Cyberverseland

Smart Contract Audit Report

SYSFIXED

23 Jan 2023

TABLE OF CONTENTS

Audited Details

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

Conclusion

Audit Results

Smart Contract Analysis

- Detected Vulnerabilities

Disclaimer

About Us

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Cyberverseland	CYBERVERSE	Binance Smart Chain

Addresses

Contract address	0x0911BBfF1F00E94a1D3FcFa331E890F05337CD4B
Contract deployer address	0x7271ed7709d8bB6f83766b76Db276b50e057d2b9

Project Website

https://www.cyberverseland.com/

Codebase

https://bscscan.com/address/0x0911BBfF1F00E94a1D3FcFa331E890F05337CD4B#code

SUMMARY

Cyberverseland - The bridge between physical and virtual worlds within the decentralized and futuristic virtual world. Your imagination is the only limitation in Cyberverseland!

Contract Summary

Documentation Quality

Cyberverseland 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 Cyberverseland 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 217, 242, 271, 303, 304, 450, 482, 522, 535, 550, 582, 590, 594, 602, 610, 614, 635, 636, 638, 644, 645, 646, 647, 654, 705, 714, 725 and 756.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 7.
- SWC-110 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 664, 665 and 757.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 582 and 734.

CONCLUSION

We have audited the Cyberverseland project which has released on January 2023 to discover issues and identify potential security vulnerabilities in Cyberverseland Project. This process is used to find technical issues and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with low-risk issues.

The most common issue found in writing code on contracts that do not pose a big risk, writing on contracts is close to the standard of writing contracts in general. The low-level issues found are some arithmetic operation issues, a floating pragma is set, weak sources of randomness and out of bounds array access which the index access expression can cause an exception in case of use of an invalid array index value.

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
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
Assert Violation	SWC-110	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 Caller	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
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.	ISSUE FOUND
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

SMART CONTRACT ANALYSIS

Started	Sunday Jan 22 2023 16:33:08 GMT+0000 (Coordinated Universal Time)
Finished	Monday Jan 23 2023 22:34:52 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Cyberverseland.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-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-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-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 217

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
216   );
217   _approve(sender, _msgSender(), currentAllowance - amount);
218
219   return true;
220   }
221
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 242

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
241   spender,
242   _allowances[_msgSender()][spender] + addedValue
243   );
244   return true;
245   }
246
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 271

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
270   );
271   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
272
273   return true;
274   }
275
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 303

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
302   );
303   _balances[sender] = senderBalance - amount;
304   _balances[recipient] += amount;
305
306   emit Transfer(sender, recipient, amount);
307
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 304

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
303   _balances[sender] = senderBalance - amount;
304   _balances[recipient] += amount;
305
306   emit Transfer(sender, recipient, amount);
307   }
308
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 450

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
449
450   uint256 public tokenLiquidityThreshold = 1e6 * 10**18;
451
452   uint256 public genesis_block;
453   uint256 private deadline = 0;
454
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 482

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
481   constructor() BEP20("Cyberverseland", "CYBERVERSE") {
482   _tokengeneration(msg.sender, 1e9 * 10**decimals());
483   exemptFee[msg.sender] = true;
484
485   IRouter _router = IRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
486
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 522

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
521   );
522   _approve(sender, _msgSender(), currentAllowance - amount);
523
524   return true;
525   }
526
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 535

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
534   spender,
535   _allowances[_msgSender()][spender] + addedValue
536   );
537   return true;
538   }
539
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 550

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
549   );
550   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
551
552   return true;
553   }
554
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 582

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
581   !exemptFee[recipient] &&
582   block.number < genesis_block + deadline;
583
584   //set fee to zero if fees in contract are handled or exempted
585   if (_interlock || exemptFee[sender] || exemptFee[recipient])
586
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 590

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
589   else if (recipient == pair && !useLaunchFee) {
590   feeswap = sellTaxes.liquidity + sellTaxes.marketing;
591   feesum = feeswap;
592   currentTaxes = sellTaxes;
593   } else if (!useLaunchFee) {
594
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 594

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
593   } else if (!useLaunchFee) {
594   feeswap = taxes.liquidity + taxes.marketing;
595   feesum = feeswap;
596   currentTaxes = taxes;
597   } else if (useLaunchFee) {
598
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 602

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
601
602   fee = (amount * feesum) / 100;
603
604   //send fees if threshold has been reached
605   //don't do this on buys, breaks swap
606
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 610

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
609   //rest to recipient
610   super._transfer(sender, recipient, amount - fee);
611   if (fee > 0) {
612   //send the fee to the contract
613   if (feeswap > 0) {
614
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 614

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
613   if (feeswap > 0) {
614   uint256 feeAmount = (amount * feeswap) / 100;
615   super._transfer(sender, address(this), feeAmount);
616   }
617   }
618
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 635

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
634   // Split the contract balance into halves
635   uint256 denominator = feeswap * 2;
636   uint256 tokensToAddLiquidityWith = (contractBalance *
637   swapTaxes.liquidity) / denominator;
638   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
639
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 636

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
635   uint256 denominator = feeswap * 2;
636   uint256 tokensToAddLiquidityWith = (contractBalance *
637   swapTaxes.liquidity) / denominator;
638   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
639
640
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 638

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
637   swapTaxes.liquidity) / denominator;
638   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
639
640   uint256 initialBalance = address(this).balance;
641
642
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 644

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
643
644   uint256 deltaBalance = address(this).balance - initialBalance;
645   uint256 unitBalance = deltaBalance /
646   (denominator - swapTaxes.liquidity);
647   uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
648
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 645

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
644   uint256 deltaBalance = address(this).balance - initialBalance;
645   uint256 unitBalance = deltaBalance /
646   (denominator - swapTaxes.liquidity);
647   uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
648
649
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 646

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
645   uint256 unitBalance = deltaBalance /
646   (denominator - swapTaxes.liquidity);
647   uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
648
649   if (ethToAddLiquidityWith > 0) {
650
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 647

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
646   (denominator - swapTaxes.liquidity);
647   uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
648
649   if (ethToAddLiquidityWith > 0) {
650   // Add liquidity to pancake
651
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 654

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
653
654   uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
655   if (marketingAmt > 0) {
656   payable(marketingWallet).sendValue(marketingAmt);
657   }
658
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 705

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
704   );
705   tokenLiquidityThreshold = new_amount * 10**decimals();
706   }
707
708   function SetBuyTaxes(uint256 _marketing, uint256 _liquidity)
709
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 714

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
713   require(
714   (_marketing + _liquidity) <= 1,
715   "Must keep fees at 1% or less"
716   );
717   }
718
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 725

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
724   require(
725   (_marketing + _liquidity) <= 1,
726   "Must keep fees at 1% or less"
727   );
728   }
729
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 756

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
755   {
756   for (uint256 i = 0; i < accounts.length; i++) {
757   exemptFee[accounts[i]] = state;
758   }
759   }
760
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.17"". 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

- Cyberverseland.sol

Locations

```
6
7   pragma solidity ^0.8.17;
8
9   abstract contract Context {
10   function _msgSender() internal view virtual returns (address) {
11
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 664

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
663   address[] memory path = new address[](2);
664   path[0] = address(this);
665   path[1] = router.WETH();
666
667   _approve(address(this), address(router), tokenAmount);
668
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 665

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
664   path[0] = address(this);
665   path[1] = router.WETH();
666
667   _approve(address(this), address(router), tokenAmount);
668
669
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 757

low SEVERITY

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

Source File

- Cyberverseland.sol

Locations

```
756   for (uint256 i = 0; i < accounts.length; i++) {
757   exemptFee[accounts[i]] = state;
758   }
759   }
760
761
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 582

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- Cyberverseland.sol

Locations

```
581   !exemptFee[recipient] &&
582   block.number < genesis_block + deadline;
583
584   //set fee to zero if fees in contract are handled or exempted
585   if (_interlock || exemptFee[sender] || exemptFee[recipient])
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 734

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- Cyberverseland.sol

Locations

```
733   providingLiquidity = true;
734   genesis_block = block.number;
735   }
736
737   function updatedeadline(uint256 _deadline) external onlyOwner {
```

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.