Project: Sushiswap's BentoBox Strategies
Commit hash: be407e27cd1a9ca8060ef9a389d1cf01b4e5540e
Contracts in scope:
- **contracts/BaseStrategy.sol**
- **contracts/AaveStrategy.sol**

Reviewer: balag3 (carbalage@gmail.com)

Findings

1. ***skim* missing access control modifier.**
   1.1. **Summary**: *BaseStrategy's skim* function missing any access control modifier resulting that it can be executed by anyone.
   1.2. **Details**: *BaseStrategy's skim* function executes the overriden *_skim* function which in the context of AaveStrategy child contract means that the strategy deposits the *amount* specified as the function parameter into the *AaveLendingPool.* In this scenario the missing modifier permits anyone to deposit into the lending pool if the *amount* is present on the contract's strategy token balance.
   1.3. **Github Permalinks**:
   BaseStrategy:
   https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L141-L144
   AaveStrategy:
   https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/strategies/AaveStrategy.sol#L73-L75
   1.4. **Mitigation**: Consider adding the *onlyBentoBox* modifier to the *skim* function.
   1.5. **Tools/Techniques**: Manual review.
   1.6. **Difficulty+Impact**: Difficulity: Low, Impact: Medium
   1.7. **Checklist Numbers**: 4. Incorrect access control, 150. Missing modifiers

2. **afterExit function's low level call doesn't check account existence.**
   2.1. **Summary**: *BaseStrategy's afterExit* function executes a low level *call* to address *to*. If *to* is non existent it will still return true.
   2.2. **Details**: Low-level calls call/delegatecall/staticcall return true even if the account called is non-existent (per EVM design). Account existence must be checked prior to calling if needed.
   Exploit scenario: After a given strategy exited the owner executes the function *afterExit* to remove/rescue any funds remaining in the contract. By mistake s/he passes a zero/non-existent account as a *to* parameter.
   The function returns with true making believe the owner that the transaction was successful.
   2.3. **Github Permalinks**:
   https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L254-L263

2.4. **Mitigation**: Consider adding zero address check and account existence check to *afterExit's to* parameter.
2.5. **Tools/Techniques**: Manual review.
2.6. **Difficulty+Impact**: Difficulty: Low, Impact: Medium
2.7. **Checklist Numbers**: 38. Account existence check for low-level calls, 49. Missing zero address validation.

3. **afterExit function's low level call doesn't check return value.**
3.1. **Summary**: *BaseStrategy's afterExit* function executes a low level *call* to address *to*. The return value is not checked if it was successful or not.
3.2. **Details**: If the return value of a low-level message call is not checked then the execution will resume even if the called contract throws an exception. If the call fails accidentally or an attacker forces the call to fail, then this may cause unexpected behavior in the subsequent program logic.
3.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L262-L263
3.4. **Mitigation**: Consider checking the return value in the *afterExit* function to avoid unexpected failures.
3.5. **Tools/Techniques**: Manual review.
3.6. **Difficulty+Impact**: Difficulty: Low, Impact: Medium
3.7. **Checklist Numbers**: 37. Return values of low-level calls

4. **afterExit function does not use Openzeppelin's Address library**
4.1. **Summary**: *BaseStrategy's afterExit* function deals with a low level *call* where mandatory checks are missing.
4.2. **Details**: OpenZeppelin maintains a library of standard, audited, community-reviewed, and battle-tested smart contracts. *Address* library has everything to achieve *afterExit's* functionality in a secure way.
4.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L262-L263
4.4. **Mitigation**: Consider using the functionCallWithValue from *Address* library Which incorporates contract existence check and call return value check.
4.5. **Tools/Techniques**: Manual review.
4.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
4.7. **Checklist Numbers**: 37. Return values of low-level calls

5. ***strategyToken* missing zero address validation.**
5.1. **Summary**: *BaseStrategy* constructor's *strategyToken* address missing zero address validation.
5.2. **Details**: *strategyToken* address defines the strategy's underlying ERC20 token used for investing/divesting by *Bentobox*. A valid token address is crucial for the strategy to function.
5.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L68-L70

5.4. **Mitigation**: Consider adding zero address validation to *BaseStrategy* constructor's *strategyToken* address parameter.
5.5. **Tools/Techniques**: Manual review.
5.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
5.7. **Checklist Numbers**: 49. Missing zero address validation.

6. ***bentoBox* missing zero address check.**
6.1. **Summary**: *BaseStrategy* constructor's *bentoBox* address missing zero address check.
6.2. **Details**: *bentoBox* address defines the main entity governing the strategy's invest/divest logic, provides liquidity and registers profits/losses. A valid *bentoBox* address is crucial for the strategy to function.
6.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L68-L71
6.4. **Mitigation**: Consider adding zero address check to *BaseStrategy* constructor's *bentoBox* address parameter.
6.5. **Tools/Techniques**: Manual review.
6.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
6.7. **Checklist Numbers**: 49. Missing zero address validation.

7. ***allowedSwapPath[] accepts* zero address.**
7.1. **Summary**: *BaseStrategy* constructor's *allowedSwapPath* address array *accepts* zero addresses.
7.2. **Details**: *allowedSwapPath* address array defines the paths that the contract can use to swap reward token to strategy token via SushiSwap.If the path contains *one or more zero address the swap will fail.*
7.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L74-L77
7.4. **Mitigation**: Consider checking that *BaseStrategy* constructor's *allowedSwapPath* address array contains any zero addresses.
7.5. **Tools/Techniques**: Manual review.
7.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
7.7. **Checklist Numbers**: 49. Missing zero address validation.

8. ***_aaveLendingPool* missing zero address validation.**
8.1. **Summary**: *AaveStrategy* constructor's *_aaveLendingPool* address missing zero address validation.
8.2. **Details**: *_aaveLendingPool* address defines the Aave Lending Pool's address where the strategy invests to and divests from to gain profit. A valid *_aaveLendingPool* address is crucial for the strategy to function.
8.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/strategies/AaveStrategy.sol#L62-L67

8.4. **Mitigation**: Consider adding zero address check to *AaveStrategy* constructor's *_aaveLendingPool* address parameter.
8.5. **Tools/Techniques**: Manual review.
8.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
8.7. **Checklist Numbers**: 49. Missing zero address validation.

9. ***_incentiveController* missing zero address validation.**
9.1. **Summary**: *AaveStrategy* constructor's *_incentiveController* address missing zero address validation.
9.2. **Details**: *_incentiveController* address is used to claim the rewards for staked strategy tokens in the Aave protocol. In case of a zero *_incentiveController* address the contract won't be able to claim any rewards for staking and prevents the strategy from functioning as intended.
9.3. **Github Permalinks**:
https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/strategies/AaveStrategy.sol#L62-L68
9.4. **Mitigation**: Consider adding zero address validation to *AaveStrategy* constructor's *_incentiveController* address parameter.
9.5. **Tools/Techniques**: Manual review.
9.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
9.7. **Checklist Numbers**: 49. Missing zero address validation.

10. ***Insufficient testing***
10.1. **Summary**: The contracts under review lack appropriate testing, very few functionalities are covered with tests and some test cases are failing. This increases the likelihood of errors in the development process and makes the code more difficult to review.
10.2. **Mitigation**: Consider adding unit tests to cover all public functions at least once, as well as all known corner cases.
Consider integrating coverage analysis tools into the development process and regularly review the coverage.
10.3. **Tools/Techniques**: Manual review.
10.4. **Difficulty+Impact**: Difficulty: Low, Impact: Medium
10.5. **Checklist Numbers**: 155. Tests.

11. ***Uncalled public functions*.**
11.1. **Summary**: *BaseStrategy*'s *swapExactTokensForUnderlying* and *afterExit* functions are declared public but never called from within the contract (nor from the the child contract)
11.2. **Github Permalinks**:
swapExactTokensForUnderlying:
11.3. https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L283
afterExit:

https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L254-L258

11.4. **Mitigation**: Consider declaring the above mentioned functions as external to save gas.
11.5. **Tools/Techniques**: Manual review.
11.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
11.7. **Checklist Numbers**: 72. Uncalled public functions.

12. ***Document token behavior restrictions.***
12.1. **Summary**: *On the level of strategies there is no token restriction beside ERC20.*
12.2. **Details:** As with any protocol that interacts with arbitrary ERC20 tokens, it is important to clearly document which tokens are supported.
Often this is best done by providing a specification for the behavior of the expected ERC20 tokens and only relaxing this specification after careful review of a particular class of tokens and their interactions with the protocol.
Understand that development team controls the strategy contracts as well as noted many times in the BentoBoxV1 contract, but restricting the list of usable ERC20 tokens in *BaseStrategy* contract would reduce
the risk of deploying a malicious derived strategy contract with a vulnerable ERC20 token. Beside risk reduction this could enhance user trust and engagement.
12.3. **Mitigation**: Known deviations from “normal” ERC20 behavior should be explicitly noted as NOT supported by the protocol
12.4. **Tools/Techniques**: Manual review.
12.5. **Difficulty+Impact**: Difficulty: Low, Impact: Low
12.6. **Checklist Numbers**: 102 107, 108, 111, 112, 113, 114, 115, 116, 117, 119, 121, 129.

13. ***System specifications***
13.1. **Summary**: Current specification is limited to a short github readme file.
Mitigation: Recommendation: Consider writing a full specification. Ensure that the specification of the entire system is considered, written and evaluated to the greatest detail possible. Specification describes how (and why) the different components of the system behave to achieve the design requirements. Without specification, a system implementation cannot be evaluated against the requirements for correctness.
13.2. **Tools/Techniques**: Manual review.
13.3. **Difficulty+Impact**: Difficulty: Low, Impact: Low
13.4. **Checklist Numbers**: 136 System specifications.

14. ***System documentation*.**
14.1. **Summary**: Current documentation is limited to inline natspec comments.
14.2. **Mitigation**: Ensure that roles, functionalities and interactions of the entire system are well documented to the greatest detail possible. Documentation describes what (and how) the implementation of different components of the system does to achieve the specification goals. Without documentation, a

system implementation cannot be evaluated against the specification for correctness and one will have to rely on analyzing the implementation itself.

14.3. **Tools/Techniques**: Manual review.
14.4. **Difficulty+Impact**: Difficulty: Low, Impact: Low
14.5. **Checklist Numbers**: 137. System documentation.

15. ***executor* missing zero address validation.**
    15.1. **Summary**: *BaseStrategy setStrategyExecutor* function's *executor* address missing zero address validation.
    15.2. **Details**: *setStrategyExecutor* function controls the *strategyExecutors* mapping that defines the addresses permitted to execute functions marked with the *onlyExecutor* modifier.
    Adding zero address to this mapping is redundant and serves no purpose.
    15.3. **Github Permalinks**:
    https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L136-L139
    15.4. **Mitigation**: Consider adding zero address validation to *BaseStrategy setStrategyExecutor* function's *executor* address parameter.
    15.5. **Tools/Techniques**: Manual review.
    15.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
    15.7. **Checklist Numbers**: 138. Function parameters.

16. ***skim* missing amount validation.**
    16.1. **Summary**: *BaseStrategy's skim* function can be called with zero amount or amount that is greater than the contract's balance.
    16.2. **Details**: *skim* function is responsible to invest the *amount* according to the underlying strategy. If the amount parameter is zero or greater than the contract's balance we either force the implementing strategy to validate the *amount* which can be missed just like in case of the AaveStrategy contract where this check is delegated to the staking pool. Not validating the amount results in one or more redundant (external) calls.
    16.3. **Github Permalinks**:
    https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L142-L144
    16.4. **Mitigation**: Consider adding a validation to *BaseStrategy skim* function's *amount* parameter.
    16.5. **Tools/Techniques**: Manual review.
    16.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
    16.7. **Checklist Numbers**: 138. Function parameters.

17. ***pathindex* missing validation.**
    17.1. **Summary**: *BaseStrategy getAllowedPath* function's *pathIndex* parameter is not validated to be an existing index of the *_allowedSwapPaths* array.
    17.2. **Details**: *getAllowedPath function* uses the *pathIndex* to access *_allowedSwapPaths two dimensional* array that contains multiple paths that the contract can use to swap reward token to strategy token.

Without validation if the *pathIndex* is not an existing index the function will produce a Panic.

17.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L265-L267

17.4. **Mitigation**: Consider validating that *pathIndex* is a valid index of the accessed array.

17.5. **Tools/Techniques**: Manual review.

17.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low

17.7. **Checklist Numbers**: 138. Function parameters.

18. ***path* missing validation.**

18.1. **Summary**: *BaseStrategy setAllowedPath's path* parameter is not validated to be an array that contains any element.

18.2. **Details**: *setAllowedPath* adds the *path* parameter to the *_allowedSwapPaths* two dimensional array. Swapping by SushiSwap needs a *path* at least with two elements. Without validation an empty array can be passed accidentally as the *path* and emitted as *allowed* by the *LogSetAllowedPath* event. This would produce an invalid path and might produce further errors during a possible swap operation.

18.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L269-L272

18.4. **Mitigation**: Consider adding validation that the *path* parameter contains at least two elements.

18.5. **Tools/Techniques**: Manual review.

18.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low

18.7. **Checklist Numbers**: 138. Function parameters.

19. ***path* missing zero address validation.**

19.1. **Summary**: *BaseStrategy setAllowedPath's path* parameter elements missing zero address validation.

19.2. **Details**: *setAllowedPath* adds the *path* parameter to the *_allowedSwapPaths* two dimensional array. Without checking that none of the *path's* elements is a zeroth address one might accidentally add an array that contains one or more zero addresses. This would produce an invalid path and might produce further errors during a possible swap operation.

19.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L269-L272

19.4. **Mitigation**: Consider adding validation that the *path* array does not contain any zero addresses.

19.5. **Tools/Techniques**: Manual review.

19.6. **Difficulty+Impact**: Difficulty: Low, Impact: Medium

19.7. **Checklist Numbers**: 138. Function parameters, 49. Missing zero address validation.

20. ***pathindex* missing validation.**

20.1. **Summary**: *BaseStrategy swapExactTokensForUnderlying* function's *pathIndex* parameter is not validated to be an existing index of the *_allowedSwapPaths* array.

20.2. **Details**: *swapExactTokensForUnderlying function* uses the *pathIndex* to access *_allowedSwapPaths two dimensional* array that contains multiple paths that the contract can use to swap reward token to strategy token. Without validation if the *pathIndex* is not an existing index the function will produce a Panic.

20.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L283-L289

20.4. **Mitigation**: Consider validating that *pathIndex* is a valid index of the accessed array.

20.5. **Tools/Techniques**: Manual review.

20.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low

20.7. **Checklist Numbers**: 138. Function parameters.

21. ***path's indexed access missing validation***

21.1. **Summary**: *BaseStrategy swapExactTokensForUnderlying* function accesses an array (that might be empty or can contain fewer elements than expected) by index without validation.

21.2. **Details:** *BaseStrategy swapExactTokensForUnderlying* function's retrieves an array of addresses from *_allowedSwapPaths* two dimensional array and assigns it to the variable named *path* and accesses it's first and second element by index. Also it is passed to UniswapV2Library's getAmountsOut function which accepts only arrays with a minimum of two elements as a valid path. Since the path variable itself can be empty or containing just one element a Panic or Error might be produced depending on the array's size.

21.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L291-L301

21.4. **Mitigation**: Consider adding a validation that the used index is existing and the path contains at least two addresses or ensure that *_allowedSwapPaths* can't contain arrays shorter than two elements as mentioned in an earlier finding.
Tools/Techniques: Manual review.

21.5. **Difficulty+Impact**: Difficulty: Low, Impact: Low

21.6. **Checklist Numbers**: 138. Function parameters.

22. ***_allowedSwapPaths accepts arbitrary length of path arrays***

22.1. **Summary**: *BaseStrategy _allowedSwapPaths* accepts arbitrary length of arrays and gets iterated through.

22.2. **Details**: A path can be added to *_allowedSwapPaths either in the constructor or via the setAllowedPath* function. The path address array can contain arbitrary number of address which gets iterated through 2 times, once in the *swapExactTokensForUnderlying* function in line 293 via *UniswapV2Library.getAmountsOut* function and after in the *_swap* function in

line 312 where even an external contract call is made. Given that the path array's size is large enough this can lead to Out Of Gas exceptions.

22.3. **Github Permalinks**:
https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L293

https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L312-L318

22.4. **Mitigation**: Consider limiting and checking the maximum element that a single path can contain.

22.5. **Tools/Techniques**: Manual review.

22.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low

22.7. **Checklist Numbers**: 138. Function parameters.

23. ***Access control specification***

23.1. **Details**: Ensure that the various system actors, their access control privileges and trust assumptions are accurately specified in great detail so that they are correctly implemented and enforced across different contracts, functions and system transitions/flows.

23.2. **Mitigation**: Consider documenting the system actors (owner, executors) and their privileges, whether they are controlled by an EOA or a multisig address.

23.3. **Tools/Techniques**: Manual review.

23.4. **Difficulty+Impact**: Difficulty: Low, Impact: Low

23.5. **Checklist Numbers**: 148. Access control specification.

24. ***Ownable might be too simple***

24.1. **Details**: The owner role has many privileges: enabling/disabling executors, allowing/disallowing paths for swap, rescuing funds from an exited strategy. Losing control over this address would have its impact. Ownable does not support two step change of the owner address which might make it too simple to use in this context.

24.2. **Mitigation**: Consider changing the role handling to a more robust one which handles two-step change of privileged roles.

24.3. **Tools/Techniques**: Manual review.

24.4. **Difficulty+Impact**: Difficulty: Low, Impact: Low

24.5. **Checklist Numbers**: 162. Two-step change of privileged roles.

25. ***Named return variables***

25.1. **Summary**: There is a mix use of named return variables across contracts in scope.

25.2. **Details**: Both in *BaseStrategy* and *AaveStrategy* there are several places in which the returned variable is declared in the function definition without the explicit return instruction within the function's body.
25.3. **Github Permalinks**:
25.4. Explicit:
25.5. https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L217-L227
Implicit:
https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L231-L236
25.6. **Mitigation**: Consider removing all named return variables, explicitly declaring them as local variables in the body of the function, and adding the necessary explicit return statements where appropriate. This should favor both explicitness and readability of the project.
25.7. **Tools/Techniques**: Manual review.
25.8. **Difficulty+Impact**: Difficulty: Low, Impact: Low
25.9. **Checklist Numbers**: 164. Explicit over Implicit

26. ***Function behaviour implicitness***
26.1. **Summary**: *setStrategyExecutor function* implicitly sets/removes executors.
26.2. **Details**: *setStrategyExecutor* external function accepts an address and a boolean a parameter where the boolean parameter defines whether executor rights are granted or revoked for the address.
26.3. **Github Permalinks**:
https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L136-L138
26.4. **Mitigation**: Consider splitting the existing *setStrategyExecutor* in two different functions which explicitly grants/revokes executor rights.
26.5. **Tools/Techniques**: Manual review.
26.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
26.7. **Checklist Numbers**: 164. Explicit over Implicit

27. ***Missing error-reporting***
27.1. **Summary**: AaveStratgey's *_exit* implementation in case of an exception in both catch blocks the contract swallows the exception.
27.2. **Details**: *BaseStrategy's* rule on the *_exit* function is that child contracts should not allow this function to revert. Following this *AaveStrategy* uses try-catch to execute the logic in the two possible scenarios. But in both blocks the catch clause is empty which can lead to unnoticed errors.
27.3. **Github Permalinks**:
https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/strategies/AaveStrategy.sol#L90-L95
27.4. **Mitigation**: Consider using an event in both catch blocks to report error conditions and/or exceptional behavior.
27.5. **Tools/Techniques**: Manual review.
27.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
27.7. **Checklist Numbers**: 175. Error-reporting issues

28. ***exited state variable visibility.***
    28.1. **Summary**: *exited* state variable's visibility could be restricted.
    28.2. **Details**: Exited logic and the depending *isActive* modifier is defined by state changes of *BaseStrategy's exited* state variable. An extending child contract might accidentally modify the *exited* variable resulting in a misbehaving modifier and an incorrect contract state.
    28.3. **Github Permalinks**: https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L33-L34
    28.4. **Mitigation**: Consider declaring the *exited* variable as private.
    28.5. **Tools/Techniques**: Manual review.
    28.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
    28.7. **Checklist Numbers**: 140. 192

29. ***maxBentoBoxBalance state variable visibility.***
    29.1. **Summary**: *maxBentoBoxBalance* state variable's visibility could be restricted.
    29.2. **Details**: *maxBentoBoxBalance defines* the maximum balance of the underlying token that is allowed to be in BentoBox. An extending child contract might accidentally modify the *maxBentoBoxBalance* variable resulting in an unintended ratio of token allocation.
    29.3. **Github Permalinks**: *https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L36-L37*
    29.4. **Mitigation**: Consider declaring the *maxBentoBoxBalance* variable as private.
    29.5. **Tools/Techniques**: Manual review.
    29.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
    29.7. **Checklist Numbers**: 140. 192

30. ***strategyExecutors state variable is visibility.***
    30.1. **Summary**: *strategyExecutors* state variable's visibility could be restricted.
    30.2. **Details**: *strategyExecutors* mapping holds the addresses that are allowed to execute the *safeHarvest* function. An extending child contract might accidentally modify the *strategyExecutors* mapping resulting in either removal of valid executors or addition of unauthorized/invalid ones.
    30.3. **Github Permalinks**: *https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L39-L40*
    30.4. **Mitigation**: Consider declaring the *maxBentoBoxBalance* variable as private.
    30.5. **Tools/Techniques**: Manual review.
    30.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low
    30.7. **Checklist Numbers**: 140. 192

31. ***Swap functionality separation.***
    31.1. **Summary**: BaseStrategy defines the possible swap functionality for every child contract even if the child contract will not intend to use it.
    31.2. **Details**: Leaving the *factory* address state variable uninitialized and the *_allowedSwapPaths* array empty is designated to implicitly signal that the

extending child contract does support token swapping. It is considered a bad practice in software development to leave methods / variables empty or throw an exception when the current class / contract was never intended to be capable of handling the called functionality. No code should be forced to depend on methods it does not use.

31.3. **Github Permalinks**:
https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L66-L67
Reverting if factory is not set:
https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L283-L287

31.4. **Mitigation**: Consider to split this functionality into a different contract (for example: Swappable). This will ensure, that only those contracts will extend it that will use and support this functionality.

31.5. **Tools/Techniques**: Manual review.

31.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low

31.7. **Checklist Numbers**: 194. Principle of Least Common Mechanism + Interface segregation principle (Not from security pitfalls)

32. ***setAllowedPath incorrectly emits the pathIndex***

32.1. **Summary**: *setAllowedPath* function emits an event *LogSetAllowedPath* with incorrectly calculated pathId parameter..

32.2. **Details**: *LogSetAllowedPath* event is emitted when a path is being added to or invalidated in the _allowedSwapPaths array. The event's first parameter is the pathId which is intended to mark the index of the added/erased path for (offchain) tracking and for later use. *setAllowedPath* function incorrectly calculates the index as _allowedSwapPaths.length, the correct calculation is _allowedSwapPaths.length - 1

32.3. **Github Permalinks**:
https://github.com/sushiswap/bentobox-strategies/blob/be407e27cd1a9ca8060ef9a389d1cf01b4e5540e/contracts/BaseStrategy.sol#L271

32.4. **Mitigation**: Consider fixing the index calculation using the correct formula.

32.5. **Tools/Techniques**: Manual review.

32.6. **Difficulty+Impact**: Difficulty: Low, Impact: Low

32.7. **Checklist Numbers**: 140. 192

Appendix