Sensei Stake

Smart Contract Audit

coinspect

SenseiStake

Smart Contract Audit

V221028

Prepared for Sensei Node • September 2022

1. Executive Summary

In **September 2022, Sensei** engaged Coinspect to perform a source code review of **SenseiStake**. The objective of the project was to evaluate the security of the smart contracts. SenseiStake allows users to deposit funds to stake an Ethereum validator and receive an NFT representing the validator's ownership. In return for a percentage of the earnings of the validator, SenseiStake will run and operate it.

The following issues were identified during the initial assessment:

	High Risk	Medium Risk	Low Risk
Open	0	1	0
Fixed	0	4	1
Reported	0	6	3

Coinspect identified a total of six medium-risk issues, such as SenseiStake owners allowed to extend the time staked funds are locked by the platform, the possibility to force users to agree to a given commission rate via frontrunning, the lack of checks regarding BLS validator signatures, the possibility of funding a validator multiple times due to public key re-use, the possibility of staking to a previously funded validator, and NFT key parameters not shown in standard NFT marketplaces.

The three remaining low-risk issues are related to SenseiStake not checking `ERC-721` allowance for SenseiStake operations, race conditions on `createContract` function calls leading to unexpected results for users, and SenseiStake potentially losing the staking commission under certain conditions.

2. Assessment and Scope

The audit started on **September 26, 2022** and was conducted on the **master** branch of the git repository at https://github.com/Sensei-Node/SenseiStake as of commit **`1661cddc1b484a900c884b0ee93b837e7348fb89`** of **September 23, 2022**.

As described in the executive summary, `SenseiStake` allows users to deposit `32 ETH` to stake an Ethereum validator. When users stake funds, the platform mints an `NFT` representing the ownership of the validator, which is later burned when funds are withdrawn. After unstaked funds are sent to the `SenseiStake` contract, funds resulting from staking profits are subject to a commission rate. The commission rate can vary from 0% to 50% and is settled when the user deposits the funds. On the other hand, if unstaked funds result in less than `32 ETH` (due to slashing for instance), the platform does not take a commission.

The audited files have the following sha256sum hash:

```
2c88409b1bb127bcd234c3fd0e310d0600dabaa88c8facb55bd43edcb0e8b7e6  SenseiStake.sol
30b7632e4e6dcf73ab6f24044c9a3294201370e6c0e3abe77e51012bb3358846  SenseistakeServicesContract.sol
```

Contracts declare a fixed pragma version at 0.8.4 and are compiled with the same release. It is recommended both to update to a newer version as many bug fixes have been introduced and use non fixed pragma versions instead.

The code was clean, easy to understand, and very well documented with inline comments. The repository includes a test suite providing high code coverage. However, some tests lack assertions, or contain minor errors such as swapped descriptions.

The platform presents a noticeable centralization degree, which is mainly due to the lack of clarity around unstaking an Ethereum validator and the dates it would be allowed, which is not under SenseiStake's control. For instance, SenseiStake owners can indefinitely push the `endDate` of the users' deposit, can modify the commission rate before users deposit funds, and generate the keys to operate the validator. SenseiStake is also in charge of operating the validators, although it is out of the scope of this audit. SenseiStake keeps the custody of the validator's credentials, which are used to operate the validator -note the user does not know these credentials at any time. Since SenseiStake operates the validator, they are responsible for avoiding penalties or slashing, as well as for signaling the validator's exit in the proof-of-stake chain. Regarding funds' withdrawal, the code specifies the address of the contract associated with the NFT as the `withdrawal_credentials` for the validator. The `withdrawal_credentials` let users set the keys allowed to withdraw funds unstaked from the validator, or the address where withdrawn funds will be sent to. SenseiStake employs the latter mechanism.

The contracts could use extra validation around user-provided inputs. Coinspect detected several opportunities for improvement apart from the issues reported:

- Check that no duplicate validator is inserted
- Check that `exitDate` is not exceeded when funding a validator in `createContract`
- Consider limiting how much the `endDate` can be increased each time. A single mistake might push this date too far away in the future and lock funds for ever
- Notice that if the validator does not unstake before the `exitDate` is reached, SenseiStake could lose the service commission. This is an important consideration when creating the `SenseiStake` validators.

The **withdrawal of staked funds is still undefined in the Ethereum network**. It is likely, but uncertain, that the unstaking will trigger an increase in the balance of the withdrawal address, based on the EIP-4895. This would be compatible with the current implementation, but the uncertainty of the process imposes an avoidable risk. An alternative approach would be to allow the platform to use a new implementation of the SenseistakeServicesContract contract if both the token owner and Sensei owner agree. This way, if an incompatibility arises during an Ethereum hard fork, funds are not irreversibly lost.

Additional fix review comments

- SenseiStake added a mechanism to send arbitrary transactions from the `SenseistakeServicesContract`, which could be used for the funds withdrawal process, yet unconfirmed. These transactions require the depositor's consent. Note this newly added functionality was not included in the scope of this audit.

3. Summary of Findings

Id	Title	Total Risk	Fixed
SNS-01	Platform owner can extend exitDate for an indefinite period	Medium	✔
SNS-02	Platform owner can force users into accepting a higher commission rate	Medium	✔
SNS-03	Invalid BLS signature can lead to funds loss	Medium	!
SNS-04	Users' funds can be lost after re-funding validator	Medium	✔
SNS-05	Deposit to pre funded validator will cause funds loss	Medium	✘
SNS-06	NFT key parameters not displayed on marketplaces	Medium	✔
SNS-07	Check ERC721 allowance instead of ownership	Low	✔
SNS-08	Users can fund an unexpected validator with different conditions	Low	!
SNS-09	Hardcoded value (32 ETH)	Info	✔
SNS-10	404 on tokenURI link	Info	✔
SNS-11	SenseiStake can lose staking commission	Low	!

4. Detailed Findings

SNS-01 Platform owner can extend exitDate for an indefinite period

Total Risk Medium	Impact High	Location `SenseistakeServicesContract.sol`
Fixed ✔	Likelihood Low	

Description

The platform owner can tempt users to acquire a validator with appealing arbitrage conditions. However, the owner can then indefinitely modify the contracts' `exitDate` (when investors can start the withdrawal process) to maximize their commission gains.

As an example, the platform owner could fund a validator with their own funds and then trade the resulting `NFT`. If the `token` trade provides appealing conditions such as a short `exitDate` and a lower purchase price, investors willing to arbitrage may be tempted to purchase it. However, the platform owner can deliberately extend the `exitDate` once it is purchased to recover the expenses and profit of the deposited capital via commissions.

The `updateExitDate` function below from the `SenseistakeServicesContract` contract allows the Sensei platform owner to freely extend the `exitDate` of a deposit for an indefinite amount of time.

```
/// @notice For updating the exitDate
/// @dev The exit date must be after the current exit date and it's only possible in validatorActive ==
true
/// @param exitDate_ The new exit date
function updateExitDate(uint64 exitDate_) external onlyOperator {
   if (!validatorActive) {
       revert ValidatorNotActive();
   }
   if (exitDate_ < exitDate) {
       revert NotEarlierThanOriginalDate();
   }
   exitDate = exitDate_;
   emit ExitDateUpdated(exitDate_);
}
```

Recommendation

Ideally, the token owner could prevent an `exitDate` update, but not require the agreement for an update. I.e. the `exitDate` is updated unless the token owner disagrees. It is in the best interest of the token owner to not disagree before the possibility to unstake a validator, as the validator can still misbehave and burn the user funds.

On the other hand, once unstaking is possible, it is in the best interest of the validator to exit on time to receive the commission for the service.

Status

Fixed. The functionality to modify the `exitDate` indefinitely was removed. The initial `exitDate` is set to 180 after the NFT is minted and the validator created.

SNS-02 Platform owner can force users into accepting a higher commission rate

Total Risk	Impact	Location
Medium	High	`SenseiStake.sol`
Fixed ✔	Likelihood Low	

Description

The platform owner can front-run users to force them to accept a higher commission rate than expected when calling the `createContract` function.

When users call the `createContract` function to stake a new validator, the current `commissionRate` set in the contract is used to lock the commission charged by Sensei.

```
/// @notice Creates service contract based on implementation
/// @dev Performs a clone of the implementation contract, a service contract handles logic for managing
user deposit, withdraw and validator
function createContract() external payable {
   if (msg.value != FULL_DEPOSIT_SIZE) {
       revert ValueSentDifferentThanFullDeposit();
   }
   // increment tokenid counter
   tokenIdCounter.increment();
   uint256 tokenId = tokenIdCounter.current();
   Validator memory validator = _validators[tokenId];
   // check that validator exists
   if (validator.validatorPubKey.length == 0) {
       revert NoMoreValidatorsLoaded();
   }
   bytes memory initData = abi.encodeWithSignature(
       "initialize(uint32,uint256,uint64,bytes,bytes,bytes32)",
       commissionRate,
       tokenId,
       validator.exitDate,
       validator.validatorPubKey,
       validator.depositSignature,
       validator.depositDataRoot
   );
   address proxy = Clones.cloneDeterministic(
       servicesContractImpl,
       bytes32(tokenId)
   );
   (bool success, ) = proxy.call{value: msg.value}(initData);
   require(success, "Proxy init failed");

   emit ContractCreated(tokenId);

   // mint the NFT
   _safeMint(msg.sender, tokenId);
}
```

However, the platform owner can front-run the `createContract` user's transaction and change the commission rate that will then be used by the `createContract` function.

```
/// @notice Changes commission rate (senseistake service fees)
/// @dev Cannot be more than 50% commission
/// @param commissionRate_ New commission rate
function changeCommissionRate(uint32 commissionRate_) external onlyOwner {
   if (commissionRate_ > (COMMISSION_RATE_SCALE / 2)) {
       revert CommisionRateTooHigh(commissionRate_);
   }
   commissionRate = commissionRate_;
   emit CommissionRateChanged(commissionRate_);
}
```

Once users stake a validator, the deposit cannot be undone and the `commissionRate` is fixed, and the only way to recover the deposited funds is to wait until the operator service contract is finished.

Recommendation

Allow the user to provide an accepted maximum commission rate when calling the `createContract` function. Consider also allowing the user to provide an accepted `endDate.`

Status

Fixed. Owners cannot modify the commission rate in the reviewed version with commit **8f97d4dfe3ca3cfbe52b1d282d4ee15e8f7f3c95**.

SNS-03 Invalid BLS signature can lead to funds loss

Total Risk	Impact	Location
Medium	High	`SenseiStake.sol`
Fixed !	Likelihood Medium	

Description

The `BLS` signature is not validated before adding a new validator. A malformed `BLS` signature will cause all funds to be lost forever for historical reasons.

Currently the Beacon Chain has the following key steps regarding the deposit process: **unknown** (deposit on the mempool), **deposited** (the input data is validated within ~12hs), **pending** (stake accessible within the beacon-chain but only 6 validators per epoch can be activated, so it is enqueued), active (the validator is actively staking) and **exited**.

If the BLS signature provided while performing a deposit is invalid but has a valid length it won't revert while making the deposit and will fail during the validation process losing the stake.

Recommendation

Users should validate the BLS signature before calling the createContract function.

Status

Partially fixed in commit **c1d95ce1a70b59d7a75d58c5a98b2269ff759f1c.** The platform now verifies the signature off-chain before adding a new validator. An additional solution would be allowing depositors to verify the BLS in the UI right before depositing funds. Note this solution would require that the next validator to be funded cannot be modified by SenseiStake to prevent front-running issues,

SNS-04 Users’ funds can be lost after re-funding validator

Total Risk	Impact	Location
Medium	High	`SenseiStake.sol`
Fixed ✓	Likelihood Low	

Description

Users' funds can be lost if the same validator public keys are used for different validators.

When the validator is added for the first time, the Beacon Chain `process_deposit` function performs the signature and credential checks. For subsequent calls made with the same public key, those checks are omitted and the stake position is only topped up.

The `addValidator` function does not check if the validator has been previously assigned to a `tokenId`. If duplicate validators are used, multiple stakers will have an `NFT` backed by the same validator and staking position. In the case of such an event, the second user will be topping the first user position, causing the user to lose all funds.

Recommendation

Save a mapping of used public keys.

Status

Fixed in commit **eb19d2ca39d25f4523ffb55f67dbef1f3f727959.** SenseiStake validators cannot be reused now.

SNS-05 Deposit to pre funded validator will cause funds loss

Total Risk	Impact	Location
Medium	High	`SenseiStake.sol`
Fixed	**Likelihood**	
✘	Low	

Description

Users could deposit funds to a validator that has been previously staked, not necessarily staked via the `SenseiStake` contract. This prevents users from getting a brand new validator and puts their funds at risk in the withdrawal process.

Currently there are several scenarios where users could stake to a validator that has previously been sent to the `ETH2 Deposit` contract. The attack starts with the owner staking just 1 ether to a new validator via the `ETH2 Deposit` contract. A benign user then calls `createContract` function sending 32 `ETH` for staking. Before their transaction is mined, the owner could front-run that transaction by calling the `addValidator` function to associate the next `tokenId` to be minted to the partially funded validator.

Recommendation

Consider selling the NFT with the validator already funded. This would require SenseiStake to manipulate a hot wallet when creating validators. The simplest implementation would be that the `addValidator` function should require 1 `ETH` per validator and make the deposit, proxy creation and initialization in that same call. Then, anybody can purchase the NFT for 32 `ETH`, which should trigger a refund of the pre-deposited funds.

Users buying the NFT should wait for block finalization and check the BLS signature off chain before purchasing the token.

Status

Not fixed. SenseiStake will consider offering pre-funded validators in the future.

SNS-06 NFT key parameters not displayed on marketplaces

Total Risk	Impact	Location
Medium	Medium	`SenseiStake.sol`
Fixed ✓	Likelihood: Medium	

Description

The `tokenURI` function does not include key parameters of the backed validator, such as the `commissionRate` and `exitDate`. As a consequence, users could be deceived when purchasing these NFTs on standard marketplaces, unaware of SenseiStake operations

Non-fungible token marketplaces (for example, Opensea) require a specific return structure for the `tokenURI` function to ensure it will be correctly displayed.

A validator-backed NFT could be adopted by the NFT community that is well aware of how to trade these types of tokens but could be unaware on how to set up a validator themselves.

It is unknown whether NFT trading will be supported by a custom-made SenseiStake UI, or it is meant to be supported by existing NFT marketplaces such as OpenSea. The lack of these parameters within the `tokenURI` could be abused to take advantage of the secondary market. For example:

- Alice mints directly a validator through the Sensei platform with a high commission rate (e.g. 50%)
- At some point, SenseiStake lowers the `commissionRate` to 10%
- Alice posts her NFT on a marketplace and Bob purchases it
- Alice instantiates a new validator with a lower commission rate

Recommendation

Add the validator `commissionRate` and `exitDate` NFT values to the `tokenURI` function output.

Status

Fixed in commit **`eb19d2ca39d25f4523ffb55f67dbef1f3f727959`**. The `TokenURI` function now shows the missing data.

SNS-07 Check ERC721 allowance instead of ownership

Total Risk **Low**	Impact Low	Location SenseiStake.sol
Fixed ✓	Likelihood Low	

Description

The `endOperationService` and `withdraw` functions require the caller to be the owner, which is an uncommon practice. Owners can approve other addresses to operate with the token which is allowed in the Open Zeppelin `ERC721` token contract implementation used. These transfer operations allowed by the owners are on a higher hierarchy than the `endOperationService` and `withdraw` functions.

Recommendation

Any allowed address should also be allowed to call the `endOperationService` and `withdraw` functions. Use the `_isApprovedOrOwner` function instead of checking for ownership.

Status

Fixed in commit eb19d2ca39d25f4523ffb55f67dbef1f3f727959. Checks for ownership were replaced by ownership or allowance.

SNS-08 Users can fund an unexpected validator with different conditions

Total Risk Low	Impact Low	Location `SenseiStake.sol`
Fixed !	Likelihood Low	

Description

When a user calls `createContract` there is an expected `exitDate` and validator to be used. If two users call this method at the same time, one of those users will get an unexpected validator and `exitDate`.

Recommendation

The `createContract` function can receive optional parameters to limit which validator or `exitDate` is accepted and fail if there is a mismatch.

Status

Partially fixed. Since SenseiStake will use a fixed commission rate and exitDate, the only variation the user could suffer is funding a different validator than expected.

SNS-09 Hardcoded value (32 ETH)

Total Risk	Impact	Location
Info	-	`SenseistakeServicesContract.sol`
Fixed ✓	Likelihood -	

Description

Hardcoded constants used across the codebase for comparisons and checks are sometimes difficult to understand without having a context or reading the documentation. Similarly, maintainability of the code is hindered by hardcoded numbers around the code. In order to provide a better understanding of each constant value used, they could be modified by `constant` variables.

Also, if the numbers are not queried by other contracts those variables could be set as `private` instead of `public.` Restricting the visibility removes the need for the compiler to create a getter for each public variable, saving gas. Users will still be able to query those values by reading the contract code, which should be verified after deployment.

```
if (balance > 32 ether) {
   unchecked {
       uint256 profit = balance - 32 ether;
```

Recommendation

Use the `FULL_DEPOSIT_SIZE` constant instead of hardcoded values.

Status

Fixed in commit 7557a28975b6ce7f749ab6ffecc64ea79cee500e. Constants are used now

SNS-10 404 on tokenURI link

Total Risk	Impact	Location
Info	-	SenseiStake.sol
Fixed ✔	Likelihood -	

Description

The link https://dashboard.senseinode.com/non-custodial/stake/eth provided in the `tokenURI` returns a `404` error when opening.

Recommendation

Fix the link or service at the given location.

Status

Fixed. SenseiStake provided a new working link https://dashboard.senseinode.com/senseistake.

SNS-11 SenseiStake can lose staking commission

Total Risk	Impact	Location
Low	Low	`SenseistakeServicesContract.sol`
Fixed !	Likelihood Low	

Description

By calling the `endOperatorServices` function before the staked ETH is withdrawn to the contract, a malicious depositor can force the SenseiStake commission to 0.

Depositors can force the `operatorClaimable` variable (the SenseiStake commission) to 0 once the `exitDate` (180 days) elapses and the contract balance is less than 32ETH. This would require the user to deposit at least 16 `ETH` as highlighted below, which can be later recovered.

```
/// @notice Allows user to start the withdrawal process
/// @dev After a withdrawal is made in the validator, the receiving address is set to this contract address, so there will be funds available in here. This function needs to be called for being able to withdraw current balance
function endOperatorServices() external {
   uint256 balance = address(this).balance;
   if (balance < 16 ether) {
       revert CannotEndZeroBalance();
   }
   if (!validatorActive) {
       revert NotAllowedInCurrentState();
   }
   if (block.timestamp < exitDate) {
       revert NotAllowedAtCurrentTime();
   }
   if (
       (msg.sender != tokenContractAddress) &&
       (
           !SenseiStake(tokenContractAddress).isApprovedOrOwner(
               msg.sender,
               tokenId
           )
       ) &&
       (msg.sender != Ownable(tokenContractAddress).owner())
   ) {
       revert CallerNotAllowed();
   }
   validatorActive = false;
   if (balance > FULL_DEPOSIT_SIZE) {
       unchecked {
           uint256 profit = balance - FULL_DEPOSIT_SIZE;
           uint256 finalCommission = (profit * commissionRate) /
               COMMISSION_RATE_SCALE;
           operatorClaimable += finalCommission;
       }
   }
```

```
    emit ServiceEnd();
}
```

Although SenseiStake added a check to require at least 16 ETH to execute this function, this does not prevent users from exploiting this issue. Depositors can exploit it in the time window between the validator exit signal and the ETH withdrawal to avoid risking any capital. On the other hand, this same check would require the user to top-up the contract's balance if withdrawn funds result in less than 16 ETH due to slashing or penalization.

Recommendation

Consider leaving the `updateExitDate` function following the recommendation in SNS-01.

Status

Acknowledged. SenseiStake assumes the risk of potentially losing the staking commission for a given validator.

5. Disclaimer

The information presented in this document is provided "as is" and without warranty. The present security audit does not cover any off-chain systems or frontends that communicate with the contracts, nor the general operational security of the organization that developed the code.