Eigen Labs

EigenLayer – Checkpoint Proofs

Security Assessment Report

Version: 1.0

July, 2024

Contents

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the EigenLayer Checkpoint Proof smart contracts. The review focused solely on the security aspects of the Solidity implementation of the contract, though general recommendations and informational comments are also provided.

Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the smart contract. Sigma Prime makes no judgements on, or provides any security review, regarding the underlying business model or the individuals involved in the project.

Document Structure

The first section provides an overview of the functionality of the EigenLayer Checkpoint Proof smart contracts contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given which assigns each vulnerability a severity rating (see Vulnerability Severity Classification), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*.

Outputs of automated testing that were developed during this assessment are also included for reference (in the Appendix: Test Suite).

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within the EigenLayer Checkpoint Proof smart contracts.

Overview

EigenLayer is a restaking protocol on Ethereum, designed to enable stakers to contribute to the cryptoeconomic security of various protocols beyond Ethereum, in return for rewards.

This review focused on an incremental change to how validator balances are recorded within EigenLayer to prepare for the introduction of AVS slashing. This is done by introducing checkpoint proofs which track the change in balances in `EigenPod` and the beacon chain.

The review was completed with an additional focus in mind of forthcoming changes in Electra to ensure the resiliency of the EigenLayer protocol in the face of changing specifications.

Security Assessment Summary

Scope

The review was conducted on the files hosted on the eigenlayer-contracts repository.

The scope of this time-boxed review was strictly limited to the following files at commit d148952:

- `interfaces/IEigenPod.sol`
- `interfaces/IEigenPodManager.sol`
- `libraries/BeaconChainProofs.sol`
- `pods/*`

Note: third party libraries and dependencies, such as OpenZeppelin, were excluded from the scope of this assessment.

Approach

The manual review focused on identifying issues associated with the business logic implementation of the contracts. This includes their internal interactions, intended functionality and correct implementation with respect to the underlying functionality of the Ethereum Virtual Machine (for example, verifying correct storage/memory layout).

Additionally, the manual review process focused on identifying vulnerabilities related to known Solidity anti-patterns and attack vectors, such as re-entrancy, front-running, integer overflow/underflow and correct visibility specifiers.

For a more detailed, but non-exhaustive list of examined vectors, see [1, 2].

To support this review, the testing team also utilised the following automated testing tools:

- Mythril: `https://github.com/ConsenSys/mythril`
- Slither: `https://github.com/trailofbits/slither`
- Surya: `https://github.com/ConsenSys/surya`
- Aderyn: `https://github.com/Cyfrin/aderyn`

Output for these automated tools is available upon request.

Coverage Limitations

Due to a time-boxed nature of this review, all documented vulnerabilities reflect best effort within the allotted, limited engagement time. As such, Sigma Prime recommends to further investigate areas of the code, and any related functionality, where majority of critical and high risk vulnerabilities were identified.

Findings Summary

The testing team identified a total of 6 issues during this assessment. Categorised by their severity:

- Critical: 1 issue.
- Medium: 2 issues.
- Informational: 3 issues.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the EigenLayer smart contracts. Each vulnerability has a severity classification which is determined from the likelihood and impact of each issue by the matrix given in the Appendix: Vulnerability Severity Classification.

A number of additional properties of the contracts, including gas optimisations, are also described in this section and are labelled as “informational”.

Each vulnerability is also assigned a **status**:

- ***Open:*** the issue has not been addressed by the project team.
- ***Resolved:*** the issue was acknowledged by the project team and updates to the affected contract(s) have been made to mitigate the related risk.
- ***Closed:*** the issue was acknowledged by the project team but no further actions have been taken.

Summary of Findings

ID	Description	Severity	Status
EGN5-01	Proofs Based On `beaconStateRoot` Break After Electra Upgrade	Critical	Open
EGN5-02	Lack Of Support For Compounding Withdrawal Credentials After Electra Upgrade	Medium	Open
EGN5-03	Validator Status Incorrectly Set To `WITHDRAWN` After Electra Upgrade	Medium	Open
EGN5-04	Double-Counting ETH From Consolidations After Electra Upgrade	Informational	Open
EGN5-05	Double-Counting Partial Withdrawals After Electra Upgrade	Informational	Open
EGN5-06	Miscellaneous General Comments	Informational	Open

EGN5-01	Proofs Based On `beaconStateRoot` Break After Electra Upgrade		
Asset	`BeaconChainProofs.sol, EigenPod.sol`		
Status	**Open**		
Rating	Severity: Critical	Impact: High	Likelihood: High

Description

The Electra upgrade increases the `BeaconState` tree height, breaking proofs that rely on the `beaconStateRoot`.

In the `BeaconChainProofs` library, both validator field and balance container proofs rely on the `beaconStateRoot` as shown below:

```
/// @notice Heights of various merkle trees in the beacon chain
/// - beaconBlockRoot
/// |                                        HEIGHT: BEACON_BLOCK_HEADER_TREE_HEIGHT
/// -- beaconStateRoot
/// |                                        HEIGHT: BEACON_STATE_TREE_HEIGHT
/// validatorContainerRoot, balanceContainerRoot
/// |                      |                 HEIGHT: BALANCE_TREE_HEIGHT
/// |                      individual balances
/// |                                        HEIGHT: VALIDATOR_TREE_HEIGHT
/// individual validators
```

After Electra, the `BeaconState` SSZ container increases from 28 to 37 fields, increasing its tree height from 5 to 6 as it passes a power of 2.

Both the `verifyValidatorFields()` and `verifyBalanceContainer()` functions rely on the `BEACON_STATE_TREE_HEIGHT` constant which is set as 5 in `BeaconChainProofs`. Hence, these functions do not support proofs after Electra.

Furthermore, the increase in `BeaconState` tree height after Electra allows for a second pre-image attack on the balance container proof. This attack allows a valid proof to be constructed from the `extra_data` field inside `ExecutionPayloadHeader` against an arbitrary `balanceRoot`. Fortunately, the attack is only possible for very high `validatorIndex` values (at least $2^{39} + 2^{37}$) which is very unlikely to be reached on the beacon chain.

Recommendations

Similar to how withdrawal proofs in M2 `EigenPod` use a different execution payload header tree height after the Deneb fork timestamp, use a different beacon state root tree height after the Electra fork timestamp.

EGN5-02	Lack Of Support For Compounding Withdrawal Credentials After Electra Upgrade		
Asset	`EigenPod.sol`		
Status	**Open**		
Rating	Severity: Medium	Impact: Medium	Likelihood: Medium

Description

The `EigenPod` does not support verifying compounding withdrawal credentials, preventing ETH in validators with compounding withdrawal credentials from being counted in `EigenPod` after Electra.

EIP-7251 included in the Electra upgrade introduces the compounding withdrawal credential represented by a `0x02` prefix. This new type of withdrawal credential allows validators to have an effective balance of up to 2048 ETH.

The `_verifyWithdrawalCredentials()` function has a check to ensure that the validator's withdrawal credentials are pointed to the `EigenPod`:

```
// Ensure the validator's withdrawal credentials are pointed at this pod
require(
    validatorFields.getWithdrawalCredentials() == bytes32(_podWithdrawalCredentials()),
    "EigenPod._verifyWithdrawalCredentials: proof is not for this EigenPod"
);
```

However, `EigenPod` only supports Eth1 (`0x01` prefixed) withdrawal credentials, and not compounding (`0x02` prefixed) credentials:

```
function _podWithdrawalCredentials() internal view returns (bytes memory) {
    // @audit `bytes1(uint8(1))` corresponds to the `0x01` prefix
    return abi.encodePacked(bytes1(uint8(1)), bytes11(0), address(this));
}
```

Hence, validators with compounding withdrawal credentials cannot be verified on the `EigenPod`, preventing the ETH in compounding validators from being restaked on EigenLayer.

Note that funds can still be recovered by exiting the validator and performing a checkpoint to record the new `EigenPod` balance. However, potential AVS rewards during this downtime are lost.

Recommendations

Consider adding a new function `_podCompoundingWithdrawalCredentials()` and add support for the `0x02` prefix in `_verifyWithdrawalCredentials()`.

EGN5-03	Validator Status Incorrectly Set To `WITHDRAWN` After Electra Upgrade		
Asset	`EigenPod.sol`		
Status	**Open**		
Rating	Severity: Medium	Impact: Medium	Likelihood: Medium

Description

The Electra upgrade changes how deposits are credited to validator balances, resulting in validator status being incorrectly set to `WITHDRAWN` in a checkpoint.

Currently, deposits on the beacon chain are credited to validators at the same time that they are processed. If the deposit belongs to a new validator, they get registered onto the beacon chain along with their deposit balance.

EIP-7251 included in Electra changes how deposits are applied on the beacon chain by introducing pending balance deposits. Due to this change, processed deposits now register new validators onto the beacon chain with a zero balance and create a pending balance deposit, which gets processed once per epoch to credit the deposit amount to the validator's balance. This means that there is now a period of time (an epoch or longer due to the churn limit) when newly-registered validators will have a zero balance.

A newly-registered validator with a zero balance can have its withdrawal credentials verified on the `EigenPod`. If this validator is checkpointed, then it will be marked as `WITHDRAWN` even though the validator has not exited the beacon chain.

This issue can be prevented if the pod owner does not verify their validator on the `EigenPod` and start a checkpoint until the pending balance deposit has been processed. Furthermore, no funds are lost since the validator can exit the beacon chain to recover the ETH, which can be withdrawn from the `EigenPod` after the checkpoint. However, the pod owner will lose out on potential AVS rewards during this downtime.

Recommendations

Preventing pod owners from verifying newly-registered validators with zero balances on the `EigenPod` by checking that the validator's effective balance is greater than zero.

EGN5-04	Double-Counting ETH From Consolidations After Electra Upgrade
Asset	`EigenPod.sol`
Status	**Open**
Rating	Informational

Description

EIP-7251 consolidations, when implemented, open the `EigenPod` to a potential attack vector where ETH from a consolidated validator can be double-counted when verifying withdrawal credentials.

EIP-7251 included in the upcoming Electra upgrade allows validators to consolidate by specifying a source and target validator. The source validator's ETH balance is sent to the target validator as the source validator initiates an exit from the beacon chain.

This opens the `EigenPod` to a timing attack where a source validator is verified on the `EigenPod` before it consolidates to a target validator, which is later verified on the `EigenPod` to have both validators' balances. This results in the source validator's balance being double-counted.

Fortunately, EIP-7251 consolidations are initiated by a request from the withdrawal credentials of the source validator, hence only the `EigenPod` is capable of submitting valid consolidation requests. Hence, it is possible to take precautionary measures when implementing consolidations to protect against this attack vector.

Note, this issue is categorised as having informational severity as the in-scope `EigenPod` code currently does not implement consolidations and hence is not vulnerable to the attack described above.

Recommendations

When implementing consolidations on the `EigenPod`, ensure `ValidatorInfo::restakedBalanceGwei` accounting is properly adjusted once the consolidation has occurred. It is important to verify that the consolidation has actually occurred on the beacon chain before making the adjustment, as it is possible for consolidation requests to be rejected on the consensus layer.

Some notable reasons for consolidation requests to be rejected on the consensus layer are if:

- Either the source or target validator is not registered on the beacon chain
- The source and target validators are the same
- The withdrawal credentials of the source or target validators are not `0x01` or `0x02` prefixed
- The source or target validators are not active or are exiting the beacon chain

Note that the consensus layer **does not** require the source and target validators to have the same withdrawal address, so this must be checked on the `EigenPod`.

A more simple solution may be to use a checkpoint proof to update the balances of both the source and target validators in the same checkpoint. This requires both validators to be verified on the `EigenPod` before the consolidation request is made, such that the target validator cannot be verified on the `EigenPod` after it receives the balance from consolidation.

EGN5-05	Double-Counting Partial Withdrawals After Electra Upgrade
Asset	`EigenPod.sol`
Status	**Open**
Rating	Informational

Description

Due to EIP-7251 and EIP-7002, partial withdrawals can occur below the `MAX_EFFECTIVE_BALANCE` after the Electra upgrade. Hence it is possible for ETH from partial withdrawals to be double-counted.

A pod owner can choose the `beaconTimestamp` used to verify a validator's withdrawal credentials on the `EigenPod`. This opens the checkpoint system to a timing attack where ETH withdrawn to the EigenPod is double-counted by picking a `beaconTimestamp` before the withdrawal has occurred.

`_verifyWithdrawalCredentials()` protects against this attack by preventing exiting validators from verifying their withdrawal credentials in the check below:

```
require(
    validatorFields.getExitEpoch() == BeaconChainProofs.FAR_FUTURE_EPOCH,
    "EigenPod._verifyWithdrawalCredentials: validator must not be exiting"
);
```

This check only protects against timing attacks for full withdrawals. Currently as of the Deneb upgrade, only ETH greater than `MAX_EFFECTIVE_BALANCE` is partially withdrawn. Hence, timing attacks for partial withdrawals are not possible as this ETH does not get counted in a validator's effective balance.

EIP-7251 and EIP-7002 both included in the upcoming Electra upgrade allow a validator to have partial withdrawals that occur below `MAX_EFFECTIVE_BALANCE` through withdrawal requests submitted from the execution layer (EL). This opens the `EigenPod` to partial withdrawal timing attacks.

Fortunately, partial withdrawal requests are only valid if they're from the validator's compounding withdrawal credentials, hence only the `EigenPod` is capable of submitting valid partial withdrawal requests.

Note, this issue has an informational severity rating as the in-scope `EigenPod` code currently does not implement EL-triggerable withdrawals and hence is not vulnerable to the attack described above.

Recommendations

Consider only allowing validators verified in the `EigenPod` to submit withdrawal requests. This would prevent EL-triggered partial withdrawals below `MAX_EFFECTIVE_BALANCE` from being processed before the validator has been verified on the `EigenPod`.

Note that the recommendation above only protects against EL-triggered partial withdrawals introduced in EIP-7002. If custom ceilings for partial withdrawal sweeps were included in another future Ethereum upgrade, then the `EigenPod` will still be susceptible to partial withdrawal timing attacks.

Alternatively, consider capping the amount of ETH credited to the validator's `ValidatorInfo::restakedBalanceGwei` to 32 ETH in `_verifyWithdrawalCredentials()`, such that any ETH that is able to be partially withdrawn is deducted

from the validator's effective balance. If this recommendation is implemented, then the pod owner will then need to start and finalize a checkpoint to accurately record their validator's balance if it is above the 32 ETH cap.

EGN5-06	Miscellaneous General Comments
Asset	All contracts
Status	Open
Rating	Informational

Description

This section details miscellaneous findings discovered by the testing team that do not have direct security implications:

1. **Custom Errors**
   Related Asset(s): /*
   The codebase makes use of string errors rather than the newer custom errors feature. Since this feature has now matured, it is advised to use custom errors to reduce gas costs.
   Replace string errors with custom errors throughout the codebase.
2. **Misleading NatSpec Comment**
   Related Asset(s): pods/EigenPod.sol
   On line 124 of `EigenPod.sol` there is the NatSpec comment:
   "and any validators with 0 balance are marked `WITHDRAWN` *"*.
   This flow is only present for checkpoints that continue through `verifyCheckpointProofs()`. It is possible for a checkpoint to be completed in `startCheckpoint()` due to there being no active validators, hence this flow may not necessarily be present.
   Update the NatSpec comment to make it clear validators are only marked as `WITHDRAWN` if the function `verifyCheckpointProofs()` is called.
3. **Typos**
   Related Asset(s): interfaces/IEigenPod.sol, interfaces/IEigenManager.sol
   There are some typos present in the interface which should be corrected to improve reader clarity.
   In the IEigenPod contract:
   - On line [**74**] *"validaor"* should read *"validator"* and *"to have"* should read *"to have had"* given a checkpoint is in the past.
   - On line [**97**] *"Create"* should read *"Creates"*.
   - On line [**99**] *"marked"* should read *"marked as"*.

   In the IEigenPodManager contract:
   - On line [**89**] *"from Pod owner owner"* should read *"from Pod owner"*.

   Correct typos where noted.
4. **Floating Version Pragma**
   Related Asset(s): /*
   The contracts specify a floating Solidity pragma `^0.8.12`. This can introduce version specific bugs should any contracts be deployed with a different Solidity version.
   The pragma version should be locked at the version where testing occurred.

Recommendations

Ensure that the comments are understood and acknowledged, and consider implementing the suggestions above.

Appendix A Test Suite

A non-exhaustive list of tests were constructed to aid this security review and are given along with this document. The `forge` framework was used to perform these tests and the output is given below.

```
Ran 1 test for test/tests-local/BeaconChainProofs.t.sol:BeaconChainProofsTest
[PASS] testFuzz_getBalanceAtIndex(uint64[4]) (runs: 1004, μ: 22815, ~: 22815)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 68.56ms (68.14ms CPU time)

Ran 4 tests for test/tests-fork/BeaconRootOracle.fork.t.sol:BeaconRootOracleForkTest
[PASS] testFuzz_getParentBlockRoot(uint64) (runs: 1004, μ: 10954, ~: 11041)
[PASS] test_beaconRootOracle_CurrentSlot() (gas: 11068)
[PASS] test_beaconRootOracle_OldSlotUnderSkippedSlot() (gas: 12388)
[PASS] test_beaconRootOracle_SkippedSlot() (gas: 16810)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 2.93s (2.35s CPU time)

Ran 2 tests for test/tests-fork/EigenLayerBeaconChainOracle.fork.t.sol:EigenLayerBeaconOracleForkTest
[PASS] test_addTimestamp_AroundSkippedSlot() (gas: 84881)
[PASS] test_addTimestamp_SkippedSlot() (gas: 13706)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 4.08s (2.49s CPU time)

Ran 2 tests for test/tests-fork/EigenPod_M4Upgrade.fork.t.sol:EigenPodM4UpgradeForkTest
[PASS] test_activateRestaking_BeforeM4Upgrade() (gas: 6251032)
[PASS] test_verifyWithdrawalCredentials_BeforeM4Upgrade() (gas: 7012968)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 8.11s (2.92s CPU time)

Ran 6 tests for test/tests-fork/EigenPod_Electra.fork.t.sol:EigenPodElectraForkTest
[FAIL. Reason: revert: BeaconChainProofs.verifyBalanceContainer: Proof has incorrect length]
      ↪  test_verifyCheckpointProofs_Electra_WrongStateTreeHeight_Vuln() (gas: 1382496)
[PASS] test_verifyWithdrawalCredentials() (gas: 1270720)
[FAIL. Reason: Pod owner shares should be equal to `podBalance + validatorBalance`: 9600000000000000000 != 6400000000000000000]
      ↪  test_verifyWithdrawalCredentials_Electra_PartialWithdrawalDoubleCount_Vuln() (gas: 2133116)
[FAIL. Reason: Pod owner shares should be equal to `podBalance + validatorBalance`: 9600000000000000000 != 6400000000000000000]
      ↪  test_verifyWithdrawalCredentials_Electra_PartialWithdrawalDoubleCount_Vuln2() (gas: 1612266)
[FAIL. Reason: Validator should be active: 2 != 1] test_verifyWithdrawalCredentials_Electra_ValidatorInitialZeroBalance_Vuln()
      ↪  (gas: 1547639)
[FAIL. Reason: revert: BeaconChainProofs.verifyValidatorFields: Proof has incorrect length]
      ↪  test_verifyWithdrawalCredentials_Electra_WrongStateTreeHeight_Vuln() (gas: 1116345)
Suite result: FAILED. 1 passed; 5 failed; 0 skipped; finished in 26.13s (65.47s CPU time)

Ran 20 tests for test/tests-fork/EigenPod.fork.t.sol:EigenPodForkTest
[PASS] testFuzz_getParentBlockRoot(uint64) (runs: 1004, μ: 21157, ~: 21231)
[PASS] testFuzz_updateCheckpoint(uint256,uint256[]) (runs: 1003, μ: 6707789, ~: 6815293)
[PASS] testFuzz_withdrawRestakedBeaconChainETH(uint256) (runs: 1004, μ: 198157, ~: 198090)
[PASS] test_recoverTokens() (gas: 794366)
[PASS] test_startCheckpoint_NoValidators() (gas: 222737)
[PASS] test_startCheckpoint_RevertConditions() (gas: 1092112)
[PASS] test_verifyCheckpointProof_ExitedValidator() (gas: 1640577)
[PASS] test_verifyCheckpointProofs_RevertIf_NoCurrentCheckpoint() (gas: 904369)
[PASS] test_verifyCheckpointProofs_RevertIf_WrongBalanceContainerRoot() (gas: 984056)
[PASS] test_verifyCheckpointProofs_RevertIf_WrongBalanceRoot() (gas: 2673033)
[PASS] test_verifyCheckpointProofs_SkipsCheckpointedValidators() (gas: 2149934)
[PASS] test_verifyCheckpointProofs_SkipsNotActiveValidators() (gas: 2912124)
[PASS] test_verifyStaleBalance() (gas: 1069531)
[PASS] test_verifyStaleBalance_RevertIf_OutdatedProof() (gas: 1172348)
[PASS] test_verifyStaleBalance_RevertIf_ValidatorHasWithdrawn() (gas: 1704845)
[PASS] test_verifyStaleBalance_RevertIf_ValidatorIsInactive() (gas: 721134)
[PASS] test_verifyStaleBalance_RevertIf_ValidatorNotSlashed() (gas: 909362)
[PASS] test_verifyWithdrawalCredentials() (gas: 898801)
[PASS] test_verifyWithdrawalCredentials_RevertIf_ExitingValidator() (gas: 1220822)
[PASS] test_verifyWithdrawalCredentials_RevertIf_NotInactiveValidator() (gas: 2560375)
Suite result: ok. 20 passed; 0 failed; 0 skipped; finished in 27.48s (90.47s CPU time)

Ran 6 test suites in 27.48s (68.80s CPU time): 30 tests passed, 5 failed, 0 skipped (35 total tests)
```

Appendix B Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurance. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

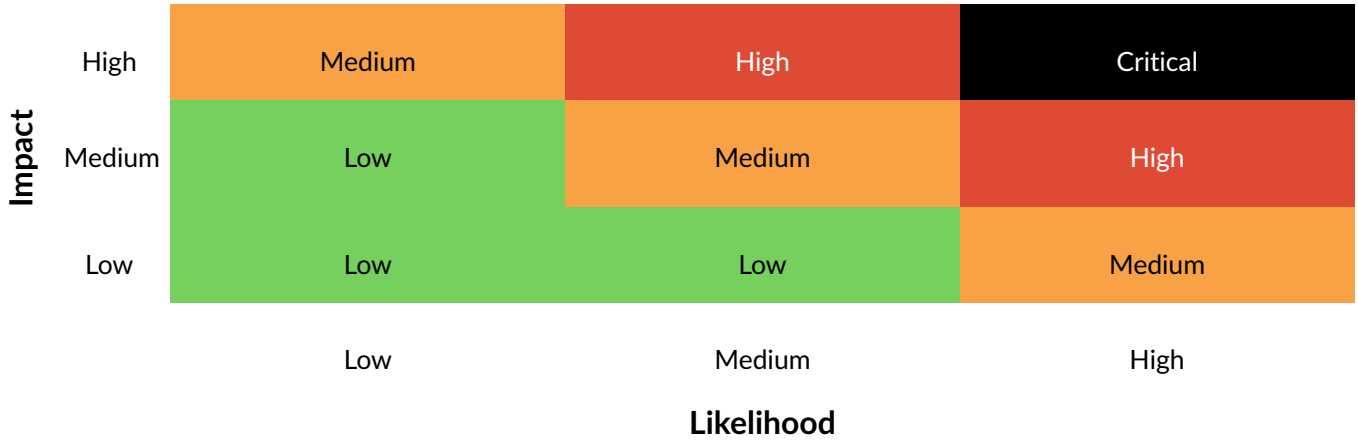

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.

References

[1] Sigma Prime. Solidity Security. Blog, 2018, Available: https://blog.sigmaprime.io/solidity-security.html. [Accessed 2018].

[2] NCC Group. DASP - Top 10. Website, 2018, Available: http://www.dasp.co/. [Accessed 2018].