

# SKELETON ECOSYSTEM

SMART CONTRACT AUDIT



**VAULTGUARD**  
**[VAULTG]**  
**BEP 20**

0xF249dbA246C9Ac0499CbC32b89fA3C7EC7c7a1f0



## Table of Contents

|                                       |    |
|---------------------------------------|----|
| Table of Contents                     | 1  |
| Disclaimer                            | 2  |
| Overview                              | 3  |
| Creation/Audit Date                   | 3  |
| Verified Socials                      | 3  |
| Contract Functions Analysis           | 4  |
| Contract Safety and Weakness          | 6  |
| Detected Vulnerability<br>Description | 9  |
| Contract Flow Chart                   | 15 |
| Inheritance Graph                     | 16 |
| Contract Descriptions                 | 17 |
| Audit Scope                           | 19 |

## Global Disclaimer

This document serves as a disclaimer for the crypto smart contract audit conducted by Skeleton Ecosystem. The purpose of the audit was to review the codebase of the smart contracts for potential vulnerabilities and issues. It is important to note the following:

**Limited Scope:** The audit is based on the code and information available up to the audit completion date. It does not cover external factors, system interactions, or changes made after the audit. The audit itself can not guarantee 100% safety and can not detect common scam methods like farming and developer sell-out.

**No Guarantee of Security:** While we have taken reasonable steps to identify vulnerabilities, it is impossible to guarantee the complete absence of security risks or issues. The audit report provides an assessment of the contract's security as of the audit date.

**Continued Development:** Smart contracts and blockchain technology are evolving fields. Updates, forks, or changes to the contract post-audit may introduce new risks that were not present during the audit.

**Third-party Code:** If the smart contract relies on third-party libraries or code, those components were not thoroughly audited unless explicitly stated. Security of these dependencies is the responsibility of their respective developers.

**Non-Exhaustive Testing:** The audit involved automated analysis, manual review, and testing under controlled conditions. It is possible that certain vulnerabilities or issues may not have been identified.

**Risk Evaluation:** The audit report includes a risk assessment for identified vulnerabilities. It is recommended that the development team carefully reviews and addresses these risks to mitigate potential exploits.

**Not Financial Advice:** This audit report is not intended as financial or investment advice. Decisions regarding the use, deployment, or investment in the smart contract should be made based on a comprehensive assessment of the associated risks.

By accessing and using this audit report, you acknowledge and agree to the limitations outlined above. Skeleton Ecosystem and its auditors shall not be held liable for any direct or indirect damages resulting from the use of the audit report or the smart contract itself.

Please consult with legal, technical, and financial professionals before making any decisions related to the smart contract.

## Overview

|                       |   |
|-----------------------|---|
| Contract Name         | VaultGuard  |
| Ticker/Symbol         | VaultG  |
| Blockchain            | Binance Smart Chain Bep20   |
| Contract Address      | 0xF249dbA246C9Ac0499CbC32b89fA3C7EC7c7a1f0  |
| Creator Address       | 0xff3166faf00EB93CFF54Fa9a41709365e203913   |
| Current Owner Address | 0xff3166faf00EB93CFF54Fa9a41709365e203913   |
| Contract Explorer     | <a href="https://bscscan.com/token/0xf249dba246c9ac0499cbc32b89fa3c7ec7c7a1f0">https://bscscan.com/token/0xf249dba246c9ac0499cbc32b89fa3c7ec7c7a1f0</a> |
| Compiler Version      | v0.8.19+commit.7dd6d404 (For token contract)  |
| License               | None  |
| Optimisation          | Yes with 200 Runs   |
| Total Supply          | 200,000,000 VaultG  |
| Decimals              | 18  |

## Creation/Audit

|                   |                        |
|-------------------|------------------------|
| Contract Deployed | Aug-27-2023            |
| Audit Created     | 31-Aug-23 21:00:00 UTC |
| Audit Update      | V 0.1                  |

## Verified Socials

|          |   |
|----------|---|
| Website  | <a href="https://vaultguard.io">https://vaultguard.io</a>                             |
| Telegram | <a href="https://t.me/VaultGuard">https://t.me/VaultGuard</a>                         |
| Twitter  | <a href="https://twitter.com/VaultGuardFlash">https://twitter.com/VaultGuardFlash</a> |



## Contract Function Analysis



Pass






















Attention Item



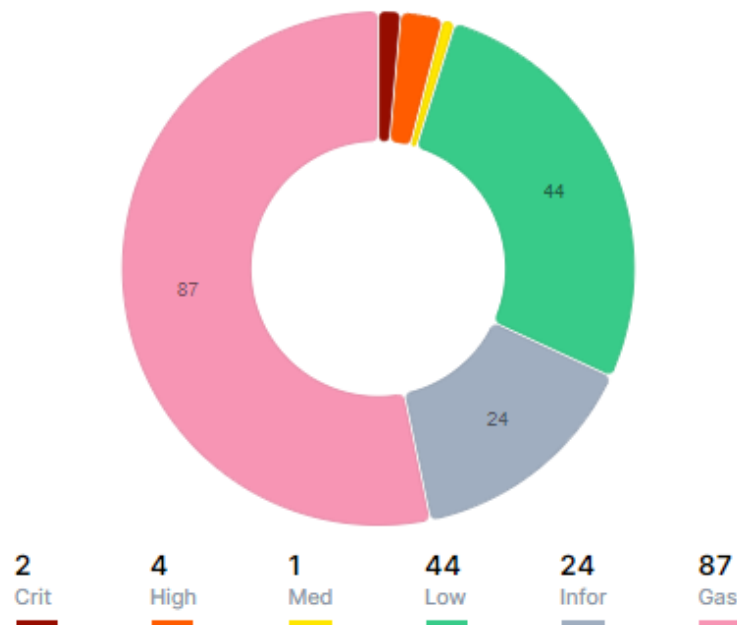
Risky Item

■ Pass ■ Attention ■ Risk

|                           |  |  |
|---------------------------|--|--|
| Contract Verified         |                 | The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.  |
| Contract Renounce         |                 | Current Owner: 0xff3166faf00EB93CFfF54Fa9a41709365e203913<br>Attention marked functions can be modified and used.  |
| Buy Tax                   | <b>9 %</b><br>(max 25%)  | Shows the taxes for purchase transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. <b>Setfee max found: 25% require(totalFees[0] &lt;= 2500 &amp;&amp; totalFees[1] &lt;= 2500 &amp;&amp; totalFees[2] &lt;= 2500, "TaxesDefaultRouter: Cannot exceed max total fee of 25%");</b>   |
| Sell Tax                  | <b>11 %</b><br>(max 25%)   | Shows the taxes for sell transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. <b>Setfee max found: 25% require(totalFees[0] &lt;= 2500 &amp;&amp; totalFees[1] &lt;= 2500 &amp;&amp; totalFees[2] &lt;= 2500, "TaxesDefaultRouter: Cannot exceed max total fee of 25%");</b>   |
| Honeypot Analyse          |                | Holder is able to buy and sell. If honeypot: The contract blocks sell transfer from holder wallet. Multiple events may cause honeypot. Trading disabled, extremely high tax  |
| Liquidity Status          |               | Locked on 31.08.2023: 94.05% Pinklock for 36523 days.<br>2.93% Pinklock for 364 days.<br>Note! Initial liquidity tokens scanned. For new LP Lockers allways re-check with skeleton scanner on telegram.  |
| Trading Disable Functions |               | No trading suspendable function found.<br>If a suspendable code is included, the token maybe neither be bought or sold (honeypot risk). If contract is renounced this function can't be used.  If there is authorised hidden owner, or there is Retrieve Ownership Function, the trading disable function may be used! |
| Set Fees function         | <br>(max 25%) | Fee Setting function found.<br><b>Setfee max found: 25%: require(totalFees[0] &lt;= 2500 &amp;&amp; totalFees[1] &lt;= 2500 &amp;&amp; totalFees[2] &lt;= 2500, "TaxesDefaultRouter: Cannot exceed max total fee of 25%");</b>   |
| Proxy Contract            |               | The proxy contract means contract owner can modify the function of the token and possibly effect the price. The Owner is not the creator but the creator may have authorisation to change functions.   |
| Mint Function             |               | No mint function found.<br>Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token. Owner can mint new tokens and sell. If contract is renounced this function can't be used.   |

|   |   |  |
|---|---|--|
| Balance Modifier Function                   |    | <p>No Balance Modifier function found.</p> <p>If there is a function for this, the contract owner can have the authority to modify the balance of tokens at other addresses. For example revoke the bought tokens from the holders wallet. Common form of scam: You buy the token, but it's disappearing from your wallet.</p> <p> If contract is renounced this function still can be used as auto self Destruct</p> |
| Whitelist Function                          |    | <p>Whitelist Function Found.</p> <p>If there is a function for this Developer can set zero fee or no max wallet size for addresses (for example team wallets can trade without fee. Can cause farming)</p> <p>If there is a whitelist, some addresses may not be able to trade normally (honeypot risk)</p>  |
| Hidden Owner Analysis                       |    | <p>No authorised hidden owner found.</p> <p>For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned. Fake renounce.</p>   |
| Retrieve Ownership Function                 |   | <p>No functions found which can retrieve ownership of the contract.</p> <p>If this function exists, it is possible for the project owner to regain ownership even after relinquishing it. Also known as fake renounce.</p>   |
| Self Destruct Function                      |  | <p>No Self Destruct function found.</p> <p>If this function exists and is triggered, the contract will be destroyed, all functions will be unavailable, and all related assets will be erased.</p>   |
| Specific Tax Changing Function              |  | <p>Specific Tax Changing Functions found.</p> <p>If it exists, the contract owner may set a very outrageous tax rate for assigned address to block it from trading. Can assign all wallets at once!</p>  |
| Trading Cooldown Function                   |  | <p>No Trading Cooldown Function found.</p> <p>If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying. Like a temporary honeypot.</p>   |
| Max Transaction and Holding Modify Function |  | <p>Max Transaction and Holding Modify function found.</p> <p>If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot</p>   |
| Transaction Limiting Function               |  | <p>No Transaction Limiter Function Found.</p> <p>The number of overall token transactions may be limited (honeypot risk)</p>   |

## Contract Safety and Weakness



|                                    |   |
|------------------------------------|---|
| ● PUBLIC BURN                      | 2 |
| ● UNCHECKED TRANSFER               | 2 |
| ● APPROVE FRONT-RUNNING ATTACK     | 2 |
| ● MODIFIER SIDE EFFECTS            | 1 |
| ● USE OF MULTIPLE PRAGMA VERSIONS  | 3 |
| ● MISCONFIGURED BEFORETOKENTRAN... | 4 |
| ● COMPILER VERSION TOO RECENT      | 1 |
| ● USE OWNABLE2STEP                 | 2 |
| ● MISSING EVENTS                   | 7 |
| ● LONG NUMBER LITERALS             | 3 |




|                                       |    |
|---------------------------------------|----|
| ● OUTDATED COMPILER VERSION           | 12 |
| ● USE OF FLOATING PRAGMA              | 12 |
| ● NAME MAPPING PARAMETERS             | 4  |
| ● BLOCK VALUES AS A PROXY FOR TIME    | 6  |
| ● UNUSED RECEIVE FALLBACK             | 1  |
| ● HARD-CODED ADDRESS DETECTED         | 5  |
| ● MISSING INDEXED KEYWORDS IN EVE...  | 8  |
| ● MULTIPLICATION/DIVISION BY TWO S... | 1  |
| ● INTERNAL FUNCTIONS NEVER USED       | 1  |
| ● DEFINE CONSTRUCTOR AS PAYABLE       | 2  |
| ● FUNCTION SHOULD BE EXTERNAL         | 6  |
| ● CHEAPER INEQUALITIES IN IF()        | 2  |
| ● SPLITTING REQUIRE STATEMENTS        | 3  |
| ● GAS OPTIMIZATION FOR STATE VARIA... | 1  |
| ● UNUSED IMPORTS                      | 2  |
| ● CHEAPER INEQUALITIES IN REQUIRE()   | 12 |
| ● CHEAPER CONDITIONAL OPERATORS       | 10 |
| ● LONG REQUIRE/REVERT STRINGS         | 19 |
| ● STORAGE VARIABLE CACHING IN MEM...  | 28 |

## Public Burn (2 item)

```

16     * @dev Destroys `amount` tokens from the caller.
17     *
18     * See {ERC20-_burn}.
19     */
20     function burn(uint256 amount) public virtual {
21         _burn(_msgSender(), amount);
22     }
23
24     /**
25     * @dev Destroys `amount` tokens from `account`, deducting from the caller's
26     * allowance.
27     *
28     * See {ERC20-_burn} and {ERC20-allowance}.
29     *
30
31     *
32     * - the caller must have allowance for ``accounts``'s tokens of at least
33     * `amount`.
34     */
35     function burnFrom(address account, uint256 amount) public virtual {
36         _spendAllowance(account, _msgSender(), amount);
37         _burn(account, amount);
38     }
39 }
40

```


| Function   | Severity  | Remediation  |
|--|---|--|
| <p>The contract was found to be using public or an external burn function. The function was missing access control to prevent another user from burning their tokens. Also, the burn function was found to be using a different address than msg.sender.</p> | <br>Severity :<br>Critical | <p>Consider adding access control modifiers to the burn function to prevent another user from burning their tokens. The burn function should use msg.sender in the _from argument.</p> |

## Unchecked Transfer (2 item)

```

312     }
313
314     if (fees > 0) {
315         super._transfer(from, address(this), fees);
316     }
317 }
318
319     super._transfer(from, to, amount);
320
321 }
322
323     function _updateRouterV2(address router) private {
324         routerV2 = IUniswapV2Router02(router);
325         pairV2 = IUniswapV2Factory(routerV2.factory()).createPair(address(this), routerV2.WETH());
326     }

```


| Function   | Severity   | Remediation   |
|--|--|---|
| Some tokens do not revert the transaction when the transfer or transferFrom fails and returns False. Hence we must check the return value after calling the transfer or transferFrom function. | <br>Severity : High | Use OpenZeppelin SafeERC20's safetransfer and safetransferFrom functions. |

## ⚠ Approve Front Running Attack (2 Item)

```


133  *
134  * = `spender` cannot be the zero address
135  */
136  function approve(address spender, uint256 amount) public virtual override returns (bool) {
137      address owner = _msgSender();
138      _approve(owner, spender, amount);
139      return true;
140  }
141
142  /**
143   * @dev See {IERC20-transferFrom}.
144   *
145   * ...

```

| Function   | Severity   | Remediation   |
|--|--|---|
| <p>The approve() method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account.</p> <p>Meanwhile, if the sender decides to change the amount and sends another approve transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the ERC20 Approve function.</p> <p>The function approve can be front-run by abusing the _approve function.</p> | <br>Severity : High | <p>Only use the approve function of the ERC/BEP standard to change the allowed amount to 0 or from 0 (wait till transaction is mined and approved).</p> <p>Token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e., that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Such checking is possible using advanced blockchain explorers such as [Etherscan.io](https://etherscan.io/)</p> <p>Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.</p> |

```

320     * Revert if not enough allowance is available.
321     *
322     * Might emit an {Approval} event.
323     */
324     function _spendAllowance(address owner, address spender, uint256 amount) internal virtual {
325         uint256 currentAllowance = allowance(owner, spender);
326         if (currentAllowance != type(uint256).max) {
327             require(currentAllowance >= amount, "ERC20: insufficient allowance");
328             unchecked {
329                 _approve(owner, spender, currentAllowance - amount);
330             }
331         }
332     }
333
334     /**
335     * @dev Hook that is called before any transfer of tokens. This includes
336     * minting and burning.
337     *
338     * Requirements:
339     * - caller must have sufficient allowance to burn
340     * - caller must have sufficient allowance to mint
341     */
342     function _beforeTokenTransfer(address from, address to, uint256 value) internal virtual {
343         // Do some sanity checks before emitting approval events
344         require(value > 0, "ERC20: cannot transfer a value of 0");
345         require(from != to, "ERC20: cannot self-transfer");
346         // Emit the Approval event
347         emit Approval(owner, from, to, value);
348     }
349
350     // ERC20 interface
351     // solhint-disable-next-line no-empty-blocks
352     function transfer(address to, uint256 value) public returns (bool) {
353         _transfer(msg.sender, to, value);
354         return true;
355     }
356
357     function transferFrom(address from, address to, uint256 value) public returns (bool) {
358         _transfer(from, to, value);
359         return true;
360     }
361
362     function approve(address spender, uint256 value) public returns (bool) {
363         _approve(msg.sender, spender, value);
364         return true;
365     }
366
367     function allowance(address owner, address spender) public view returns (uint256) {
368         return _allowance(owner, spender);
369     }
370
371     function totalSupply() public view returns (uint256) {
372         return _totalSupply();
373     }
374
375     function balanceOf(address owner) public view returns (uint256) {
376         return _balanceOf(owner);
377     }
378
379     function _allowance(address owner, address spender) internal view returns (uint256) {
380         uint256 currentAllowance = allowance(owner, spender);
381         if (currentAllowance != type(uint256).max) {
382             require(currentAllowance >= 0, "ERC20: insufficient allowance");
383         }
384         return currentAllowance;
385     }
386
387     function _balanceOf(address owner) internal view returns (uint256) {
388         uint256 currentBalance = balanceOf(owner);
389         if (currentBalance != type(uint256).max) {
390             require(currentBalance >= 0, "ERC20: insufficient balance");
391         }
392         return currentBalance;
393     }
394
395     function _totalSupply() internal view returns (uint256) {
396         uint256 currentTotalSupply = totalSupply();
397         if (currentTotalSupply != type(uint256).max) {
398             require(currentTotalSupply >= 0, "ERC20: insufficient total supply");
399         }
400         return currentTotalSupply;
401     }
402
403     function _transfer(address from, address to, uint256 value) internal {
404         require(from != to, "ERC20: cannot self-transfer");
405         // Emit the Transfer event
406         emit Transfer(from, to, value);
407     }
408
409     function _approve(address owner, address spender, uint256 value) internal {
410         require(owner != spender, "ERC20: cannot approve self");
411         // Emit the Approval event
412         emit Approval(owner, from, to, value);
413     }
414 }
415
416 // ERC20 Errors
417
418 error ERC20InsufficientAllowance(uint256 currentAllowance, uint256 amount, string message);
419 error ERC20InsufficientBalance(uint256 currentBalance, uint256 value, string message);
420 error ERC20InvalidApprover(address caller, address owner);
421 error ERC20InvalidSpender(address caller, address spender);
422 error ERC20Unauthorized(address caller);
423 error ERC20InvalidReceiver(address to);
424 error ERC20InvalidSender(address from);
425 error ERC20InvalidValue(uint256 value);
426 error ERC20NoTokens(uint256 value);
427 error ERC20SelfTransfer();
428 error ERC20TotalSupplyExceedsMax();
429 error ERC20ValueExceedsMax(uint256 value);
430 error ERC20ZeroValue();
431 error ERC20ZeroBalance();
432 error ERC20ZeroTotalSupply();
433 error ERC20ZeroValueFromMax();
434 error ERC20ZeroValueToMax();
435 error ERC20ZeroValueToZero();
436 error ERC20ZeroValueToMax();
437 error ERC20ZeroValueToZero();
438 error ERC20ZeroValueToMax();
439 error ERC20ZeroValueToZero();
440 error ERC20ZeroValueToMax();
441 error ERC20ZeroValueToZero();
442 error ERC20ZeroValueToMax();
443 error ERC20ZeroValueToZero();
444 error ERC20ZeroValueToMax();
445 error ERC20ZeroValueToZero();
446 error ERC20ZeroValueToMax();
447 error ERC20ZeroValueToZero();
448 error ERC20ZeroValueToMax();
449 error ERC20ZeroValueToZero();
450 error ERC20ZeroValueToMax();
451 error ERC20ZeroValueToZero();
452 error ERC20ZeroValueToMax();
453 error ERC20ZeroValueToZero();
454 error ERC20ZeroValueToMax();
455 error ERC20ZeroValueToZero();
456 error ERC20ZeroValueToMax();
457 error ERC20ZeroValueToZero();
458 error ERC20ZeroValueToMax();
459 error ERC20ZeroValueToZero();
460 error ERC20ZeroValueToMax();
461 error ERC20ZeroValueToZero();
462 error ERC20ZeroValueToMax();
463 error ERC20ZeroValueToZero();
464 error ERC20ZeroValueToMax();
465 error ERC20ZeroValueToZero();
466 error ERC20ZeroValueToMax();
467 error ERC20ZeroValueToZero();
468 error ERC20ZeroValueToMax();
469 error ERC20ZeroValueToZero();
470 error ERC20ZeroValueToMax();
471 error ERC20ZeroValueToZero();
472 error ERC20ZeroValueToMax();
473 error ERC20ZeroValueToZero();
474 error ERC20ZeroValueToMax();
475 error ERC20ZeroValueToZero();
476 error ERC20ZeroValueToMax();
477 error ERC20ZeroValueToZero();
478 error ERC20ZeroValueToMax();
479 error ERC20ZeroValueToZero();
480 error ERC20ZeroValueToMax();
481 error ERC20ZeroValueToZero();
482 error ERC20ZeroValueToMax();
483 error ERC20ZeroValueToZero();
484 error ERC20ZeroValueToMax();
485 error ERC20ZeroValueToZero();
486 error ERC20ZeroValueToMax();
487 error ERC20ZeroValueToZero();
488 error ERC20ZeroValueToMax();
489 error ERC20ZeroValueToZero();
490 error ERC20ZeroValueToMax();
491 error ERC20ZeroValueToZero();
492 error ERC20ZeroValueToMax();
493 error ERC20ZeroValueToZero();
494 error ERC20ZeroValueToMax();
495 error ERC20ZeroValueToZero();
496 error ERC20ZeroValueToMax();
497 error ERC20ZeroValueToZero();
498 error ERC20ZeroValueToMax();
499 error ERC20ZeroValueToZero();
500 error ERC20ZeroValueToMax();
501 error ERC20ZeroValueToZero();
502 error ERC20ZeroValueToMax();
503 error ERC20ZeroValueToZero();
504 error ERC20ZeroValueToMax();
505 error ERC20ZeroValueToZero();
506 error ERC20ZeroValueToMax();
507 error ERC20ZeroValueToZero();
508 error ERC20ZeroValueToMax();
509 error ERC20ZeroValueToZero();
510 error ERC20ZeroValueToMax();
511 error ERC20ZeroValueToZero();
512 error ERC20ZeroValueToMax();
513 error ERC20ZeroValueToZero();
514 error ERC20ZeroValueToMax();
515 error ERC20ZeroValueToZero();
516 error ERC20ZeroValueToMax();
517 error ERC20ZeroValueToZero();
518 error ERC20ZeroValueToMax();
519 error ERC20ZeroValueToZero();
520 error ERC20ZeroValueToMax();
521 error ERC20ZeroValueToZero();
522 error ERC20ZeroValueToMax();
523 error ERC20ZeroValueToZero();
524 error ERC20ZeroValueToMax();
525 error ERC20ZeroValueToZero();
526 error ERC20ZeroValueToMax();
527 error ERC20ZeroValueToZero();
528 error ERC20ZeroValueToMax();
529 error ERC20ZeroValueToZero();
530 error ERC20ZeroValueToMax();
531 error ERC20ZeroValueToZero();
532 error ERC20ZeroValueToMax();
533 error ERC20ZeroValueToZero();
534 error ERC20ZeroValueToMax();
535 error ERC20ZeroValueToZero();
536 error ERC20ZeroValueToMax();
537 error ERC20ZeroValueToZero();
538 error ERC20ZeroValueToMax();
539 error ERC20ZeroValueToZero();
540 error ERC20ZeroValueToMax();
541 error ERC20ZeroValueToZero();
542 error ERC20ZeroValueToMax();
543 error ERC20ZeroValueToZero();
544 error ERC20ZeroValueToMax();
545 error ERC20ZeroValueToZero();
546 error ERC20ZeroValueToMax();
547 error ERC20ZeroValueToZero();
548 error ERC20ZeroValueToMax();
549 error ERC20ZeroValueToZero();
550 error ERC20ZeroValueToMax();
551 error ERC20ZeroValueToZero();
552 error ERC20ZeroValueToMax();
553 error ERC20ZeroValueToZero();
554 error ERC20ZeroValueToMax();
555 error ERC20ZeroValueToZero();
556 error ERC20ZeroValueToMax();
557 error ERC20ZeroValueToZero();
558 error ERC20ZeroValueToMax();
559 error ERC20ZeroValueToZero();
560 error ERC20ZeroValueToMax();
561 error ERC20ZeroValueToZero();
562 error ERC20ZeroValueToMax();
563 error ERC20ZeroValueToZero();
564 error ERC20ZeroValueToMax();
565 error ERC20ZeroValueToZero();
566 error ERC20ZeroValueToMax();
567 error ERC20ZeroValueToZero();
568 error ERC20ZeroValueToMax();
569 error ERC20ZeroValueToZero();
570 error ERC20ZeroValueToMax();
571 error ERC20ZeroValueToZero();
572 error ERC20ZeroValueToMax();
573 error ERC20ZeroValueToZero();
574 error ERC20ZeroValueToMax();
575 error ERC20ZeroValueToZero();
576 error ERC20ZeroValueToMax();
577 error ERC20ZeroValueToZero();
578 error ERC20ZeroValueToMax();
579 error ERC20ZeroValueToZero();
580 error ERC20ZeroValueToMax();
581 error ERC20ZeroValueToZero();
582 error ERC20ZeroValueToMax();
583 error ERC20ZeroValueToZero();
584 error ERC20ZeroValueToMax();
585 error ERC20ZeroValueToZero();
586 error ERC20ZeroValueToMax();
587 error ERC20ZeroValueToZero();
588 error ERC20ZeroValueToMax();
589 error ERC20ZeroValueToZero();
590 error ERC20ZeroValueToMax();
591 error ERC20ZeroValueToZero();
592 error ERC20ZeroValueToMax();
593 error ERC20ZeroValueToZero();
594 error ERC20ZeroValueToMax();
595 error ERC20ZeroValueToZero();
596 error ERC20ZeroValueToMax();
597 error ERC20ZeroValueToZero();
598 error ERC20ZeroValueToMax();
599 error ERC20ZeroValueToZero();
600 error ERC20ZeroValueToMax();
601 error ERC20ZeroValueToZero();
602 error ERC20ZeroValueToMax();
603 error ERC20ZeroValueToZero();
604 error ERC20ZeroValueToMax();
605 error ERC20ZeroValueToZero();
606 error ERC20ZeroValueToMax();
607 error ERC20ZeroValueToZero();
608 error ERC20ZeroValueToMax();
609 error ERC20ZeroValueToZero();
610 error ERC20ZeroValueToMax();
611 error ERC20ZeroValueToZero();
612 error ERC20ZeroValueToMax();
613 error ERC20ZeroValueToZero();
614 error ERC20ZeroValueToMax();
615 error ERC20ZeroValueToZero();
616 error ERC20ZeroValueToMax();
617 error ERC20ZeroValueToZero();
618 error ERC20ZeroValueToMax();
619 error ERC20ZeroValueToZero();
620 error ERC20ZeroValueToMax();
621 error ERC20ZeroValueToZero();
622 error ERC20ZeroValueToMax();
623 error ERC20ZeroValueToZero();
624 error ERC20ZeroValueToMax();
625 error ERC20ZeroValueToZero();
626 error ERC20ZeroValueToMax();
627 error ERC20ZeroValueToZero();
628 error ERC20ZeroValueToMax();
629 error ERC20ZeroValueToZero();
630 error ERC20ZeroValueToMax();
631 error ERC20ZeroValueToZero();
632 error ERC20ZeroValueToMax();
633 error ERC20ZeroValueToZero();
634 error ERC20ZeroValueToMax();
635 error ERC20ZeroValueToZero();
636 error ERC20ZeroValueToMax();
637 error ERC20ZeroValueToZero();
6
```


| Function  | Severity   | Remediation   |
|---|--|---|
| <p>The <code>_spendAllowance()</code> method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account.</p> <p>Meanwhile, if the sender decides to change the amount and sends another approve transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the ERC20 Approve function.</p> <p>The function <code>_spendAllowance</code> can be front-run by abusing the approve function.</p> |  <p>Severity : High</p> | <p>Only use the approve function of the ERC/BEP standard to change the allowed amount to 0 or from 0 (wait till transaction is mined and approved).</p> <p>Token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e., that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Such checking is possible using advanced blockchain explorers such as [Etherscan.io](https://etherscan.io/)</p> <p>Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.</p> |

### ⚠️ Modifier Side Effects (1 Item)

```

28
29
30  * @dev Modifier to protect an initializer function from being invoked twice.
31  */
32  modifier initializer() {
33      require(_initializing || !_initialized, "Initializable: contract is already initialized");
34
35      bool isTopLevelCall = !_initializing;
36      if (isTopLevelCall) {
37          _initializing = true;
38          _initialized = true;
39      }
40
41      _;
42
43      if (isTopLevelCall) {
44          _initializing = false;
45      }
46  }

```


| Function   | Severity  | Remediation   |
|--|---|---|
| <p>Solidity functions should always use the Checks-Effects-Interactions pattern which states that the initial stage will contain only checks and validations which resides in the modifiers. Due to this reason, modifiers should only implement checks and validations inside of it and should not make state changes and external calls.</p> <p>The contract Initializable was found to be violating this pattern and the modifier initializer was making sensitive state changes and modifications.</p> | <br>Severity<br>: Medium | <p>Only use modifiers for implementing checks and validations. Do not make external calls or state changing actions inside modifiers.</p> |

### ⚠ Use of Multiple Pragma Versions (1 Item)

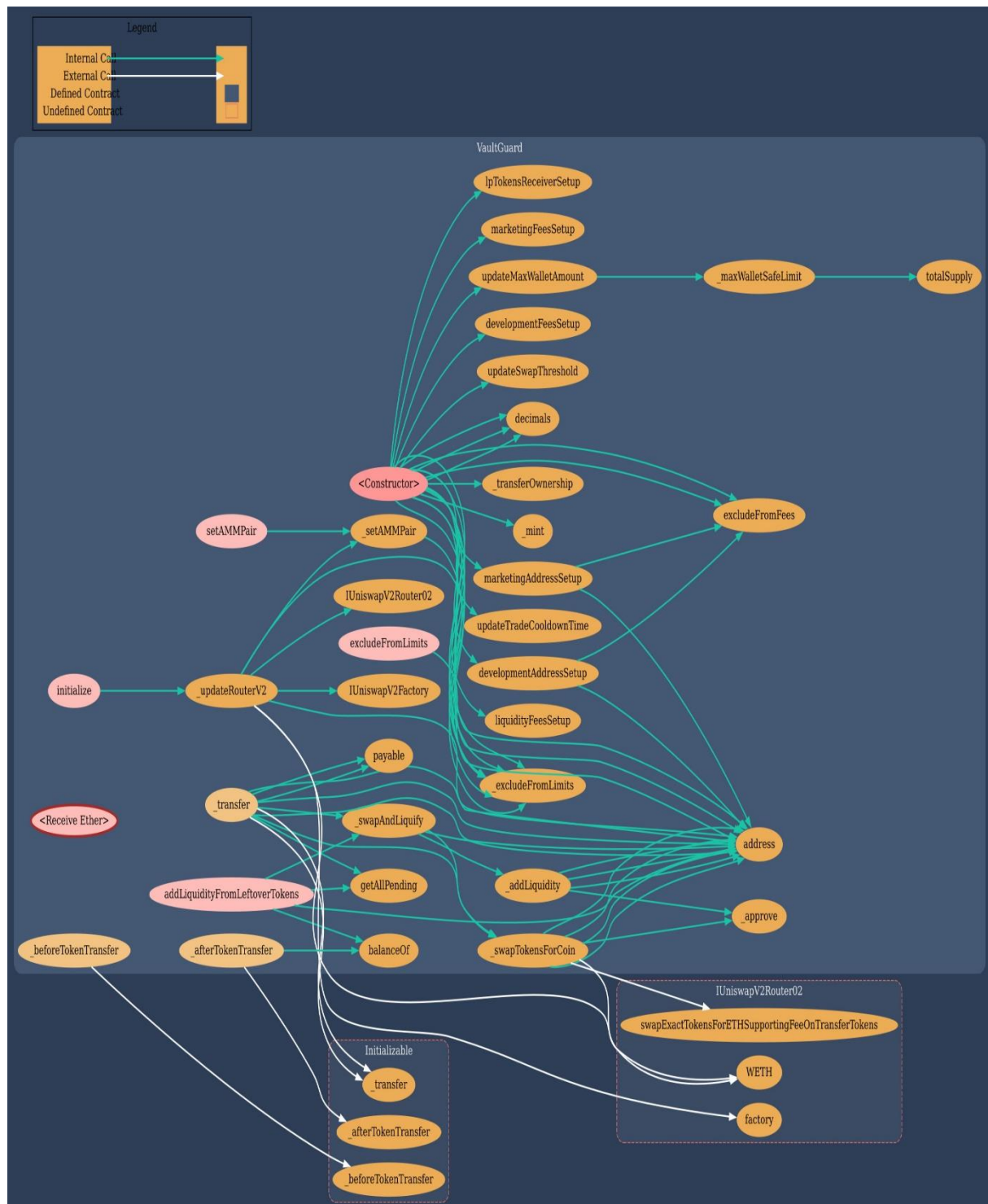
```

1  pragma solidity >=0.5.0;
2
3  interface IUniswapV2Factory {
4      event PairCreated(address indexed token0, address indexed token1, address pair, uint);
5
1  pragma solidity >=0.5.0;
2
3  interface IUniswapV2Pair {
4      event Approval(address indexed owner, address indexed spender, uint value);
5
1  pragma solidity >=0.6.2;
2
3  import './IUniswapV2Router01.sol';
4
1  pragma solidity >=0.6.2;
2
3  interface IUniswapV2Router01 {
4      function factory() external pure returns (address);

```

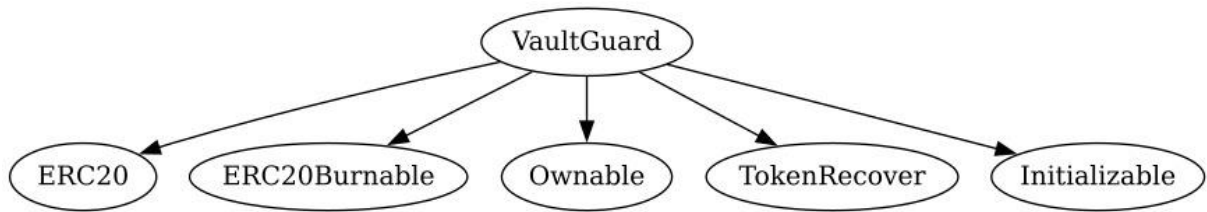
| Function  | Severity  | Remediation   |
|---|---|---|
| Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version. The project was found to be using multiple pragma versions across files which are not considered safe as they can be compiled with all the versions described. | <br>Severity : Low | >=0.6.2 and pragma version and >=0.5.0 pragma version was found in the above solidity files. We recommend updating the Solidity version to 0.8.18 which is considered stable and time-tested. It is also recommended to keep only one version of Solidity across all the contracts. |

## Contract Flow Graph








































## Inheritance Graph



## Contract Descriptions

| Contract   | Type                           | Bases   |   |             |
|------------|--------------------------------|---|---|-------------|
|            | Function Name                  | Visibility  | Mutability  | Modifiers   |
| VaultGuard | Implementation                 | ERC20,<br>ERC20Burnable,<br>Ownable,<br>TokenRecover,<br>Initializable                      |   |             |
|            |                                | Public !  |    | ERC20       |
|            | initialize                     | External !  |   | initializer |
|            |                                | External !  |  | NO !        |
|            | decimals                       | Public !  |   | NO !        |
|            | _swapTokensForCoin             | Private  |  |             |
|            | updateSwapThreshold            | Public !  |  | onlyOwner   |
|            | getAllPending                  | Public !  |   | NO !        |
|            | developmentAddressSetup        | Public !  |  | onlyOwner   |
|            | developmentFeesSetup           | Public !  |  | onlyOwner   |
|            | marketingAddressSetup          | Public !  |  | onlyOwner   |
|            | marketingFeesSetup             | Public !  |  | onlyOwner   |
|            | _swapAndLiquify                | Private  |  |             |
|            | _addLiquidity                  | Private  |  |             |
|            | addLiquidityFromLeftoverTokens | External !  |  | onlyOwner   |
|            | lpTokensReceiverSetup          | Public !  |  | onlyOwner   |

|  |                         |  |   |           |
|--|-------------------------|--|---|-----------|
|  | liquidityFeesSetup      | Public !   |    | onlyOwner |
|  | excludeFromFees         | Public !   |    | onlyOwner |
|  | _transfer               | Internal    |    |           |
|  | _updateRouterV2         | Private     |    |           |
|  | setAMMPair              | External !   |    | onlyOwner |
|  | _setAMMPair             | Private     |    |           |
|  | excludeFromLimits       | External !   |    | onlyOwner |
|  | _excludeFromLimits      | Internal    |    |           |
|  | updateMaxWalletAmount   | Public !   |    | onlyOwner |
|  | _maxWalletSafeLimit     | Private     |   |           |
|  | updateTradeCooldownTime | Public !   |  | onlyOwner |
|  | _beforeTokenTransfer    | Internal  |  |           |
|  | _afterTokenTransfer     | Internal  |  |           |



Function  
can modify  
state



Function  
is payable

## Source:

File Name SHA-1 Hash

c:\Solidity\vaultguard.sol de073a5a03a546819dff8c35914a0876af383728



## Audit Scope

### Audit Method.

Our smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. Goal: discover errors, issues and security vulnerabilities in the code. Findings getting reported and improvements getting suggested.

### Automatic and Manual Review

We are using automated tools to scan functions and weaknesses of the contract. Transfers, integer over-undeflow checks such as all CWE events.

### Tools we use:

Visual Studio Code

CWE

SWC

Solidity Scan

SVD

In manual code review our auditor looking at source code and performing line by line examination. This method helps to clarify developer's coding decisions and business logic.

### Skeleton Ecosystem

<https://skeletonecosystem.com>

<https://github.com/SkeletonEcosystem/Audits>

