

# GAMEE Smart Contract Audit Report



09 Sep 2021



# **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	
GAMEE	GMEE	Polygon Matic	

### Addresses

Contract address	0xcf32822ff397ef82425153a9dcb726e5ff61dca7	
Contract deployer address	0x566B788dE3969d35711f6A0Fc543b00ed7260CDc	

### Project Website

https://www.gamee.com/token

### Codebase

https://polygonscan.com/address/0xcf32822ff397ef82425153a9dcb726e5ff61dca7#code



# SUMMARY

The GMEE Token is a utility token designed to be the currency of purchase, utility, and reward in supported play-to-earn games provided on the GAMEE casual gaming platform.

### Contract Summary

#### **Documentation Quality**

GAMEE 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 GAMEE with the discovery of several low issues.

#### Test Coverage

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

### Audit Findings Summary

- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 14, 53, 128, 149, 181, 238, 308, 329, 423, 459, 497, 548, 590, 608, 677, 704, 1186, 1195, 1213, 1241, 1307, 1326, 1334, 1342 and 1399.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1372 and 1388.



# CONCLUSION

We have audited the GAMEE project released in September 2021 to discover issues and identify potential security vulnerabilities in GAMEE 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 satisfactory results with low-risk issues.

The issues in the GAMEE smart contract code 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, and "tx.origin" as a part of authorization control. Specifying a fixed compiler version is recommended 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. The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to act on their behalf. It is recommended to use "msg.sender" instead.



# 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.	
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.	PASS
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.	
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.	ISSUE FOUND
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.	
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.	
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.	
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	



## **SMART CONTRACT ANALYSIS**

Started	Wednesday Sep 08 2021 23:06:55 GMT+0000 (Coordinated Universal Time)			
Finished	Thursday Sep 09 2021 15:43:55 GMT+0000 (Coordinated Universal Time)			
Mode	Standard			
Main Source File	PolygonGMEE.sol			

### Detected Issues

ID	Title	Severity	Status
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

### SYSFIXED

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-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged



LINE 14

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
13
14 pragma solidity >=0.7.6 <0.8.0;
15
16 /**
17 * @dev Upgrades the address type to check if it is a contract.
18</pre>
```





LINE 53

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
52
53 pragma solidity >=0.7.6 <0.8.0;
54
55 /**
56 * @title ERC20Wrapper
57</pre>
```



**LINE 128** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
127
128 pragma solidity >=0.7.6 <0.8.0;
129
130 /*
131 * Provides information about the current execution context, including the
132</pre>
```





**LINE 149** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
148
149 pragma solidity >=0.7.6 <0.8.0;
150
151 /**
152 * @title ERC-173 Contract Ownership Standard
153</pre>
```



LINE 181

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
180
181 pragma solidity >=0.7.6 <0.8.0;
182
183
184 /**
185</pre>
```



**LINE 238** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

#### Locations

237 238 pragma solidity >=0.7.6 <0.8.0; 239 240 241 242



**LINE 308** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
307
308 pragma solidity >=0.7.6 <0.8.0;
309
310 /**
311 * @dev Interface of the ERC165 standard, as defined in the
312</pre>
```



**LINE 329** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
328
329 pragma solidity >=0.7.6 <0.8.0;
330
331 /**
332 * @title ERC20 Token Standard, basic interface
333</pre>
```



**LINE 423** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
422
423 pragma solidity >=0.7.6 <0.8.0;
424
425 /**
426 * @title ERC20 Token Standard, optional extension: Detailed
427</pre>
```





**LINE 459** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
458
459 pragma solidity >=0.7.6 <0.8.0;
460
461 /**
462 * @title ERC20 Token Standard, optional extension: Allowance
463</pre>
```



LINE 497

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
496
497 pragma solidity >=0.7.6 <0.8.0;
498
499 /**
500 * @title ERC20 Token Standard, optional extension: Safe Transfers
501</pre>
```





**LINE 548** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
547
548 pragma solidity >=0.7.6 <0.8.0;
549
550 /**
551 * @title ERC20 Token Standard, optional extension: Multi Transfers
552</pre>
```



**LINE 590** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
589
590 pragma solidity >=0.7.6 <0.8.0;
591
592 /**
593 * @title ERC20 Token Standard, ERC1046 optional extension: Metadata
594</pre>
```





**LINE 608** 

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
607
608 pragma solidity >=0.7.6 <0.8.0;
609
610 /**
611 * @title ERC20 Token Standard, ERC2612 optional extension: permit - 712-signed
approvals
612</pre>
```





LINE 677

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
676
677 pragma solidity >=0.7.6 <0.8.0;
678
679 /**
680 * @title ERC20 Token Standard, Receiver
681</pre>
```



**LINE** 704

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

#### Locations

703
704 pragma solidity >=0.7.6 <0.8.0;
705
706
707
708</pre>



LINE 1186

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

#### Locations

1185
1186 pragma solidity >=0.7.6 <0.8.0;
1187
1188 interface IChildToken {
1189 function deposit(address user, bytes calldata depositData) external;
1190</pre>



LINE 1195

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
1194
1195 pragma solidity >=0.7.6 <0.8.0;
1196
1197
1198 abstract contract ERC20Receiver is IERC20Receiver, IERC165 {
1199</pre>
```



LINE 1213

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
1212
1213 pragma solidity >=0.7.6 <0.8.0;
1214
1215
1216 /**
1217</pre>
```



LINE 1241

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

```
1240
1241 pragma solidity >=0.7.6 <0.8.0;
1242
1243
1244 abstract contract ChildERC20 is ERC20, ChildERC20Base {
1245</pre>
```



LINE 1307

#### **Iow SEVERITY**

The current pragma Solidity directive is ""^0.7.0"". 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

- PolygonGMEE.sol

```
1306 // File ethereum-universal-
forwarder/src/solc_0.7/ERC2771/UsingAppendedCallData.sol@v0.1.4
1307 pragma solidity ^0.7.0;
1308
1309 abstract contract UsingAppendedCallData {
1310 function _lastAppendedDataAsSender() internal pure virtual returns (address
payable sender) {
1311
```





LINE 1326

#### **Iow SEVERITY**

The current pragma Solidity directive is ""^0.7.0"". 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

- PolygonGMEE.sol

```
1325 // File ethereum-universal-forwarder/src/solc_0.7/ERC2771/IERC2771.sol@v0.1.4
1326 pragma solidity ^0.7.0;
1327
1328 interface IERC2771 {
1329 function isTrustedForwarder(address forwarder) external view returns (bool);
1330
```





LINE 1334

#### **Iow SEVERITY**

The current pragma Solidity directive is ""^0.7.0"". 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

- PolygonGMEE.sol

```
1333 // File ethereum-universal-
forwarder/src/solc_0.7/ERC2771/IForwarderRegistry.sol@v0.1.4
1334 pragma solidity ^0.7.0;
1335
1336 interface IForwarderRegistry {
1337 function isForwarderFor(address, address) external view returns (bool);
1338
```



LINE 1342

#### **Iow SEVERITY**

The current pragma Solidity directive is ""^0.7.0"". 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

- PolygonGMEE.sol

#### Locations

1341 // File ethereum-universalforwarder/src/solc\_0.7/ERC2771/UsingUniversalForwarding.sol@v0.1.4 1342 pragma solidity ^0.7.0; 1343 1344 1345 1346



LINE 1399

#### **Iow SEVERITY**

The current pragma Solidity directive is "">=0.7.6<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

- PolygonGMEE.sol

#### Locations

1398
1399 pragma solidity >=0.7.6 <0.8.0;
1400 contract PolygonGMEE is Recoverable, UsingUniversalForwarding, ChildERC20 {
1401 using ERC20Wrapper for IWrappedERC20;
1402
1403</pre>



# SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 1372

#### **Iow SEVERITY**

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

#### Source File

- PolygonGMEE.sol

```
1371 // solhint-disable-next-line avoid-tx-origin
1372 if (msgSender != tx.origin && _forwarderRegistry.isForwarderFor(sender,
msgSender)) {
1373 return sender;
1374 }
1375
1376
```





# SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 1388

#### **Iow SEVERITY**

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

#### Source File

- PolygonGMEE.sol

```
1387 // solhint-disable-next-line avoid-tx-origin
1388 if (msgSender != tx.origin &&
_forwarderRegistry.isForwarderFor(_lastAppendedDataAsSender(), msgSender)) {
1389 return _msgDataAssuming20BytesAppendedData();
1390 }
1391 return msg.data;
1392
```





## 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.