



Cyberscope

Audit Report

CHAMP

July 2025

Address 0x8B3fC2d3Fc2156f2C19e6e6A4c24b5C7fFD21BC1

Network BASE

Audited by © cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

| Severity | Code | Description | Status |
|----------|------|-------------------------|--------|
| ● | ST | Stops Transactions | Passed |
| ● | OTUT | Transfers User's Tokens | Passed |
| ● | ELFM | Exceeds Fees Limit | Passed |
| ● | MT | Mints Tokens | Passed |
| ● | BT | Burns Tokens | Passed |
| ● | BC | Blacklists Addresses | Passed |

Diagnostics

● Critical ● Medium ● Minor / Informative

| Severity | Code | Description | Status |
|----------|------|--|------------|
| ● | MVN | Misleading Variables Naming | Unresolved |
| ● | PLPI | Potential Liquidity Provision Inadequacy | Unresolved |
| ● | PMRM | Potential Mocked Router Manipulation | Unresolved |
| ● | PVC | Price Volatility Concern | Unresolved |
| ● | RCS | Redundant Conditional Statements | Unresolved |
| ● | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| ● | L13 | Divide before Multiply Operation | Unresolved |
| ● | L16 | Validate Variable Setters | Unresolved |

Table of Contents

| | |
|--|-----------|
| Analysis | 1 |
| Diagnostics | 2 |
| Table of Contents | 3 |
| Risk Classification | 4 |
| Review | 5 |
| Audit Updates | 5 |
| Source Files | 6 |
| Findings Breakdown | 7 |
| MVN - Misleading Variables Naming | 8 |
| Description | 8 |
| Recommendation | 8 |
| PLPI - Potential Liquidity Provision Inadequacy | 9 |
| Description | 9 |
| Recommendation | 10 |
| PMRM - Potential Mocked Router Manipulation | 11 |
| Description | 11 |
| Recommendation | 11 |
| PVC - Price Volatility Concern | 12 |
| Description | 12 |
| Recommendation | 12 |
| RCS - Redundant Conditional Statements | 13 |
| Description | 13 |
| Recommendation | 13 |
| L04 - Conformance to Solidity Naming Conventions | 14 |
| Description | 14 |
| Recommendation | 15 |
| L13 - Divide before Multiply Operation | 16 |
| Description | 16 |
| Recommendation | 16 |
| L16 - Validate Variable Setters | 17 |
| Description | 17 |
| Recommendation | 17 |
| Functions Analysis | 18 |
| Inheritance Graphs | 19 |
| Flow Graph | 20 |
| Summary | 21 |
| Disclaimer | 22 |
| About Cyberscope | 23 |

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation:** This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation:** This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical:** Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium:** Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor:** Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative:** Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

| Severity | Likelihood / Impact of Exploitation |
|-----------------------|--|
| ● Critical | Highly Likely / High Impact |
| ● Medium | Less Likely / High Impact or Highly Likely/ Lower Impact |
| ● Minor / Informative | Unlikely / Low to no Impact |

Review

| | |
|-------------------|---|
| Contract Name | CHAMP |
| Explorer | https://basescan.org/address/0x8b3fc2d3fc2156f2c19e6e6a4c24b5c7ffd21bc1 |
| Symbol | CCG |
| Decimals | 18 |
| Total Supply | 2,500,000,000 |
| Badge Eligibility | Yes |

Audit Updates

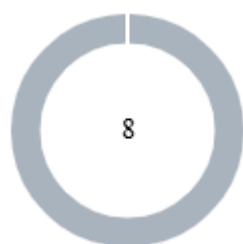
| | |
|-------------------|--|
| Initial Audit | 09 Mar 2025 https://github.com/cyberscope-io/audits/blob/main/4-ccg/v1/audit.pdf |
| Corrected Phase 2 | 21 Mar 2025 https://github.com/cyberscope-io/audits/blob/main/4-ccg/v2/audit.pdf |
| Corrected Phase 3 | 22 Apr 2025 https://github.com/cyberscope-io/audits/blob/main/4-ccg/v3/audit.pdf |
| Corrected Phase 4 | 15 May 2025 https://github.com/cyberscope-io/audits/blob/main/4-ccg/v4/audit.pdf |
| Corrected Phase 5 | 19 Jun 2025 |

| | |
|--------------------------|---|
| | https://github.com/cyberscope-io/audits/blob/main/4-ccg/v5/audit.pdf |
| Corrected Phase 6 | 09 Jul 2025 https://github.com/cyberscope-io/audits/blob/main/4-ccg/v6/audit.pdf |
| Corrected Phase 7 | 22 Jul 2025 https://github.com/cyberscope-io/audits/blob/main/4-ccg/v7/audit.pdf |
| Corrected Phase 8 | 26 Jul 2025 |
| Test Deploys | https://sepolia.etherscan.io/address/0x6D6acef7381626dfaeB40fA01796EFFba3fD9EB4 |

Source Files

| | |
|-------------------------|--|
| Filename | SHA256 |
| CryptoChamps.sol | acb3e0b967afdd26643a4e07b38b4514b36d2a37ec3f662d6327182b4a28f74c |

Findings Breakdown



| | |
|-----------------------|---|
| ● Critical | 0 |
| ● Medium | 0 |
| ● Minor / Informative | 8 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|-----------------------|------------|--------------|----------|-------|
| ● Critical | 0 | 0 | 0 | 0 |
| ● Medium | 0 | 0 | 0 | 0 |
| ● Minor / Informative | 8 | 0 | 0 | 0 |

MVN - Misleading Variables Naming

| | |
|-------------|------------------------|
| Criticality | Minor / Informative |
| Location | CryptoChamps.sol#L1135 |
| Status | Unresolved |

Description

Variables can have misleading names if their names do not accurately reflect the value they contain or the purpose they serve. The contract uses some variable names that are too generic or do not clearly convey the information stored in the variable. Misleading variable names can lead to confusion, making the code more difficult to read and understand. In this case, the `buyTax` is invoked for all taxed transactions, even if they are not buys.

```
uint256 transactionTax = (to == liquidityPool) ? sellTax : buyTax;
```

Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the readability of the code.

PLPI - Potential Liquidity Provision Inadequacy

| | |
|-------------|-----------------------------|
| Criticality | Minor / Informative |
| Location | CryptoChamps.sol#L1117,1181 |
| Status | Unresolved |

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
uint256 wethUsedInLiquidity = _addToLiquidity(
    ethForLiquidity,
    liquidityHalf
);
...
function _addToLiquidity(
    uint256 amountInEth,
    uint256 amountInToken
) private nonReentrant returns (uint256) {
    uint256 initialBalance = address(this).balance;
    uniswapRouter.addLiquidityETH{value: amountInEth}(
        address(this),
        amountInToken,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
    return initialBalance - address(this).balance;
}
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

PMRM - Potential Mocked Router Manipulation

| | |
|-------------|------------------------|
| Criticality | Minor / Informative |
| Location | CryptoChamps.sol#L1034 |
| Status | Unresolved |

Description

The contract includes a method that allows the owner to modify the router address and create a new pair. While this feature provides flexibility, it introduces a security threat. The owner could set the router address to any contract that implements the router's interface, potentially containing malicious code. In the event of a transaction triggering the swap functionality with such a malicious contract as the router, the transaction may be manipulated.

```
constructor(  
    address _uniswapRouter,  
    address _marketingWallet  
) ERC20("CryptoChamps", "CCG") Ownable(_msgSender())
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

PVC - Price Volatility Concern

| | |
|-------------|------------------------|
| Criticality | Minor / Informative |
| Location | CryptoChamps.sol#L1058 |
| Status | Unresolved |

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapThresholdAmount` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function changeSwapThreshold(uint256 _tokens) external onlyOwner {
    require(_tokens > 0, "Minimum token value must be higher than 0");
    swapThresholdAmount = _tokens;
    emit SwapThresholdUpdated(_tokens);
}
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RCS - Redundant Conditional Statements

| | |
|-------------|-----------------------------|
| Criticality | Minor / Informative |
| Location | CryptoChamps.sol#L1100,1106 |
| Status | Unresolved |

Description

The contract contains redundant conditional statements that can be simplified to improve code efficiency and performance. Conditional statements that are always true are unnecessary and lead to larger code size, increased memory usage, and slower execution times. Specifically, the `totalAllocation` is always greater than zero and the `liquidityTax` will never surpass the `contractTokenBalance`.

```
require(totalAllocation > 0, "Allocations must be > 0");
...
require(
    liquidityTax <= contractTokenBalance,
    "Invalid allocation: exceeds balance"
);
```

Recommendation

It is recommended to refactor conditional statements for redundancies. This practice minimizes the number of operations required, reduces the code footprint, and optimizes memory and gas usage. Simplifying such statements makes the code more readable and improves its overall performance.

L04 - Conformance to Solidity Naming Conventions

| | |
|--------------------|---|
| Criticality | Minor / Informative |
| Location | CryptoChamps.sol#L1054,1060,1221,1233,1247,1248 |
| Status | Unresolved |

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
uint256 _tokens
bool _applyTax
address _marketingWallet
uint256 _buyTax, uint256 _sellTax
uint256 _liquidityAllocation
uint256 _marketingAllocation
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/stable/style-guide.html#naming-conventions>.

L13 - Divide before Multiply Operation

| | |
|-------------|-----------------------------|
| Criticality | Minor / Informative |
| Location | CryptoChamps.sol#L1105,1111 |
| Status | Unresolved |

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
uint256 liquidityHalf = liquidityTax / 2;  
uint256 ethForLiquidity = (wethOutFromSwap * liquidityHalf) / tokensToSwap;
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L16 - Validate Variable Setters

| | |
|--------------------|------------------------|
| Criticality | Minor / Informative |
| Location | CryptoChamps.sol#L1225 |
| Status | Unresolved |

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
marketingWallet = _marketingWallet
```

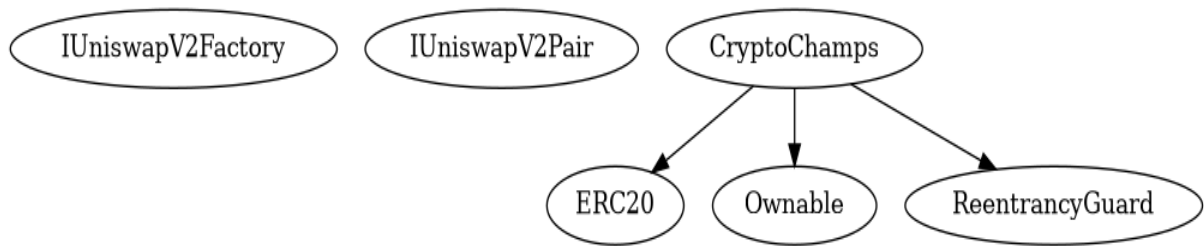
Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

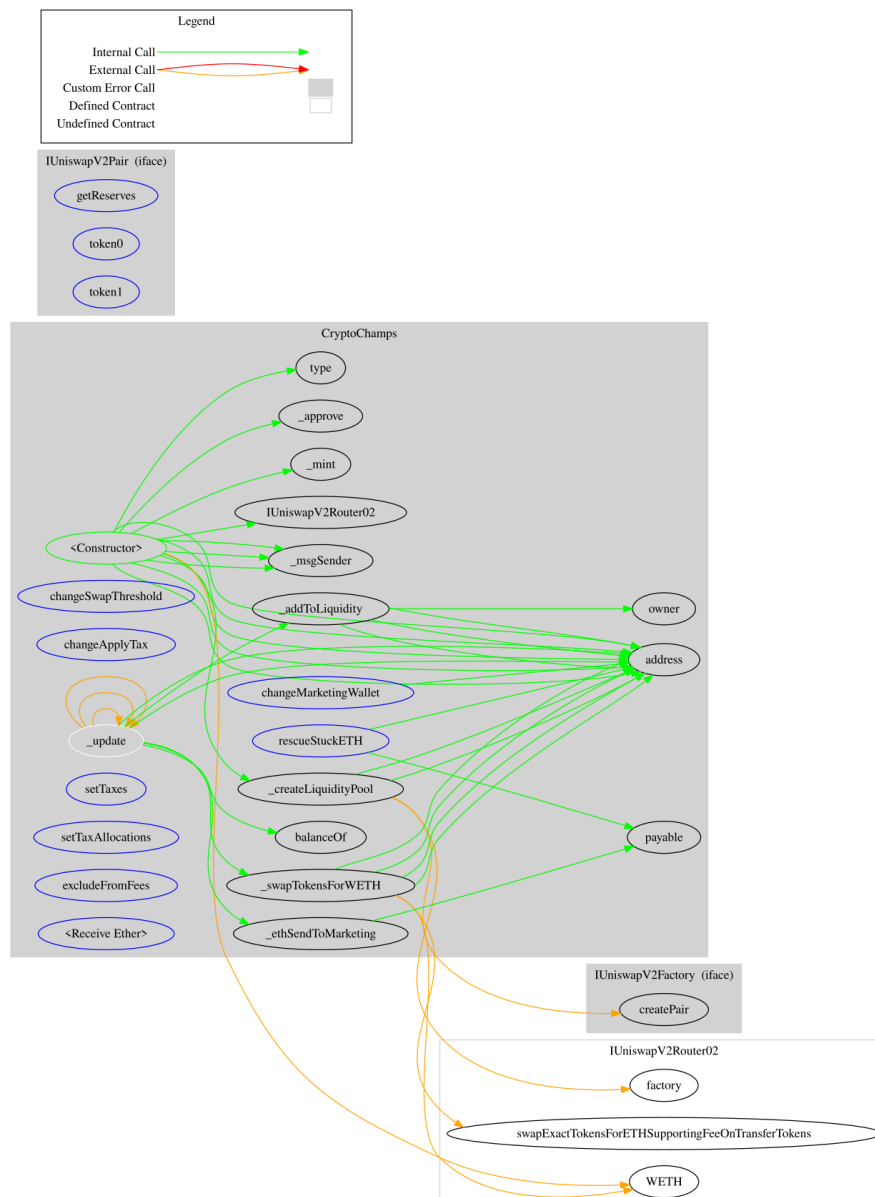
Functions Analysis

| Contract | Type | Bases | | |
|---------------------|-----------------------|---------------------------------------|------------|------------------|
| | Function Name | Visibility | Mutability | Modifiers |
| | | | | |
| CryptoChamps | Implementation | ERC20, Ownable, ReentrancyGuard | | |
| | | Public | ✓ | ERC20 Ownable |
| | changeSwapThreshold | External | ✓ | onlyOwner |
| | changeApplyTax | External | ✓ | onlyOwner |
| | _createLiquidityPool | Internal | ✓ | |
| | _update | Internal | ✓ | |
| | _swapTokensForWETH | Private | ✓ | nonReentrant |
| | _addToLiquidity | Private | ✓ | nonReentrant |
| | _ethSendToMarketing | Internal | ✓ | |
| | changeMarketingWallet | External | ✓ | onlyOwner |
| | setTaxes | External | ✓ | onlyOwner |
| | setTaxAllocations | External | ✓ | onlyOwner |
| | excludeFromFees | External | ✓ | onlyOwner |
| | rescueStuckETH | External | ✓ | onlyOwner |
| | | External | Payable | - |

Inheritance Graphs



Flow Graph



Summary

CHAMP contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. CHAMP is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 10% fees.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

cyberscope.io