommendations pertain to the development process and long-term design goals.

Short Term

Ensure that when calling `createRound` in the `UpgradeDelegate` contract that the relevant stakes have already been cleared via `destroyStake`. Calling delete on a `Round` will not clear the `stakes` mapping a `Round` contains.

Do not rely on the return value of `disableContractUpgradability` when disabling upgradability. If necessary, use the public getter for the `contractUpgradable` variable to check that upgradability has been properly disabled.

Ensure the calling convention used with the Gnosis wallet is compatible with the data provided. Also ensure the calling convention is compatible with the contract method being called.

Ensure all calls attempt to check the existence of a contract at the destination address. If contract existence is not checked before a transaction is submitted, the system may be misled by the result of the transaction.

Ensure dependencies are up-to-date. Several node modules have been reported and documented as malicious, executing malicious code when installing dependencies to projects. Keep modules up to date and verify their integrity after installation.

Avoid chaining multiple groups of commands together. These should be expanded into their own named Yarn commands, allowing operators to validate the correctness of each group's execution.

Ensure the system is appropriately paused and prevents users from changing the state of the system for the duration of the upgrade. This will help avoid concerns regarding the migration of system state for the duration of an upgrade.

Ensure on-chain calls are not successful by explicitly comparing the `succeeded` variable to false instead of `shouldSucceed` in the migrations. This will help prevent a call from succeeding when the function returns false.

Ensure logging captures raised errors and errors are appropriately returned to the caller. If errors are not appropriately raised and reported, they may propagate unnoticed, affecting the integrity and validity of the system.

Measure the gas savings from optimizations, and carefully weigh that against the possibility of an optimization-related bug. Optimizations could pose additional risk for limited benefits. Carefully review this tradeoff.

Long Term

Use Slither to detect when `delete` is being called on structs containing mappings, and when functions are missing appropriate return statements. Slither's static analysis can help catch common bugs like this during the development process, preventing them from being accidentally deployed or missed during an assessment.

Document the different calling conventions in the Gnosis wallet well, including the affected versions used by Numerai. Because a fix is provided in later versions of the wallet, consider upgrading to use an unaffected version of the wallet.

Ensure the Gnosis wallet's lack of contract existence checking is well documented and accounted for in any systems depending on the Gnosis wallet. This will help prevent users from being misled by the result of a transaction.

Consider integrating automated dependency auditing into the development workflow. If dependencies cannot be updated when a vulnerability is disclosed, ensure the Numerai codebase does not use, and is not affected by, the vulnerable functionality of the dependency.

Ensure the Yarn commands are well tested and documented. Refactor and expand the existing Yarn commands into individual steps for an operator to invoke. This allows the process to be better documented, and prevents unintended executions in the event of error.

Ensure both upgrade and migration paths account for changes in system state. Upgrades and migrations can cause changes in the operations performed against contract data, as well as the contract data itself. It is important to ensure the integrity and validity of the system's state is preserved both during and after the upgrade and migration process is performed.

Consider refactoring the `call` function in the migration script to move assertion logic to its own wrapper which can be re-used across the other functions. Expand testing to ensure call return values are correct with all given parameter values.

Ensure error handling appropriately raises and returns errors to the user. Avoid implicitly handling errors. This type of functionality could lead to dangerous behavior with future changes to the system.

Monitor the development and adoption of Solidity compiler optimizations to assess its maturity. Due to concerns in the stability and security of the optimizations produced by the Solidity compiler, optimizations should be carefully reviewed before use.

Findings Summary

#	Title	Type	Severity
1	createRound in UpgradeDelegate calls delete on a struct containing a mapping	Data Validation	Low
2	Missing return on disableContractUpgradability	Undefined Behavior	Low
3	The Gnosis wallet has incompatible behaviors between wallet implementations regarding data padding	Data Validation	Medium
4	The Gnosis wallet does not check for contract existence and may mislead the user about the transaction's result	Error Reporting	Medium
5	Project dependencies contain vulnerabilities	Patching	Low
6	Yarn commands can fail and still continue execution	Undefined Behavior	Medium
7	Transactions can occur during the upgrade process	Access Controls	Medium
8	The call function has confusing success logic and comorbid return values	Error Reporting	Informational
9	Error logging is disabled for calls to contracts on-chain	Error Reporting	Medium
10	Solidity compiler optimizations can be dangerous	Configuration	Informational

1. createRound in UpgradeDelegate calls delete on a struct containing a mapping

Severity: Low
Difficulty: Low
Type: Data Validation
Finding ID: TOB-NMR-001
Target: `NMR_monorepo/0.4-contracts/UpgradeDelegate.sol`

Description
The `createRound` function in the `UpgradeDelegate` contract has been repurposed to clear out active rounds from the tournament contract. As part of this process, it loops over the `tournaments` state variable and deletes all of its rounds. However, a `Round` struct contains a mapping called `stakes`. Calling `delete` on a `Round` will delete the other variables but have no effect on the `stakes` mapping it contains.

`Stakes` that are not deleted properly may lead to unexpected storage values in the future.

Exploit Scenario
Alice begins to execute a migration of the Numerai system. As part of this migration, `createRound` is called on the tournament data before clearing stakes. After the migration completes, a future tournament round overlaps with the orphaned state and results in unintended side effects.

Recommendation
Short term, ensure that when calling `createRound` in the `UpgradeDelegate` contract that the relevant stakes have already been cleared via `destroyStake`. Calling delete on a `Round` will not clear the `stakes` mapping a `Round` contains.

Long term, use Slither to detect when `delete` is being called on structs containing mappings.

2. Missing return on disableContractUpgradability

Severity: Low
Difficulty: Low
Type: Undefined Behavior
Finding ID: TOB-NMR-002
Target: `NMR_monorepo/0.4-contracts/NumeraireBackend.sol`

Description
The function `disableContractUpgradability` in `NumeraireBackend` is intended to prevent the contract from being upgraded beyond its current state. The function declaration indicates a boolean return value, however the function body does not include a return statement. As a result, this function will always return false. This behavior is noted as a known bug with a comment above the function.

However, instead of using this function, the `NumeraireDelegateV3` contract directly toggles the `contractUpgradable` variable to false in its `constructor`. As a result, this is unlikely to be an issue once the upgrade has been carried out.

Exploit Scenario
Alice begins executing a new migration of the Numerai system. As part of this new migration, `disableContractUpgradability` is called. The calling function checks the return value and reverts if the function does not return `true`. Due to the missing return statement in `disableContractUpgradability`, the calling function will always revert and Alice will need to redeploy the affected code in order for the migration to progress.

Recommendation
Short term, do not rely on the return value of `disableContractUpgradability` when disabling upgradability. If necessary, use the public getter for the `contractUpgradable` variable to check that upgradability has been properly disabled.

Long term, use Slither to detect when functions are missing appropriate return statements.

3. The Gnosis wallet has incompatible behaviors between wallet implementations regarding data padding

Severity: Medium	Difficulty: Low
Type: Data Validation	Finding ID: TOB-NMR-003
Target: gnosis/MultiSigWallet:MultiSigWallet/contracts/MultiSigWallet.sol

Description
`MultiSigWalletWithDailyLimit` is meant to be an extension of `MultiSigWallet` with a daily limit. In versions of the wallet prior to commit `95d51ae89ddec56859720fbb28cfe9d6732a26cf`, the use of different calling patterns results in different transaction behaviors if the length of the data sent is not a multiple of 32 bytes.

`MultiSigWallet` calls the destination using assembly code:

```
   function external_call(address destination, uint value, uint dataLength, bytes data)
internal returns (bool) {
        bool result;
        assembly {
            let x := mload(0x40)   // "Allocate" memory for output (0x40 is where "free
memory" pointer is stored by convention)
            let d := add(data, 32) // First 32 bytes are the padded length of data, so
exclude that
            result := call(
                sub(gas, 34710),   // 34710 is the value that solidity is currently emitting
                                   // It includes callGas (700) + callVeryLow (3, to pay for
SUB) + callValueTransferGas (9000) +
                                   // callNewAccountGas (25000, in case the destination
address does not exist and needs creating)
                destination,
                value,
                d,
                dataLength,        // Size of the input (in bytes) - this is what fixes the
padding problem
                x,
                0                  // Output is ignored, therefore the output size is zero
            )
        }
        return result;
    }
```

Figure 3.1: The `external_call` function of the `MultiSigWallet`, performing a contract call with an assembly implementation.

`MultiSigWalletWithDailyLimit` (prior to the aforementioned commit) uses a low-level call:

```
if (txn.destination.call.value(txn.value)(txn.data))
```

Figure 3.2: The external call used in the `MultiSigWalletWithDailyLimit`.

Low-level calls pad the data to a multiple of 32 bytes. Assembly calls do not. As a result, a transaction with a data length which is not a multiple of 32 bytes will have different behavior if sent by `MultiSigWallet` or `MultiSigWalletWithDailyLimit`.

This has been documented and reported by Trail of Bits during a previous engagement here.

Exploit Scenario
Alice submits two calls with a data length that is not a multiple of 32 bytes. One call is handled by the `MultiSigWalletWithDailyLimit.sol` and the other by the `MultiSigWallet.sol`. Because of the different calling conventions, the two transactions are not treated the same, potentially resulting in unexpected behavior.

Recommendation
Short term, ensure that the calling convention used is compatible with the data provided, as well as the contract method being called.

Long term, ensure this difference in calling convention is well documented, as well as the affected versions. Because this is fixed in later commits, consider upgrading to an unaffected version.

4. The Gnosis wallet does not check for contract existence and may mislead the user about the transaction's result

Severity: Medium	Difficulty: Low
Type: Error Reporting	Finding ID: TOB-NMR-004
Target: gnosis/MultiSigWallet:MultiSigWallet/contracts/MultiSigWallet.sol and gnosis/MultiSigWallet:MultiSigWallet/contracts/MultiSigWalletWithDailyLimit.sol

Description
A failure to check for a contract's existence may mislead a user into thinking that a failed transaction was successful.

`MultiSigWallet` and `MultiSigWalletWithDailyLimit` use low-level and assembly calls to execute external transactions.

```
   function external_call(address destination, uint value, uint dataLength, bytes data)
internal returns (bool) {
        bool result;
        assembly {
            let x := mload(0x40)   // "Allocate" memory for output (0x40 is where "free
memory" pointer is stored by convention)
            let d := add(data, 32) // First 32 bytes are the padded length of data, so
exclude that
            result := call(
                sub(gas, 34710),   // 34710 is the value that solidity is currently emitting
                                   // It includes callGas (700) + callVeryLow (3, to pay for
SUB) + callValueTransferGas (9000) +
                                   // callNewAccountGas (25000, in case the destination
address does not exist and needs creating)
                destination,
                value,
                d,
                dataLength,        // Size of the input (in bytes) - this is what fixes the
padding problem
                x,
                0                  // Output is ignored, therefore the output size is zero
            )
        }
        return result;
    }
```

Figure 4.1: The `external_call` *implementation in* `MultiSigWallet`*, showing an absence of contract existence check.*

This has been documented and reported by Trail of Bits during a previous engagement here.

Exploit Scenario
Ailce uses the Gnosis wallet to submit a call to an address believed to be a contract. Unbeknownst to Alice, the contract had previously been destructed. Due to a lack of

contract existence checks in the Gnosis wallet, Alice's call returns a success even though it did not successfully execute.

Recommendation
Short term, ensure all calls check the existence of a contract at the destination address.

Long term, ensure this limitation is well documented and accounted for in any systems depending on the Gnosis wallet.

5. Project dependencies contain vulnerabilities

Severity: Low
Difficulty: Low
Type: Patching
Finding ID: TOB-NMR-005
Target: `NMR_monorepo dependencies`

Description
Although dependency scans did not yield a direct threat to the Numerai codebase, `yarn audit` has identified dependencies with known vulnerabilities. Because of the sensitivity of the deployment code and its environment, it is important to ensure dependencies are not malicious. Problems with dependencies in the JavaScript community could have a significant effect on the Numerai system as a whole. The output detailing the identified issues have been included in Appendix B.

Exploit Scenario
Alice installs the dependencies for Numerai on a clean machine. Unknown to Alice, a dependency of the project has become malicious or exploitable. Alice subsequently uses the dependency, disclosing sensitive information to an unknown actor.

Recommendation
Short term, ensure dependencies are up-to-date. Several node modules have been reported and documented as malicious, executing malicious code when installing dependencies to projects. Keep modules up to date and verify their integrity after installation.

Long term, consider integrating automated dependency auditing in the development workflow. If dependencies cannot be updated when a vulnerability is disclosed, ensure the Numerai codebase does not use, and is not affected by, the vulnerable functionality of the dependency.

6. Yarn commands can fail and still continue execution

Severity: Medium
Type: Undefined behavior
Target: `NMR_monorepo/package.json`

Difficulty: Low
Finding ID: TOB-NMR-006

Description
When performing an upgrade or migration, commands are executed through Yarn scripts, defined in the package.json. These commands are chained together through the use of `&&`, and the chains are grouped by `;`. Because `;` is used, the execution of a group can fail but execution can continue to other groups of commands. This can cause unintended behavior of commands executing after a failed group of commands. An example of such a failure has been included in Figure 6.1.

```
user@user:~/Desktop/NMR_monorepo$ yarn exampleFail
yarn run v1.16.0
$ true && echo 'executed'; python -c 'raise Exception()' && echo 'not executed'; true &&
echo 'executed as well'
executed
Traceback (most recent call last):
  File "<string>", line 1, in <module>
Exception
executed as well
Done in 0.15s.
```

Figure 6.1: An example of logic failure in a Yarn command resulting in continued execution.

Exploit Scenario
Alice executes a Yarn command to perform the automated portions of an upgrade or migration. When the Yarn command executes, a group of commands fails to execute successfully, but the Yarn command continues to execute. This leads to a potentially erroneous upgrade or migration.

Recommendation
Short term, avoid chaining multiple groups of commands together. These should be expanded into their own named Yarn commands, allowing operators to validate the correctness of each group's execution.

Long term, ensure the Yarn commands are well tested and documented. Refactor and expand the existing Yarn commands into individual steps for an operator to invoke. This supports better documentation of the process, and prevents unintended executions in the event of error.

7. Transactions can occur during the upgrade process

Severity: Medium
Difficulty: Low
Type: Access Controls
Finding ID: TOB-NMR-007
Target: `Numerai smart contracts`

Description
The current process to upgrade does not account for transactions which could occur during an upgrade. This could lead to data loss, and consequently an invalid system state.

Exploit Scenario
Alice attempts to perform an upgrade of the Numerai system following the defined upgrade process. Alice retrieves stake data from BigQuery, validates it against on-chain values, then stores it on disk to continue the upgrade. Subsequently, a user of Alice's Numerai system interacts with the system and changes its state. Because Alice did not account for users interacting with the system during an upgrade, Alice's data does not represent the active system state, or the transactions that occur after data has been stored on disk.

Recommendation
Short term, ensure the system is appropriately paused and prevents users from changing the state of the system for the duration of the upgrade. This will help avoid concerns regarding the migration of system state for the duration of an upgrade.

Long term, ensure both upgrade and migration paths account for changes in system state. Upgrades and migrations can cause changes in the operations performed against contract data, as well as the contract data itself. It is important to ensure the integrity and validity of the system's state is preserved both during and after the upgrade and migration process is performed.

8. The call function has confusing success logic and comorbid return values

Severity: Informational
Difficulty: Low
Type: Error Reporting
Finding ID: TOB-NMR-008
Target: `NMR_monorepo/scripts/test/migrationProcedure.js`

Description
Within `$PROJECT_ROOT/scripts/test/migrationProcedure.js` the `call` helper function is used to invoke on-chain contract methods. As a parameter, the `call` function accepts a `shouldSucceed` parameter, which is intended to be used when determining if a particular contract call should produce an error or not. The logic used to determine whether an error occurred is based on two variables: `shouldSucceed` and `succeeded`. If `shouldSucceed` is false and the contract call succeeds, this results in `succeeded` being true. Given these conditions, the function call will result in a return value of false because `succeeded !==` `shouldSucceed`, despite the call being successful.

```
async function call(
    title,
    instance,
    method,
    args,
    from,
    value,
    gas,
    gasPrice,
    shouldSucceed,
    assertionCallback
  ) {
    let succeeded = true
    returnValues = await instance.methods[method](...args).call({
      from: from,
      value: value,
      gas: gas,
      gasPrice: gasPrice
    }).catch(error => {
      //console.error(error)
      succeeded = false
    })

    if (succeeded !== shouldSucceed) {
      return false
    } else if (!shouldSucceed) {
      return true
    }
...
```
Figure 8.1: A snippet of the `call` function definition highlighting a portion of the success logic.

Furthermore, if the `shouldSucceed` logic does not result in a `return`, another check is performed to set the value of `assertionsPassed` which is then returned. Because of this, the return value of the `call` function can be difficult to interpret correctly.

```
...
    let assertionsPassed
    try {
      assertionCallback(returnValues)
      assertionsPassed = true
    } catch(error) {
      assertionsPassed = false
      console.log(error);
    }

    return assertionsPassed
  }
```

Figure 8.2: Another snippet of the `call` *function definition, highlighting a second portion of success logic.*

Exploit Scenario

Alice invokes the `call` function with the `shouldSucceed` parameter being false. Alice's call succeeds and false is still returned.

Recommendation

Short term, ensure the on-chain call is not successful by explicitly comparing the `succeeded` variable to false instead of `shouldSucceed`. This will help avoid a situation in which a call succeeds but the function returns false.

Long term, consider refactoring the `call` function to move assertion logic to its own wrapper which can be re-used across the other functions. Expand testing to ensure call return values are correct with all given parameter values.

9. Error logging is disabled for calls to contracts on-chain

Severity: Medium
Difficulty: Low
Type: Error Reporting
Finding ID: TOB-NMR-009
Target: `NMR_monorepo/scripts/test/migrationProcedure.js`

Description

Within `$PROJECT_ROOT/scripts/test/migrationProcedure.js` the `call` helper function is used to invoke on-chain contract methods. When the `call` function is executed and an error occurs, error-handling logic denotes that an error has occurred, but this error is not captured, returned, or logged. This could cause an error in the migration process, and the user will be unable to easily identify the error that was raised.

```
...
    let succeeded = true
    returnValues = await instance.methods[method](...args).call({
      from: from,
      value: value,
      gas: gas,
      gasPrice: gasPrice
    }).catch(error => {
      //console.error(error)
      succeeded = false
    })
...
```

Figure 9.1: The `call` *function, containing the commented* `console.error`*.*

Exploit Scenario

Alice executes a contract migration. During the migration, a call to a contract fails. Due to the `console.error` being commented out, the error is not logged and Alice is unable to identify the error that occurred.

Recommendation

Short term, ensure logging captures raised errors and errors are appropriately returned to the caller. If errors are not appropriately raised and reported, errors may propagate unnoticed, affecting the integrity and validity of the system.

Long term, ensure error handling appropriately raises and returns errors to the user. Avoid implicitly handling errors. This type of functionality could lead to dangerous behavior with future changes to the system.

10. Solidity compiler optimizations can be dangerous

Severity: Informational　　Difficulty: Low
Type: Configuration　　Finding ID: TOB-NMR-010
Target: `NMR_monorepo/truffle-config.js` and `NMR_monorepo/truffle-config-0.4.js`

Description
Numerai has enabled optional compiler optimizations in Solidity.

There have been several bugs with security implications related to optimizations. Moreover, optimizations are actively being developed. Solidity compiler optimizations are disabled by default. It is unclear how many contracts in the wild actually use them. Therefore, it is unclear how well they are being tested and exercised.

High-severity security issues due to optimization bugs have occurred in the past. A high-severity bug in the emscripten-generated solc-js compiler used by Truffle and Remix persisted until just a few months ago. The fix for this bug was not reported in the Solidity CHANGELOG.

A compiler audit of Solidity from November, 2018 concluded that the optional optimizations may not be safe. Moreover, the Common Subexpression Elimination (CSE) optimization procedure is "implemented in a very fragile manner, with manual access to indexes, multiple structures with almost identical behavior, and up to four levels of conditional nesting in the same function." Similar code in other large projects have resulted in bugs.

There are likely latent bugs related to optimization, and/or new bugs that will be introduced due to future optimizations.

Exploit Scenario
A latent or future bug in Solidity compiler optimizations—or in the Emscripten transpilation to `solc-js`—causes a security vulnerability in the Numerai contracts.

Recommendation
Short term, measure the gas savings from optimizations, and carefully weigh that against the possibility of an optimization-related bug. Optimizations could pose additional risk for limited benefits. Carefully review this tradeoff.

Long term, monitor the development and adoption of Solidity compiler optimizations to assess its maturity. Due to concerns over the stability and security of the optimizations produced by the Solidity compiler, optimizations should be carefully reviewed before use.

A. Vulnerability Classifications

Vulnerability Classes	
Class	**Description**
Access Controls	Related to authorization of users and assessment of rights
Auditing and Logging	Related to auditing of actions or logging of problems
Authentication	Related to the identification of users
Configuration	Related to security configurations of servers, devices or software
Cryptography	Related to protecting the privacy or integrity of data
Data Exposure	Related to unintended exposure of sensitive information
Data Validation	Related to improper reliance on the structure or values of data
Denial of Service	Related to causing system failure
Error Reporting	Related to the reporting of error conditions in a secure fashion
Patching	Related to keeping software up to date
Session Management	Related to the identification of authenticated users
Timing	Related to race conditions, locking or order of operations
Undefined Behavior	Related to undefined behavior triggered by the program

Severity Categories	
Severity	**Description**
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth
Undetermined	The extent of the risk was not determined during this engagement
Low	The risk is relatively small or is not a risk the customer has indicated is important
Medium	Individual user's information is at risk, exploitation would be bad for client's reputation, moderate financial impact, possible legal

	implications for client
High	Large numbers of users, very bad for client’s reputation, or serious legal or financial implications

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploit was not determined during this engagement
Low	Commonly exploited, public tools exist or can be scripted that exploit this flaw
Medium	Attackers must write an exploit, or need an in-depth knowledge of a complex system
High	The attacker must have privileged insider access to the system, may need to know extremely complex technical details or must discover other weaknesses in order to exploit this issue

B. Yarn dependency vulnerability scans

```
bobby@bobby-VirtualBox:~/Desktop/NMR_monorepo$ yarn audit
yarn audit v1.16.0
┌───────────────┬──────────────────────────────────────────────────────────────┐
│ high          │ Arbitrary File Overwrite                                     │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Package       │ fstream                                                      │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Patched in    │ >=1.0.12                                                     │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Dependency of │ truffle-hdwallet-provider                                    │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Path          │ truffle-hdwallet-provider > web3 > web3-bzz > swarm-js >     │
│               │ tar.gz > fstream                                             │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ More info     │ https://www.npmjs.com/advisories/886                         │
└───────────────┴──────────────────────────────────────────────────────────────┘
┌───────────────┬──────────────────────────────────────────────────────────────┐
│ high          │ Arbitrary File Overwrite                                     │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Package       │ fstream                                                      │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Patched in    │ >=1.0.12                                                     │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Dependency of │ truffle-hdwallet-provider                                    │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Path          │ truffle-hdwallet-provider > web3 > web3-bzz > swarm-js >     │
│               │ tar.gz > tar > fstream                                       │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ More info     │ https://www.npmjs.com/advisories/886                         │
└───────────────┴──────────────────────────────────────────────────────────────┘
┌───────────────┬──────────────────────────────────────────────────────────────┐
│ high          │ Arbitrary File Overwrite                                     │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Package       │ fstream                                                      │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Patched in    │ >=1.0.12                                                     │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Dependency of │ web3                                                         │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Path          │ web3 > web3-bzz > swarm-js > tar.gz > fstream                │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ More info     │ https://www.npmjs.com/advisories/886                         │
└───────────────┴──────────────────────────────────────────────────────────────┘
┌───────────────┬──────────────────────────────────────────────────────────────┐
│ high          │ Arbitrary File Overwrite                                     │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Package       │ fstream                                                      │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Patched in    │ >=1.0.12                                                     │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Dependency of │ web3                                                         │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ Path          │ web3 > web3-bzz > swarm-js > tar.gz > tar > fstream          │
├───────────────┼──────────────────────────────────────────────────────────────┤
│ More info     │ https://www.npmjs.com/advisories/886                         │
```

```
└───────────────┴────────────────────────────────────────────────────────────┘
┌───────────────┬────────────────────────────────────────────────────────────┐
│ high          │ Arbitrary File Overwrite                                   │
├───────────────┼────────────────────────────────────────────────────────────┤
│ Package       │ tar                                                        │
├───────────────┼────────────────────────────────────────────────────────────┤
│ Patched in    │ >=2.2.2 <3.0.0 || >=4.4.2                                  │
├───────────────┼────────────────────────────────────────────────────────────┤
│ Dependency of │ truffle-hdwallet-provider                                  │
├───────────────┼────────────────────────────────────────────────────────────┤
│ Path          │ truffle-hdwallet-provider > web3 > web3-bzz > swarm-js >   │
│               │ tar.gz > tar                                               │
├───────────────┼────────────────────────────────────────────────────────────┤
│ More info     │ https://www.npmjs.com/advisories/803                       │
└───────────────┴────────────────────────────────────────────────────────────┘
┌───────────────┬────────────────────────────────────────────────────────────┐
│ high          │ Arbitrary File Overwrite                                   │
├───────────────┼────────────────────────────────────────────────────────────┤
│ Package       │ tar                                                        │
├───────────────┼────────────────────────────────────────────────────────────┤
│ Patched in    │ >=2.2.2 <3.0.0 || >=4.4.2                                  │
├───────────────┼────────────────────────────────────────────────────────────┤
│ Dependency of │ web3                                                       │
├───────────────┼────────────────────────────────────────────────────────────┤
│ Path          │ web3 > web3-bzz > swarm-js > tar.gz > tar                  │
├───────────────┼────────────────────────────────────────────────────────────┤
│ More info     │ https://www.npmjs.com/advisories/803                       │
└───────────────┴────────────────────────────────────────────────────────────┘
6 vulnerabilities found - Packages audited: 249008
Severity: 6 High
Done in 3.11s.
```

```
bobby@bobby-VirtualBox:~/Desktop/NMR_monorepo$ ./node_modules/retire/bin/retire -v --jspath
. --nodepath .
Loading from cache:
https://raw.githubusercontent.com/RetireJS/retire.js/master/repository/jsrepository.json
Reading /tmp/.retire-cache/1558713496925.json ...
Loading from cache:
https://raw.githubusercontent.com/RetireJS/retire.js/master/repository/npmrepository.json
Reading /tmp/.retire-cache/1558713497550.json ...
/home/bobby/Desktop/NMR_monorepo/node_modules/bignumber.js/bower.json
 ↳ bignumber.js 7.2.1
/home/bobby/Desktop/NMR_monorepo/node_modules/buffer-to-arraybuffer/bower.json
 ↳ buffer-to-arraybuffer 0.0.4
/home/bobby/Desktop/NMR_monorepo/node_modules/core-js/bower.json
 ↳ core.js 2.6.5
/home/bobby/Desktop/NMR_monorepo/node_modules/immediate/bower.json
 ↳ immediate 3.2.1
/home/bobby/Desktop/NMR_monorepo/node_modules/js-sha3/bower.json
 ↳ js-sha3 0.5.7
/home/bobby/Desktop/NMR_monorepo/node_modules/semaphore/bower.json
 ↳ semaphore.js 1.0.3
/home/bobby/Desktop/NMR_monorepo/node_modules/sprintf-js/bower.json
 ↳ sprintf 1.0.3
/home/bobby/Desktop/NMR_monorepo/node_modules/tslib/bower.json
```

```
 ↳ tslib 1.9.3
/home/bobby/Desktop/NMR_monorepo/node_modules/browserify-sha3/node_modules/js-sha3/bower.jso
n
 ↳ js-sha3 0.6.1
```

C. Code Quality Recommendations

The following recommendations are not associated with specific vulnerabilities. However, they enhance code readability and may prevent the introduction of vulnerabilities in the future.

General Recommendations

- Thoroughly document constants throughout the codebase.

Migration and Upgrade process Recommendations

- Avoid hardcoding values. Configuration should be possible through centralized locations. This helps to avoid subtle differences across the codebase introduced by developer error.
- Avoid manual operations requiring operator input. Scripted operations help prevent human error.
- Reduce the complexity of the deployment, initialization, and validation processes by abstracting the implementation logic and properly modularizing the migration script.
- Ensure the appropriate use of `process.exit`, and verify the correct return code is provided. This will help prevent accidental false-positive and false-negative builds when scripted.

D. ERC20 Compliance

The Numerai team requested Trail of Bits review compliance with ERC20 of their upgraded Numeraire token.

`NumeraireBackend` is the entrypoint for token logic of the Numeraire token. Part of the logic is implemented in this backend contract and part is accessed via a `delegatecall` proxy to the `NumeraireDelegateV3` contract. These contracts both contain logic and data, in addition to the ERC20 token functionality. Both contracts also inherit `NumeraireShared`, where some ERC20 variables and additional functions are declared.

The table below outlines the functions and events required to implement the ERC20 token standard, and the conformance of the Numeraire token to that standard. A green background indicates the function or event correctly implements the function and emits events at the appropriate times. Additional details about the function and event implementations are noted alongside each as necessary.

Functions	
`totalSupply`	`totalSupply` has been set to 11 million as part of the migration to the new delegate contract and there is no way to increase it. `totalSupply` can only be reduced by a user invoking the `_burn` or `_burnFrom` functions through `mint` or `numeraiTransfer`, respectively.
`balanceOf`	
`transfer`	
`transferFrom`	Cannot be called with an owner address or the `NumeraireBackend` contract address as `_from`. Comments direct those transfers to happen through `numeraiTransfer`, however, that function has been repurposed to `_burnFrom` tokens. This means that tokens transferred to the contract itself are unrecoverable.
`approve`	Can only be called when `_value` is 0 or the `allowance` is 0, otherwise `changeApproval` must be used as a mitigation of the ERC20 race condition.
`allowance`	
Events	
`Transfer`	

Approval	An extra Approval event is also emitted when tokens are burnt via _burnFrom. However, no additional Approval is emitted when tokens are transferred via transferFrom so Approval events cannot be relied on to fully capture the current state of an allowance.

Numerai Special Privileges

The Numerai team requested Trail of Bits investigate the amount of control Numerai is able to exert over its Numeraire (NMR) ERC20 token.

Some Numeraire contracts have built-in support for ownership by multiple addresses. Some privileged functionality can be invoked by just one owner, while other functionality requires a threshold of owners to approve it. In this section, we describe what privileged functionality the Numeraire contract owners are able to perform and how this may impact NMR holders.

Many `NumeraireBackend` contract functions are protected by two modifiers: `stopInEmergency` and `onlyPayloadSize`.

- `onlyPayloadSize` is a modifier that some contracts have included in the past to mitigate the ERC20 short address attack.
- `stopInEmergency` allows any one of the contract owners to pause most of the contract's functionality, including token transfers and approvals. A threshold of owners is required to unpause the contract or disable toggling pausability altogether.

`NumeraireBackend` contains one function to be aware of: `withdraw`. This function can only be invoked by the `Relay` contract owner or manager. It allows the `Relay` owner to perform transfers from accounts managed by Numerai, tracked in addresses below one million (0xF4240). Tokens held by addresses greater than 0xF4240 cannot be transferred by calls to `withdraw`. This behavior is only available while the `Relay` contract is active. The `Relay` can be disabled by its owner and once disabled can never be re-enabled, preventing further calls to `withdraw`. The only other path to invoke `withdraw` is through `Relay.burnZeroAddress` which burns tokens sent to the 0 address or a designated burn address.

By default, delegate contract upgradability is still enabled. However, it can be disabled in one of two ways: a call to `Relay.disableTokenUpgradability` by the `Relay` owner or `NumeraireBackend.disableContractUpgradability` by a threshold of `NumeraireBackend` owners. Once disabled, there is no mechanism to re-enable contract upgradability and the delegate implementation cannot be changed.

An owner of the `NumeraireBackend` contract is able to remove non-NMR ERC20 tokens or any ethers that have been sent to the `NumeraireBackend` contract through `claimTokens`.

In an attempt to limit these ownership capabilities, during the upgrade process, Numerai deleted the owners from the `NumeraireBackend` by executing the `UpgradeDelegate.numeraiTransfer` function, visible in Figure D.1. This function first iterates through the length of the `owners` array, deleting each indexed address while also removing it from the `ownersIndex`. Hardcoded addresses known to be previous owners of the contract are then explicitly removed from the `ownerIndex`, ensuring removal.

```
       // Clear ownership - BE SURE THAT ownerIndex IS CLEARED PROPERLY!
        clearPending();
        for (uint256 z = 0; z < owners.length; z++) {
            if (owners[z] != address(0)) {
                delete ownerIndex[owners[z]];
                delete owners[z];
            }
        }
        delete ownerIndex[address(0)]; // just in case...

        // Double check all previous owners have been cleared
        address[28] memory previousOwners = [
            address(0x9608010323ed882a38ede9211d7691102b4f0ba0),
            address(0xb4207031bb146e90cab72230e0030823e02b923b),
            address(0x0387f0f731b518590433cd0b37e5b3eb9d3aef98),
            address(0x8ad69ae99804935d56704162e3f6a6f442d2ed4a),
            address(0x16e7115f6595668ca34f0cae8e76196274c14ff7),
            address(0xdc6997b078c709327649443d0765bcaa8e37aa6c),
            address(0x257988b95ee87c30844abec7736ff8a7d0be2eb1),
            address(0x70153f8f89f6869037fba270233409844f1f2e2e),
            address(0xae0338fefd533129694345659da36c4fe144e350),
            address(0x444ab8ad5c74f82fada4765f4a4e595109903f11),
            address(0x54479be2ca140163031efec1b7608b9759ec897a),
            address(0x193b78eb3668982f17862181d083ff2e2a4dcc39),
            address(0x6833b2469e80ef0c72aa784e27b2666ab43568f5),
            address(0x1d68938194004722b814f00003d3eca19357344a),
            address(0x54479be2ca140163031efec1b7608b9759ec897a),
            address(0x9608010323ed882a38ede9211d7691102b4f0ba0),
            address(0x638141cfe7c64fe9a22400e7d9f682d5f7b3a99b),
            address(0x769c72349aa599e7f63102cd3e4576fd8f306697),
            address(0xe6a2be73d9f8eb7a56f27276e748b05f7d6d7500),
            address(0x707ad29e43f053d267854478642e278e78243666),
            address(0x193b78eb3668982f17862181d083ff2e2a4dcc39),
            address(0x6833b2469e80ef0c72aa784e27b2666ab43568f5),
            address(0x1d68938194004722b814f00003d3eca19357344a),
            address(0x22926dd58213ab6601addfa9083b3d01b9e20fe8),
            address(0x769c72349aa599e7f63102cd3e4576fd8f306697),
            address(0xe6a2be73d9f8eb7a56f27276e748b05f7d6d7500),
            address(0x638141cfe7c64fe9a22400e7d9f682d5f7b3a99b),
            address(0x707ad29e43f053d267854478642e278e78243666)
        ];
        for (uint256 y = 0; y < previousOwners.length; y++) {
            delete ownerIndex[previousOwners[y]];
        }
```

Figure D.1: The deletion of the previous owners from the `NumeraireBackend` *proxy contract when executing* `UpgradeDelegate.numeraiTransfer` *during the migration process.*

By performing this operation, assuming all owners have been properly cleared from the `owners` and `ownerIndex` storage variables, functions protected by the `onlyOwner` and `onlyManyOwners` modifiers are no longer reachable. The affected functions have been detailed in Figure D.2.

```
onlyOwner:
    ●  claimTokens
    ●  emergencyStop

onlyManyOwners:
    ●  disableContractUpgradability
    ●  changeDelegate
    ●  changeShareable
    ●  release
    ●  disableStopping
```

Figure D.2: The functions affected by clearing the `ownerIndex` *and* `owners` *storage variables.*