Cyberscope

Audit Report

ATM88

May 2023

Network	BSC
Address	0x260673f471b00d6468942570eEc9d2C99488c745
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	TPP	Token Pair Prevalidation	Unresolved
●	MVN	Misleading Variables Naming	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved

Table of Contents

Review

Contract Name	ATM88
Compiler Version	v0.8.2+commit.661d1103
Optimization	150 runs
Explorer	https://bscscan.com/address/0x260673f471b00d6468942570eec9d2c99488c745
Address	0x260673f471b00d6468942570eec9d2c99488c745
Network	BSC
Symbol	ATM88
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	18 May 2023 https://github.com/cyberscope-io/audits/blob/main/atm88/v1/audit.pdf
Corrected Phase 2	21 May 2023 https://github.com/cyberscope-io/audits/blob/main/atm88/v2/audit.pdf
Corrected Phase 3	29 May 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	75e3c97011e75627ffb36f4a2799a4e887e1a3e27ed427490e82d7b6f51cc5c9
@openzeppelin/contracts/security/Pausable.sol	5b6abc290190f46b9941c674594eee083a3fe6b92d1828d0cfefacc94d1cac9a
@openzeppelin/contracts/token/ERC20/ERC20.sol	f7831910f2ed6d32acff6431e5998baf50e4a00121303b27e974aab0ec637d79
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	c2b06bb4572bb4f84bfc5477dadc0fcc497cb66c3a1bd53480e68bedc2e154a6
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	b5a1340c5232f387b15592574f27eef78f6017bdc66542a1cea512ad4f78a0d2
@openzeppelin/contracts/utils/Address.sol	aafa8f3e41700a8353aabcdf020e06735753e6bc4b615279b43de53cfbb4f2cd
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
contracts/coin/ATM88.sol	52a30e7c650735716b24abd531e97e37d44c1a81be830e729d4b3158f315d538
contracts/interface/IFactory.sol	6d4372a8b92ad975c0f89034fea0c2e9ae072e72268c8439ebbd5eca6c8bd149
contracts/interface/IPinkAntiBot.sol	7225b28b58bf1954a21981e01b72ed6e8f88595d1ad52672eabb1011fb7aca8b
contracts/interface/IRouter.sol	1327fa034fffa54c1f44f16fea2847eaaa77fe85fa904591ee1451050534af06

Findings Breakdown

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	0	0	0	0
Minor / Informative	6	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	contracts/coin/ATM88.sol#L72
Status	Unresolved

Description

The contract owner has the authority to allow users to sell only when they hold a specific amount of Athos Meta (ATM) tokens. The maximum amount of tokens that can be set as a limit is 5,000 $ATM.

The $ATM address is:
https://www.bscscan.com/address/0xF02b31b0B6dCabd579e41A0250288608FA43F898

```
require(balanceOfATM >= minATM, "Hold ATM to sell");
```

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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Critical
Location	contracts/coin/ATM88.sol#L126
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `updateBlacklist` function.

```
function updateBlacklist(address account, bool state) external
onlyOwner {
    require(account != address(0), "Invalid address");
    blackList[account] = state;
}
```

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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

TPP - Token Pair Prevalidation

Criticality	Minor / Informative
Location	contracts/coin/ATM88.sol#L116
Status	Unresolved

Description

The variable `pair` can be any address. Additionally, the contract does not validate if a token pair exists between the following four addresses. This lack of validation can lead to unintended behavior and potential security vulnerabilities.

```
function updatePair(address _pair) external onlyOwner {
    require(_pair != address(0), "Invalid address");
    pair = _pair;
}
```

Recommendation

It is recommended to perform a prevalidation check on the contract addresses used for swapping, to ensure a smooth transaction flow within the contract. This validation should confirm that the addresses have valid pair address values associated with them.

MVN - Misleading Variables Naming

Criticality	Minor / Informative
Location	contracts/coin/ATM88.sol#L66
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.

The variable `feeSwap` stores the fee amount that is added to the `devWallet`. The variable should be named `devFee`.

```
uint256 feeSwap;
```

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.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	contracts/coin/ATM88.sol#L39,40,52
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
router
tokenATM
pinkAntiBot
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/coin/ATM88.sol#L89,93,98,103,111,116
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.

```
bool _enable
uint256 _fee
uint256 _minATM
address _address
address _pair
```

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/v0.8.17/style-guide.html#naming-convention.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	contracts/coin/ATM88.sol#L95,100,108
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.

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.

```
sellFee = _fee
buyFee = _fee
minATM = _minATM * 10 ** 18
```

Recommendation

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 with its arithmetic.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	contracts/coin/ATM88.sol#L66
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 feeSwap
```

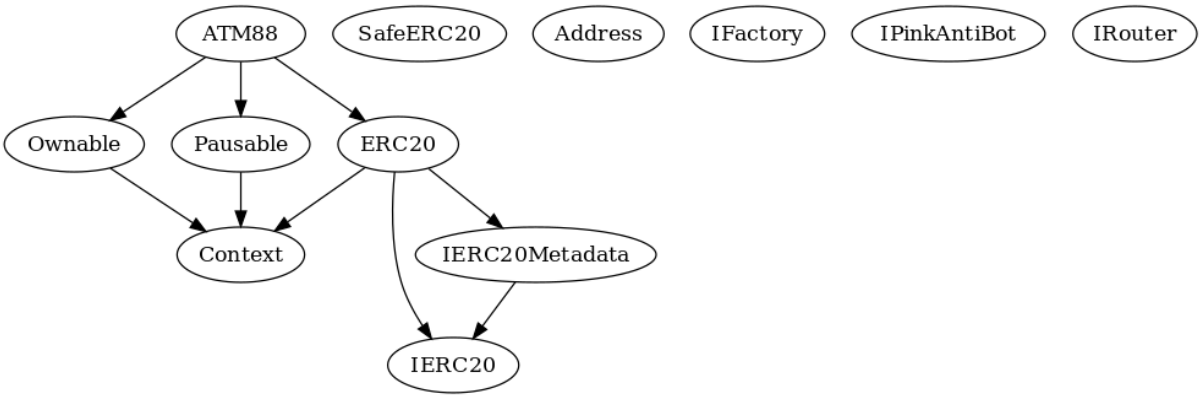
Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

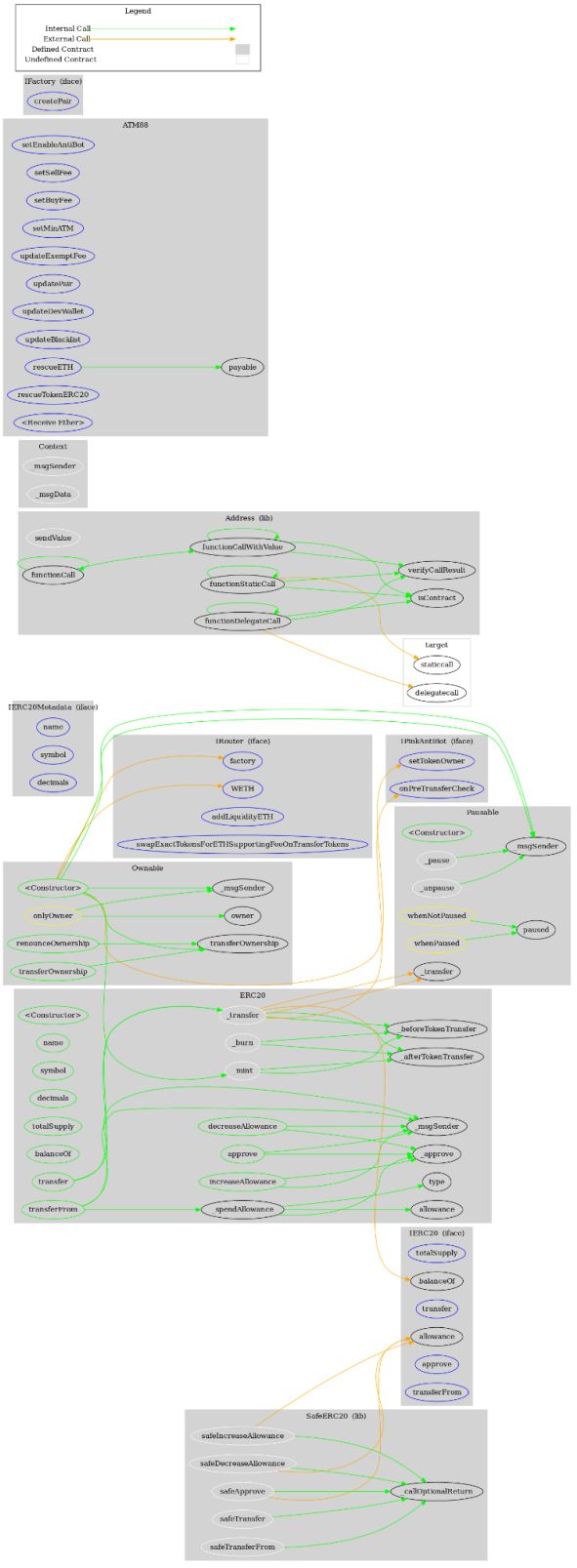
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
ATM88	Implementation	ERC20, Ownable, Pausable		
		Public	✓	ERC20
	_transfer	Internal	✓	
	setEnableAntiBot	External	✓	onlyOwner
	setSellFee	External	✓	onlyOwner
	setBuyFee	External	✓	onlyOwner
	setMinATM	External	✓	onlyOwner
	updateExemptFee	External	✓	onlyOwner
	updatePair	External	✓	onlyOwner
	updateDevWallet	External	✓	onlyOwner
	updateBlacklist	External	✓	onlyOwner
	rescueETH	External	✓	onlyOwner
	rescueTokenERC20	External	✓	onlyOwner
		External	Payable	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IFactory (iface)
createPair
ATM88
setEnableAntiBot
setSellFee
setBuyFee
setMinATM
updateExemptFee
updatePair
updateDevWallet
updateBlacklist
rescueETH
payable
rescueTokenERC20
<Receive Ether>
Context
_msgSender
_msgData
Address (lib)
sendValue
functionCall
functionCallWithValue
functionStaticCall
functionDelegateCall
verifyCallResult
isContract
Target
staticcall
delegatecall
IERC20Metadata (iface)
name
symbol
decimals
IRouter (iface)
factory
WETH
addLiquidityETH
swapExactTokensForETHSupportingFeeOnTransferTokens
IPinkAntiBot (iface)
setTokenOwner
onPreTransferCheck
Pausable
<Constructor>
_pause
_unpause
whenNotPaused
whenPaused
paused
_transfer
Ownable
<Constructor>
onlyOwner
owner
renounceOwnership
transferOwnership
_msgSender
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
transferFrom
_transfer
_burn
_mint
beforeTokenTransfer
afterTokenTransfer
decreaseAllowance
approve
increaseAllowance
spendAllowance
_msgSender
_approve
type
allowance
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
SafeERC20 (lib)
safeIncreaseAllowance
safeDecreaseAllowance
safeApprove
safeTransfer
safeTransferFrom
callOptionalReturn

Summary

ATM88 - Casino Game Series contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like stopping transactions and massively blacklist addresses. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 10% buy and sell fees.

Initial Audit, 18 May 2023

At the time of the audit report, the contract with address 0x52c9d415173489625de99dbbe4c7a79e90e9e1d0 is pointed by the following proxy address: 0xbc0640a9af9048385a241b899bb5184796ae0c98.

Corrected Phase 2, 21 May 2023

At the time of the audit report, the contract with address 0xa4790d888647ca2a287ba648a28bcad7b66ec233 is pointed by the following proxy address: 0xbc0640a9af9048385a241b899bb5184796ae0c98.

Corrected Phase 3, 29 May 2023

The proxy implementation has been removed, thus the audited address 0x260673f471b00d6468942570eEc9d2C99488c745 is the actual token.

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

https://www.cyberscope.io