Audit Report

NPC WEB3 GAMEFI

June 2025

Network	ARBITRUM
Address	0xc0c6698e11a7dd3d194cf8f57aa53f57b0cc56da
Audited by	© cyberscope

Table of Contents

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

Explorer	https://arbiscan.io/address/0xc0c6698e11a7dd3d194cf8f57aa53f57b0cc56da

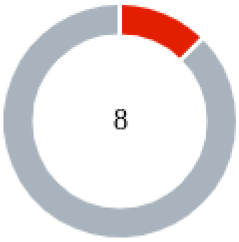
Audit Updates

Initial Audit	20 Jun 2025

Source Files

Filename	SHA256
U3.sol	5972af9c95e0b13314c50d70bb1612897494cd3186ac46f84c2d2500384d61f1
PancakeRouter.sol	87163f1e730c71e7c4c0cf34eed0375b3787b3deaf94ed9cb7796719145c4565
IERC20.sol	3f2306bb3fae7f4300917cc70efed3e46c3086d812a821edc040040ac826ebaf
AAInterface.sol	62b9c4ea55cdfe9412d08f3a36db183e9193e5c96a58b04a9c70e448125a1f9c

Findings Breakdown



Critical	1
Medium	0
Minor / Informative	7

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	0	0	0	0
Minor / Informative	7	0	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	ECV	External Contract Vulnerabilities	Unresolved
	MMN	Misleading Method Naming	Unresolved
	MEM	Missing Error Messages	Unresolved
	MU	Modifiers Usage	Unresolved
	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L06	Missing Events Access Control	Unresolved
	L16	Validate Variable Setters	Unresolved

ECV - External Contract Vulnerabilities

Criticality	Critical
Location	U3.sol
Status	Unresolved

Description

The contract's design introduces significant security risks due to inadequate validation of external contract interactions. It accepts unverified external contract addresses as parameters, allowing potential integration with malicious contracts that could execute unauthorized actions or malicious code. Additionally, the contract makes unprotected calls to a hardcoded, publicly accessible `burnMyTokenAmount` function on an external `AAInterface`, enabling any user to burn tokens from the external contract without validation, potentially draining its balance. The reliance on unverified third-party dependencies further exacerbates the risk, as these untrusted contracts could disrupt transaction flows or introduce malicious behavior. Even if the contract is designed as a utility with multiple functionalities, users must exercise caution when integrating with it, as granting allowances or permissions to the contract could allow malicious actors to exploit these vulnerabilities, leading to unauthorized actions or loss of assets.

Recommendation

It is recommended to implement robust validation mechanisms, such as whitelisting approved contract addresses, verifying interface compatibility, or using audited, trusted external contracts with immutable addresses. Enforce strict access controls and sender validation for sensitive functions like token burning, ensuring only authorized users can trigger such actions. These measures will mitigate risks of malicious exploitation and enhance the contract's security posture.

MMN - Misleading Method Naming

Criticality	Minor / Informative
Location	U3.sol#L42,62
Status	Unresolved

Description

Methods can have misleading names if their names do not accurately reflect the functionality they contain or the purpose they serve. The contract uses some method names that are too generic or do not clearly convey the underneath functionality. Misleading method names can lead to confusion, making the code more difficult to read and understand.

```
    function stT(address payable toAddr, uint256 value, uint256 order,
address invitation, string memory p1, string memory p2, string memory p3)
public payable onlyAdmin returns (bool) {
        ...
    }

     function stTK(address token, address to, uint value,uint256
order,address invitation,string memory p1,string memory p2,string memory
p3) public returns (bool){
        ...
    }
```

Recommendation

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

MEM - Missing Error Messages

Criticality	Minor / Informative
Location	U3.sol#L63,66,86,92,95,104,112,127,137,146,148,163,165
Status	Unresolved

Description

The contract is missing error messages. Specifically, there are no error messages to accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
adminAddr = newAdmin;
require(msg.sender == adminAddr)
require(success)
require(order > 0)
require(IERC20(inToken).transferFrom(msg.sender,address(this),tokenAm
ount))
require(IERC20(inToken).approve(toContract,tokenAmount))
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

MU - Modifiers Usage

Criticality	Minor / Informative
Location	U3.sol#L64
Status	Unresolved

Description

The contract is using repetitive statements on some methods to validate some preconditions. In Solidity, the form of preconditions is usually represented by the modifiers. Modifiers allow you to define a piece of code that can be reused across multiple functions within a contract. This can be particularly useful when you have several functions that require the same checks to be performed before executing the logic within the function.

```
require(msg.sender == adminAddr);
```

Recommendation

The team is advised to use modifiers since it is a useful tool for reducing code duplication and improving the readability of smart contracts. By using modifiers to perform these checks, it reduces the amount of code that is needed to write, which can make the smart contract more efficient and easier to maintain.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	U3.sol#L171
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.

```
PancakeRouter pancakeRouter = PancakeRouter(toContract);
address[] memory path = new address[](2);
path[0] = inToken;
path[1] = toToken;
pancakeRouter.swapExactTokensForTokens(tokenAmount,1,path,to,block.timest
amp  + 1 days);
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	U3.sol#L11,13,15,17,19,21,23,25,119,161
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.

```
event getBNB(address invitation,address indexed from, uint256 indexed
order, uint256 indexed value,string p1,string p2,string p3);
event receiveTokenAndBnb(address invitation,address from,address indexed
to, address indexed tokenAddress,uint256 tokenAmount,uint256
bnbValue,uint256 indexed order,string p1,string p2,string p3);
event getTokenAndBnb(address invitation,address from,address indexed
tokenAddress,uint256 tokenAmount,uint256 bnbValue,uint256 indexed
order,string p1,string p2,string p3);
event getTokenAndBnbAndRc(uint256 rcAmount,address from,address indexed
tokenAddress,uint256 tokenAmount,uint256 bnbValue,uint256 indexed
order,string p1,string p2,string p3);
event setBNB(address invitation,address from,address indexed to, uint256
value, uint256 indexed order,string p1,string p2,string p3);
event setToken(address invitation,address from,address indexed
tokenAddress,address indexed to, uint256 value, uint256 indexed
order,string p1,string p2,string p3);
event receiveTokenForTokenAndBnb(address invitation,address from,address
```

```
indexed to, address inToken,address toToken,uint256 indexed
tokenAmount,uint256 bnbValue,string order);
event receiveTokenForTokenAndBnbAndRc(uint256 rcAmount,address
from,address indexed to, address inToken,address toToken,uint256 indexed
tokenAmount,uint256 bnbValue,string order);
uint256 RcAmount
```

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.

L06 - Missing Events Access Control

Criticality	Minor / Informative
Location	U3.sol#L39
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
adminAddr = newAdmin
```

Recommendation

To avoid this issue, it's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	U3.sol#L39,43,64,85
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.

```
adminAddr = newAdmin
toAddr.transfer(value)
(bool success, bytes memory data) =
token.call(abi.encodeWithSelector(0xa9059cbb, to, value))
(bool success,) = token.call(abi.encodeWithSelector(0xa9059cbb,
recipients[i], amounts[i]))
```

Recommendation

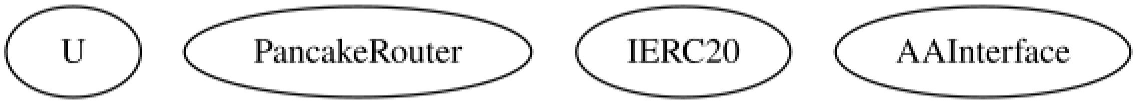
By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

Functions Analysis

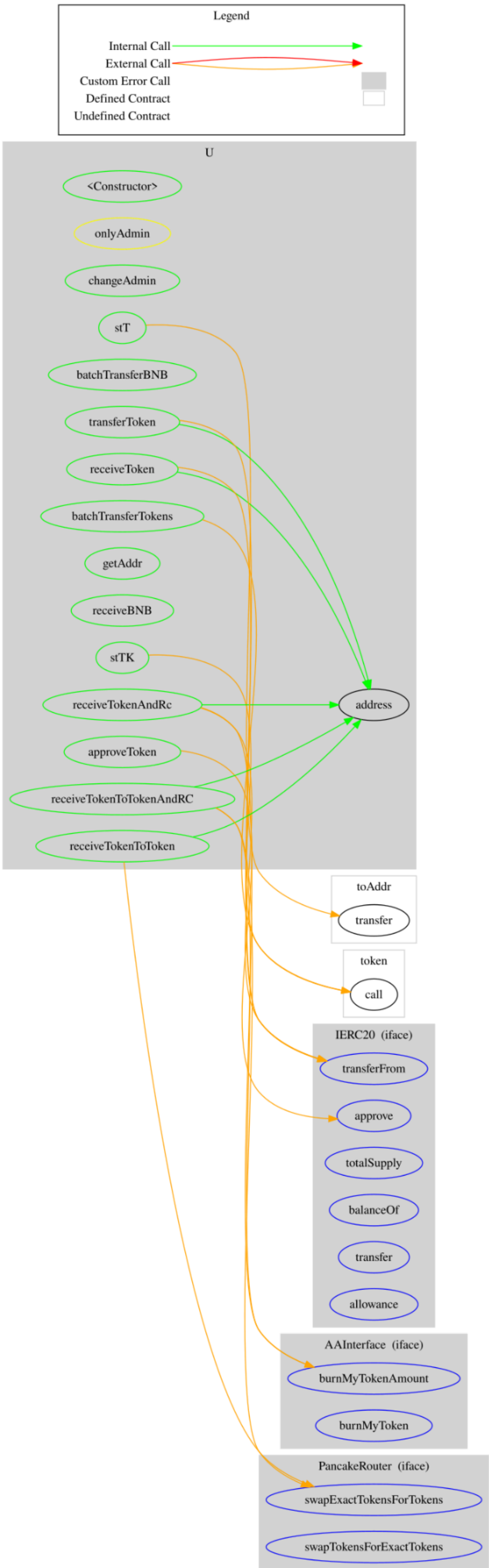
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
U	Implementation			
	<Constructor>	Public	✓	-
	changeAdmin	Public	✓	onlyAdmin
	stT	Public	Payable	onlyAdmin
	batchTransferBNB	Public	✓	onlyAdmin
	stTK	Public	✓	-
	batchTransferTokens	Public	✓	onlyAdmin
	approveToken	Public	✓	-
	getAddr	Public		-
	receiveBNB	Public	Payable	-
	receiveToken	Public	Payable	-
	receiveTokenAndRc	Public	Payable	-
	transferToken	Public	Payable	-
	receiveTokenToToken	Public	Payable	-
	receiveTokenToTokenAndRC	Public	Payable	-
PancakeRouter	Interface			
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-

IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
AAInterface	Interface			
	burnMyToken	External	✓	-
	burnMyTokenAmount	External	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
U
<Constructor>
onlyAdmin
changeAdmin
stT
batchTransferBNB
transferToken
receiveToken
batchTransferTokens
getAddr
receiveBNB
stTK
receiveTokenAndRc
address
approveToken
receiveTokenToTokenAndRC
receiveTokenToToken
toAddr
transfer
token
call
IERC20 (iface)
transferFrom
approve
totalSupply
balanceOf
transfer
allowance
AAInterface (iface)
burnMyTokenAmount
burnMyToken
PancakeRouter (iface)
swapExactTokensForTokens
swapTokensForExactTokens

Summary

NPC WEB3 GAMEFI contract implements token transfer and swap operations combined with external token burning and event-based logging. This audit investigates security issues, business logic flaws, and potential improvements related to input validation, external contract calls, and token approval handling.

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 TAC 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.

A TAC Security Company

The Cyberscope team

cyberscope.io