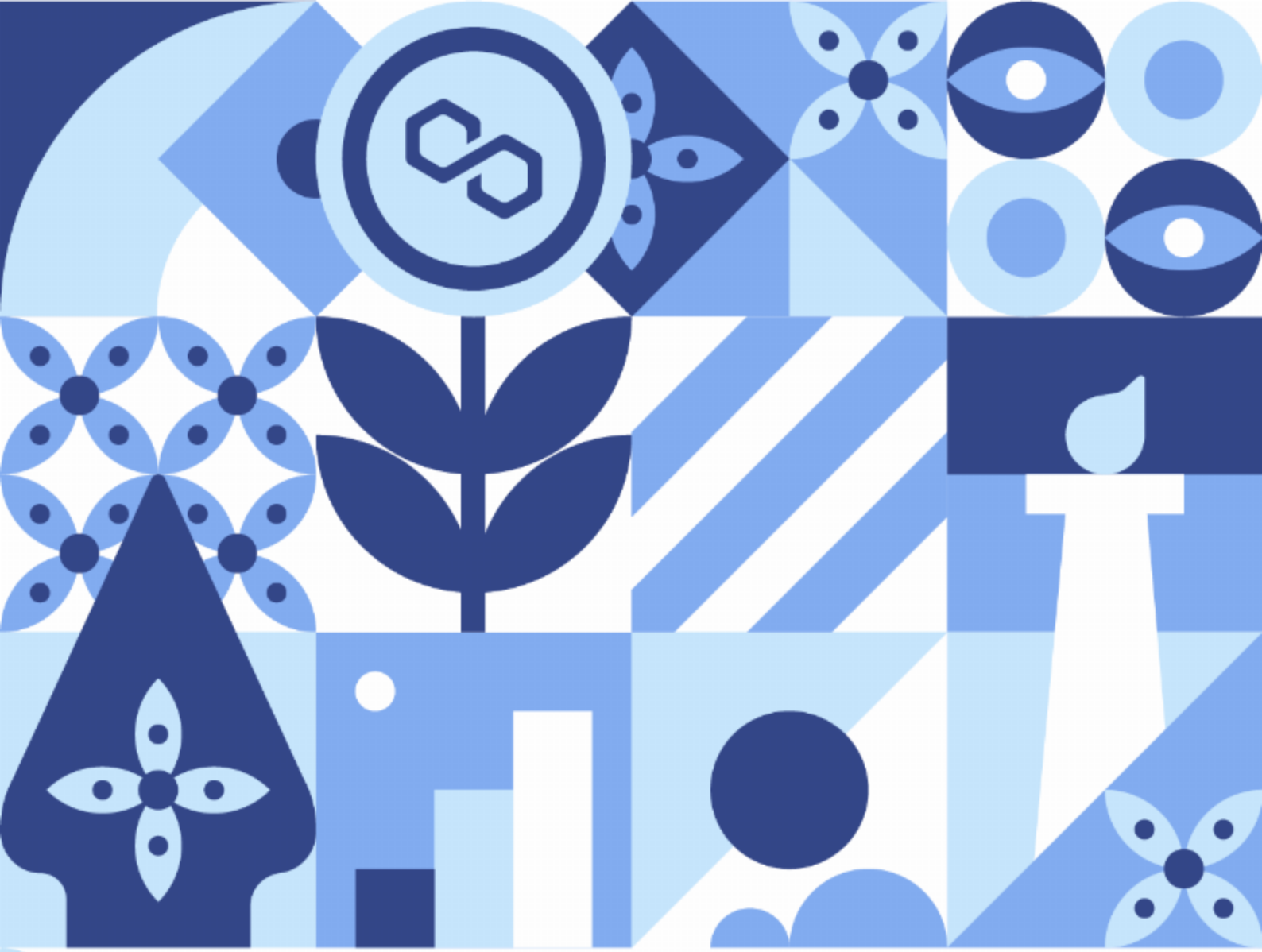

Nakamoto.Games

Smart Contract Audit Report

SYSFIXED

12 Nov 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Nakamoto.Games	NAKA	Polygon Matic

Addresses

Contract address	0x311434160d7537be358930def317afb606c0d737
Contract deployer address	0xB4675d1895d3D572c7B6A72bd0EbfbBF7ed5A4Eb

Project Website

https://www.nakamoto.games/

Codebase

https://polygonscan.com/address/0x311434160d7537be358930def317afb606c0d737#code

SUMMARY

The NAKA Token is integral to the Nakamoto Games play-to-earn ecosystem. It gives players access to any of the games within the ecosystem while also providing a system to reward the most skilful players.

Contract Summary

Documentation Quality

Nakamoto.Games provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by Nakamoto.Games with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 327, 353, 383, 419, 421, 442, 443, 468, 470, 580, 617, 1589, 1590, 1594, 1595, 1595, 1596, 1611, 1625, 1625, 1628, 1628 and 1628.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 9, 100, 126, 150, 545, 588, 626, 880, 905, 938, 1031, 1063, 1093, 1279, 1355, 1378, 1402, 1465, 1568, 1639, 1875 and 1896.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 1595, 1626, 1627, 1629 and 1629.

CONCLUSION

We have audited the Nakamoto.Games project released on November 2021 to discover issues and identify potential security vulnerabilities in Nakamoto.Games Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides a satisfactory result with some low-risk issues.

The issues found in the Nakamoto.Games smart contract codes do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some floating pragma is set. It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Thursday Nov 11 2021 07:04:58 GMT+0000 (Coordinated Universal Time)
Finished	Friday Nov 12 2021 01:16:11 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	PowerfulERC20.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "--" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged

SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 327

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
326   unchecked {
327   _approve(sender, _msgSender(), currentAllowance - amount);
328   }
329
330   return true;
331
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 353

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
352   spender,
353   _allowances[_msgSender()][spender] + addedValue
354   );
355   return true;
356   }
357
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 383

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
382   unchecked {
383   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
384   }
385
386   return true;
387
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 419

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
418   unchecked {
419   _balances[sender] = senderBalance - amount;
420   }
421   _balances[recipient] += amount;
422
423
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 421

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
420   }
421   _balances[recipient] += amount;
422
423   emit Transfer(sender, recipient, amount);
424
425
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 442

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
441
442   _totalSupply += amount;
443   _balances[account] += amount;
444   emit Transfer(address(0), account, amount);
445
446
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 443

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
442   _totalSupply += amount;
443   _balances[account] += amount;
444   emit Transfer(address(0), account, amount);
445
446   _afterTokenTransfer(address(0), account, amount);
447
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 468

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
467   unchecked {
468   _balances[account] = accountBalance - amount;
469   }
470   _totalSupply -= amount;
471
472
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 470

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
469   }
470   _totalSupply -= amount;
471
472   emit Transfer(account, address(0), amount);
473
474
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 580

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
579   unchecked {
580   _approve(account, _msgSender(), currentAllowance - amount);
581   }
582   _burn(account, amount);
583   }
584
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 617

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
616   require(
617   ERC20.totalSupply() + amount <= cap(),
618   "ERC20Capped: cap exceeded"
619   );
620   super._mint(account, amount);
621
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1589

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1588   while (temp != 0) {
1589   digits++;
1590   temp /= 10;
1591   }
1592   bytes memory buffer = new bytes(digits);
1593
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 1590

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1589   digits++;
1590   temp /= 10;
1591   }
1592   bytes memory buffer = new bytes(digits);
1593   while (value != 0) {
1594
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1594

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1593   while (value != 0) {
1594   digits -= 1;
1595   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
1596   value /= 10;
1597   }
1598
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1595

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1594   digits -= 1;
1595   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
1596   value /= 10;
1597   }
1598   return string(buffer);
1599
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1595

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1594   digits -= 1;
1595   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
1596   value /= 10;
1597   }
1598   return string(buffer);
1599
```

SWC-101 | ARITHMETIC OPERATION "/=" DISCOVERED

LINE 1596

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1595   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
1596   value /= 10;
1597   }
1598   return string(buffer);
1599   }
1600
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1611

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1610   while (temp != 0) {
1611   length++;
1612   temp >>= 8;
1613   }
1614   return toHexString(value, length);
1615
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1625

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1624   {
1625   bytes memory buffer = new bytes(2 * length + 2);
1626   buffer[0] = "0";
1627   buffer[1] = "x";
1628   for (uint256 i = 2 * length + 1; i > 1; --i) {
1629
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1625

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1624   {
1625   bytes memory buffer = new bytes(2 * length + 2);
1626   buffer[0] = "0";
1627   buffer[1] = "x";
1628   for (uint256 i = 2 * length + 1; i > 1; --i) {
1629
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1628

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1627   buffer[1] = "x";
1628   for (uint256 i = 2 * length + 1; i > 1; --i) {
1629   buffer[i] = _HEX_SYMBOLS[value & 0xf];
1630   value >>= 4;
1631   }
1632
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1628

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1627   buffer[1] = "x";
1628   for (uint256 i = 2 * length + 1; i > 1; --i) {
1629   buffer[i] = _HEX_SYMBOLS[value & 0xf];
1630   value >>= 4;
1631   }
1632
```

SWC-101 | ARITHMETIC OPERATION "--" DISCOVERED

LINE 1628

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- PowerfulERC20.sol

Locations

```
1627   buffer[1] = "x";
1628   for (uint256 i = 2 * length + 1; i > 1; --i) {
1629   buffer[i] = _HEX_SYMBOLS[value & 0xf];
1630   value >>= 4;
1631   }
1632
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 9

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
8
9   pragma solidity ^0.8.0;
10
11   /**
12   * @dev Interface of the ERC20 standard as defined in the EIP.
13
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 100

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
99
100   pragma solidity ^0.8.0;
101
102   /**
103   * @dev Interface for the optional metadata functions from the ERC20 standard.
104
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 126

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
125
126   pragma solidity ^0.8.0;
127
128   /**
129   * @dev Provides information about the current execution context, including the
130
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 150

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
149
150   pragma solidity ^0.8.0;
151
152   /**
153   * @dev Implementation of the {IERC20} interface.
154
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 545

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
544
545   pragma solidity ^0.8.0;
546
547   /**
548   * @dev Extension of {ERC20} that allows token holders to destroy both their own
549
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 588

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
587
588   pragma solidity ^0.8.0;
589
590   /**
591   * @dev Extension of {ERC20} that adds a cap to the supply of tokens.
592
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 626

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
625
626   pragma solidity ^0.8.0;
627
628   /**
629   * @dev Collection of functions related to the address type
630
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 880

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
879
880   pragma solidity ^0.8.0;
881
882   /**
883   * @dev Interface of the ERC165 standard, as defined in the
884
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 905

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
904
905   pragma solidity ^0.8.0;
906
907   /**
908   * @dev Implementation of the {IERC165} interface.
909
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 938

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
937
938   pragma solidity ^0.8.0;
939
940   /**
941   * @title IERC1363 Interface
942
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1031

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1030
1031   pragma solidity ^0.8.0;
1032
1033   /**
1034   * @title IERC1363Receiver Interface
1035
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1063

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1062
1063   pragma solidity ^0.8.0;
1064
1065   /**
1066   * @title IERC1363Spender Interface
1067
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1093

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1092
1093   pragma solidity ^0.8.0;
1094
1095   /**
1096   * @title ERC1363
1097
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1279

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1278
1279   pragma solidity ^0.8.0;
1280
1281   /**
1282   * @dev Contract module which provides a basic access control mechanism, where
1283
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1355

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1354
1355   pragma solidity ^0.8.0;
1356
1357   /**
1358   * @title TokenRecover
1359
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1378

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1377
1378   pragma solidity ^0.8.0;
1379
1380   /**
1381   * @title ERC20Decimals
1382
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1402

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1401
1402   pragma solidity ^0.8.0;
1403
1404   /**
1405   * @title ERC20Mintable
1406
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1465

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1464
1465   pragma solidity ^0.8.0;
1466
1467   /**
1468   * @dev External interface of AccessControl declared to support ERC165 detection.
1469
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1568

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1567
1568   pragma solidity ^0.8.0;
1569
1570   /**
1571   * @dev String operations.
1572
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1639

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1638
1639   pragma solidity ^0.8.0;
1640
1641   /**
1642   * @dev Contract module that allows children to implement role-based access
1643
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1875

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1874
1875   pragma solidity ^0.8.0;
1876
1877   contract Roles is AccessControl {
1878   bytes32 public constant MINTER_ROLE = keccak256("MINTER");
1879
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1896

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- PowerfulERC20.sol

Locations

```
1895
1896   pragma solidity ^0.8.0;
1897
1898   /**
1899   * @title PowerfulERC20
1900
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1595

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- PowerfulERC20.sol

Locations

```
1594   digits -= 1;
1595   buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
1596   value /= 10;
1597   }
1598   return string(buffer);
1599
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1626

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- PowerfulERC20.sol

Locations

```
1625   bytes memory buffer = new bytes(2 * length + 2);
1626   buffer[0] = "0";
1627   buffer[1] = "x";
1628   for (uint256 i = 2 * length + 1; i > 1; --i) {
1629   buffer[i] = _HEX_SYMBOLS[value & 0xf];
1630
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1627

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- PowerfulERC20.sol

Locations

```
1626   buffer[0] = "0";
1627   buffer[1] = "x";
1628   for (uint256 i = 2 * length + 1; i > 1; --i) {
1629   buffer[i] = _HEX_SYMBOLS[value & 0xf];
1630   value >>= 4;
1631
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1629

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- PowerfulERC20.sol

Locations

```
1628   for (uint256 i = 2 * length + 1; i > 1; --i) {
1629   buffer[i] = _HEX_SYMBOLS[value & 0xf];
1630   value >>= 4;
1631   }
1632   require(value == 0, "Strings: hex length insufficient");
1633
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1629

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- PowerfulERC20.sol

Locations

```
1628   for (uint256 i = 2 * length + 1; i > 1; --i) {
1629   buffer[i] = _HEX_SYMBOLS[value & 0xf];
1630   value >>= 4;
1631   }
1632   require(value == 0, "Strings: hex length insufficient");
1633
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. 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 Sysfixed's prior written consent in each instance.

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 considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report 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 is a limited report on our findings based on our analysis, in accordance with good industry practice as of 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

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 Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.