HackSafe

Smart Contract Security Audit Report

Meta Apes Shell

July 2022

Security Status

★★★★☆

www.hacksafe.io

Audit Details

Audited project
Meta Apes Shell

Deployer address
0x58b529F9084D7eAA598EB3477Fe36064C5B7bbC1

Client contacts
Meta Apes Shell team

Blockchain
Binance Smart Chain

Website
https://metaapesgame.com/

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (HackSafe) owe no duty of care towards you or any other person, nor does HackSafe make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and HackSafe hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HackSafe hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HackSafe, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Step 1 - In-Depth Manual Review

Manual line-by-line code reviews to ensure the logic behind each function is sound and safe from various attack vectors. This is the most important and lengthy portion of the audit process (as automated tools often cannot find the nuances that lead to exploits such as flash loan attacks).

Step 2 - Automated Testing

Simulation of a variety of interactions with your Smart Contract on a test blockchain leveraging a combination of automated test tools and manual testing to determine if any security vulnerabilities exist.

Step 3 – Leadership Review

The engineers assigned to the audit will schedule meetings with our leadership team to review the contracts, any comments or findings, and ask questions to further apply adversarial thinking to discuss less common attack vectors.

Step 4 - Resolution of Issues

Consulting with the team to provide our recommendations to ensure the code's security and optimize its gas efficiency, if possible. We assist project team's in resolving any outstanding issues or implementing our recommendations.

Step 5 - Published Audit Report

Boiling down results and findings into an easy-to-read report tailored to the project. Our audit reports highlight resolved issues and any risks that exist to the project or its users, along with any remaining suggested remediation measures. Diagrams are included at the end of each report to help users understand the interactions which occur within the project.

Background

HackSafe was commissioned by Meta Apes Shell to perform an audit of smart contracts:

- https://bscscan.com/address/0x208cfEc94d2BA8B8537da7A9BB361c6baAD77272#code

The purpose of the audit was to achieve the

- Ensutre that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contract Details

Token contract details for 22.07.2022

Token Type	: BEP20
Contract name	: MintSwapCanonicalToken
Contract address	: 0x208cfEc94d2BA8B8537da7A9BB361c6baAD77272
Compiler version	: v0.8.9+commit.e5eed63a
Total supply	: 380,436,553.879725
Token Ticker	: SHELL
Decimals	: 18
Token Holders	: 440
Top 100 token holder's dominance	: 99.75%
Transactions count	: 13,853
Contract deployer address	: 0x58b529f9084d7eaa598eb3477fe36064c5b7bbc1
Owner address	: 0x58b529f9084d7eaa598eb3477fe36064c5b7bbc1

Social profiles

Twitter Profile	: https://twitter.com/MetaApesGame
Coinmarketcap profile	: https://coinmarketcap.com/currencies/meta-apes-shell/

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are “Secure”. This token contract does contain owner control, which do not make it fully decentralized as owner does have control over smart contract.

Insecure	Poor secured	Secure	Well-secured

You are here 

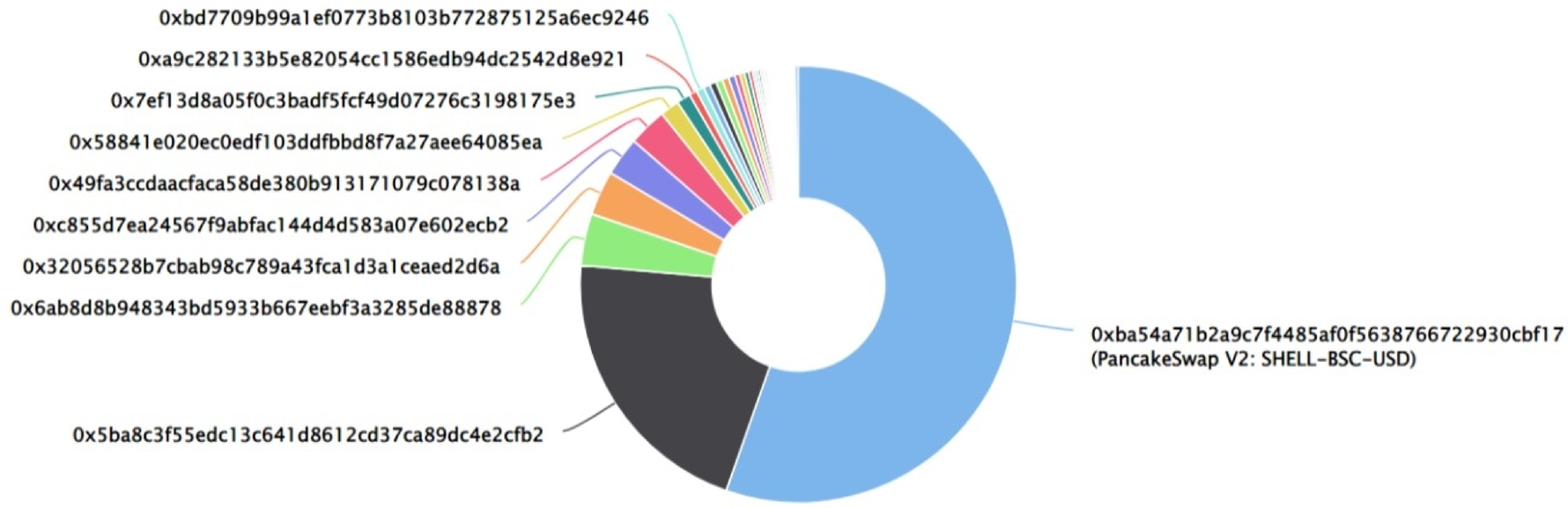
We used various tools like Slither, Mythril and Remix IDE. At the same time this finding is based on critical analysis of the manual audit. All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the issues checking status.

We found 0 critical, 0 high, 0 medium and 3 low and some very low-level issues. These issues are not critical ones.

Meta Apes Shell Token Distribution

The top 100 holders collectively own 99.75% (379,600,978.22 Tokens) of Meta Apes Shell

Token Total Supply: 380,559,862.66 Token | Total Token Holders: 441



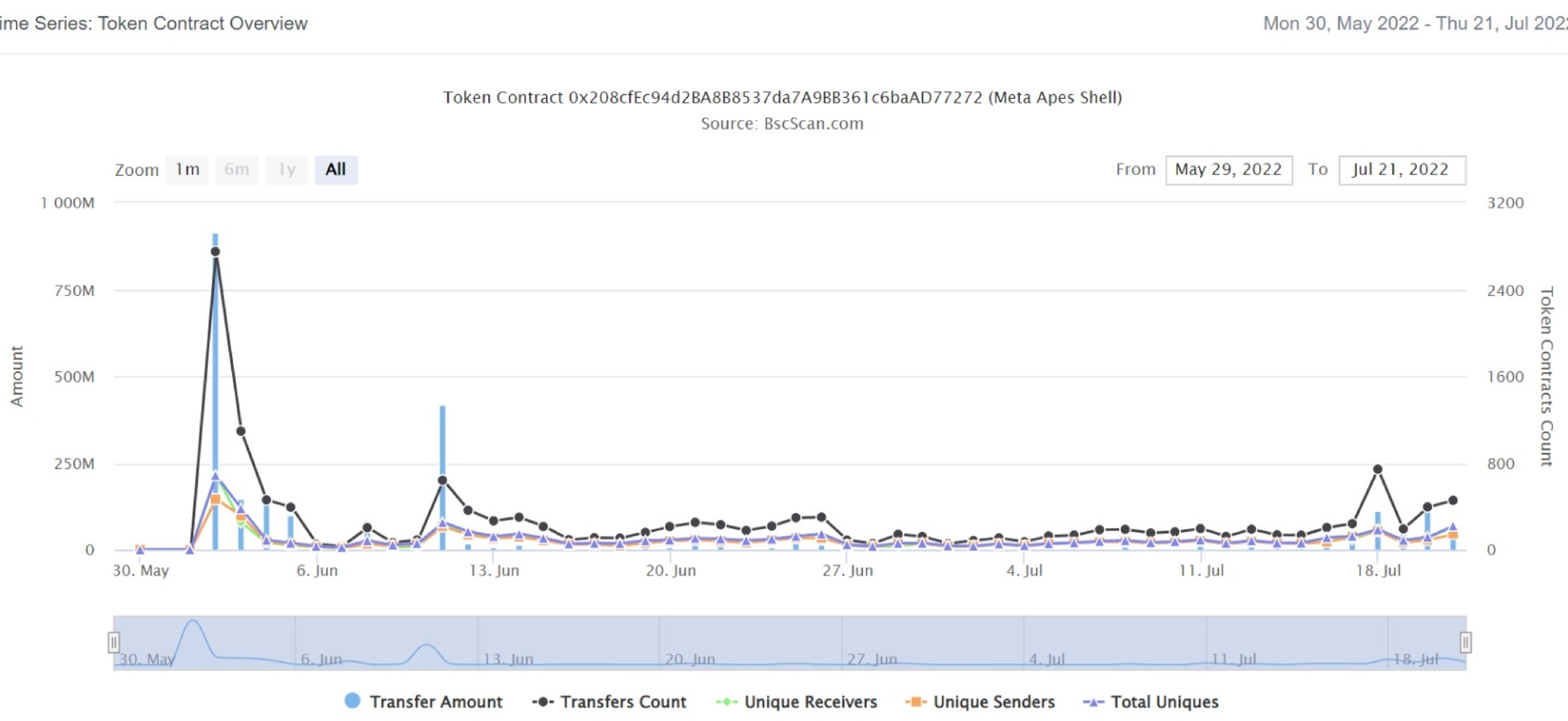
Meta Apes Shell Top 20 Token Holders

(A total of 379,600,978.22 tokens held by the top 100 accounts from the total supply of 380,559,862.66 token)

Rank	Address	Quantity (Token)	Percentage
1	PancakeSwap V2: SHELL-BSC-USD	210,735,748.795580150292607194	55.3752%
2	0x5ba8c3f55edc13c641d8612cd37ca89dc4e2cfb2	80,000,000	21.0217%
3	0x6ab8d8b948343bd5933b667eebf3a3285de88878	14,550,992.550152765214032536	3.8236%
4	0x32056528b7cbab98c789a43fca1d3a1ceaed2d6a	12,543,532.632088411625194015	3.2961%
5	0xc855d7ea24567f9abfac144d4d583a07e602ecb2	11,000,000.000000716567303162	2.8905%
6	0x49fa3ccdaacfaca58de380b913171079c078138a	10,803,481.356119473286749755	2.8388%
7	0x58841e020ec0edf103ddfbbd8f7a27aee64085ea	5,410,154.139973589726391187	1.4216%
8	0x7ef13d8a05f0c3badf5fcf49d07276c3198175e3	3,920,173.124858842950677972	1.0301%
9	0xa9c282133b5e82054cc1586edb94dc2542d8e921	2,202,907.508514481251111978	0.5789%
10	0xbd7709b99a1ef0773b8103b772875125a6ec9246	2,158,297.157280133872478611	0.5671%
11	0xcd6f15ab174d4f5abc1cb9cf515d1be7498a2747	1,904,136.599134694109391275	0.5004%
12	0x5b0e9d92467a076c461a37ebac54239ffa037d5f	1,891,319.639659405612932746	0.4970%
13	PancakeSwap V2: SHELL-BUSD	1,865,057.46411049053376407	0.4901%
14	0xab553e542f0eed992529ad9adbf75611079ba811	1,807,719.672203286572532165	0.4750%
15	0xabe5c51387a1a3757f56d6419a68554f77c420fd	1,787,465.025922500694640233	0.4697%
16	0x8f0d4687a07273ae91fd899932c0d30e1068f8ad	1,441,530.05757727621707584	0.3788%
17	0x483d3c007573b8be39c1a4b44300f53c9c92e693	1,369,864.141895423741109448	0.3600%
18	0xcfa9b093a75c98ebc89e9977a026d81c6ad07d36	1,194,382.888289806718415834	0.3138%
19	0x8280b08f90363ec123f8cb7ff0b5b265d05cf287	1,102,449.935908199957390493	0.2897%
20	0x04fbef2b7014a203b3c63c8a426c727252efa1a3	773,226.687274506733077202	0.2032%

Meta Apes Shell Token Distribution

Meta Apes Shell Contract Overview



Contract functions details

ERC20.sol
ERC20 (Context, IERC20, IERC20Metadata)

- -<constructor>
- -[Pub] name
- -[Pub] symbol
- -[Pub] decimals
- -[Pub] totalSupply
- -[Pub] balanceOf
- -[Pub] transfer #
- -[Pub] allowance
- -[Pub] approve #
- -[Pub] transferFrom #
- -[Pub] increaseAllowance
- -[Pub] decreaseAllowance
- -[Int] _transfer #
- -[Int] _mint #
- -[Int] _burn #
- -[Int] _approve #
- -[Int] _spendAllowance #
- -[Int] _beforeTokenTransfer #
- -[Int] _beforeTokenTransfer #

IERC20.sol
+ [Int] IERC20

- -[Ext] totalSupply
- -[Ext] balanceOf
- -[Ext] transfer
- -[Ext] allowance
- -[Ext] approve
- -[Ext] transferFrom

IERC20Metadata.sol
+ [Int] IERC20Metadata (IERC20)

- -[Pub] name
- -[Pub] symbol
- -[Pub] decimals

SafeERC20.sol
+[Lib] SafeERC20

- -[Int] safeTransfer

Contract functions details

-[Int] safeTransferFrom
-[Int] safeApprove
-[Int] safeIncreaseAllowance
-[Int] safeDecreaseAllowance
-[Pvt] _callOptionalReturn

Address.sol
+[Lib] Address
-[Int] isContract
-[Int] sendValue
-[Int] functionCall
-[Int] functionCall
-[Int] functionCallWithValue
-[Int] functionCallWithValue
-[Int] functionStaticCall
-[Int] functionStaticCall
-[Int] functionDelegateCall
-[Int] functionDelegateCall
-[Int] verifyCallResult

Context.sol
+ Context
-[Int] _msgSender
-[Int] _msgData

MintSwapCanonicalToken.sol
+ MintSwapCanonicalToken (MultiBridgeToken)
-<constructor>
-[Ext] swapBridgeForCanonical
-[Ext] swapCanonicalForBridge
-[Ext] setBridgeTokenSwapCap
-modifiers: onlyOwner

MultiBridgeToken.sol
MultiBridgeToken (ERC20, Ownable)
-<constructor>
-[Ext] mint
-[Ext] burn
-[Ext] burn
-[Ext] burnFrom

Contract functions details

-[Int] _burnFrom
-[Pub] decimals
-[Ext] updateBridgeSupplyCap
  -modifiers: onlyOwner
-[Ext] getOwner

Ownable.sol
+ Ownable
  -<constructor>
  -[Int] initOwner
  -[Pub] owner
  -[Pub] transferOwnership
    -modifiers: onlyOwner
  -[Pvt] _setOwner

($) = payable function
= non-constant function

Issues Checking Status

No.	Title	Status
1.	Unlocked Compiler Version	Low issue
2.	Missing Input Validation	Passed
3.	Race conditions and Reentrancy. Cross-function race conditions.	Passed
4.	Possible delays in data delivery	Passed
5.	Oracle calls.	Passed
6.	Timestamp dependence.	Passed
7.	Integer Overflow and Underflow	Passed
8.	DoS with Revert.	Passed
9.	DoS with block gas limit.	Passed
10.	Methods execution permissions.	Passed
11.	Economy model of the contract.	Passed
12.	Private use data leaks.	Passed
13.	Malicious Event log.	Passed
14.	Scoping and Declarations.	Passed
15.	Uninitialized storage pointers.	Passed
16.	Arithmetic accuracy.	Passed
17.	Design Logic.	Low issues
18.	Safe Open Zeppelin contracts implementation and usage.	Low issues
19.	Incorrect Naming State Variable	Passed
20.	Too old version	Passed

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.

Security Issues

Critical Severity Issues

No critical severity issue found.

High Severity Issues

No high severity issue found.

Medium Severity Issues

No medium severity issues found.

Low Severity Issues

Three low severity issue found.

1. Unlocked Compiler Version.

- **Description**

  The contract utilizes an unlocked compiler version. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

- **Recommendation**

  It is advisable that the compiler version is alternatively locked at the lowest version possible so that the contract can be compiled. For example, for version >=0.8.9 the contract should contain the following line:

  pragma solidity v0.8.9;

2. Design logic

Unused code

- **Description**

  The contract Address.sol has few functions which is unused.

- **Location**

  Library Address is unsed.

- **Recommendation**

  It is advisable to remove unused code for better coding style and it will save some computational power too.

3. Safe Open Zeppelin contracts implementation and usage.

- **Description**

  MintSwapCanonicalToken.sol and MultiBridgeToken.sol have direct imported openzeppelin contracts.

- **Recommendation**

  It is advisable to not use direct imports of openzeppelin contracts or any github repository as any changes in that contract can effect this assets.

Centralization

Owner privileges :

- Meta Apes Shell Contract:
    - Owner can transfer ownership.
    - Owner can update bridge supply cap.
    - Owner can set bridge token for swap.

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble as smart contract ownership has not been renounced. Following are Admin functions:

- Transferownership
- Updatebridgesupplycap
- Setbridgetokenswapcap

Conclusion

Smart contract contains low severity issues! The further transfer and operations with the fund raised are not related to this particular contract.

HackSafe note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.