Smart Contract Audit Report

DeFiGeek-Community/yamawake
2024/03/07

目次

1. イントロダクション・・・ 2

└ 1.1 監査対象・・・ 2

2. 監査手法・・・ 3

└ 3.1 スマートコントラクト監査手法概要・・・ 3

3. 脆弱性評価基準・・・ 4

4. 監査結果概要・・・ 5～12

├ 4.1 マニュアルレビュー・・・ 5

├ 4.2 SWC レジストリ・・・ 6～7

├ 4.3 検出脆弱性一覧・・・ 8

├ 4.4 脆弱性詳細・・・ 9～10

└ 4.5 静的解析詳細・・・ 11～14

5. 免責事項・・・ 15

1.イントロダクション

サミーデジタルセキュリティ株式会社は、合同会社 SG 様との契約に基づき「yamawake」のスマートコントラクト監査を実施します。

スマートコントラクト実装における設計時の潜在的な問題、実装時の不備、コードと設計ドキュメントの不一致等を体系的なアプローチで検出し評価を致します。

検出された脆弱性に関してはシステムのセキュリティに影響を及ぼす可能性がある為、指摘事項についてはリスクレベルや影響度に応じた適切な対策を実施することを推奨します。

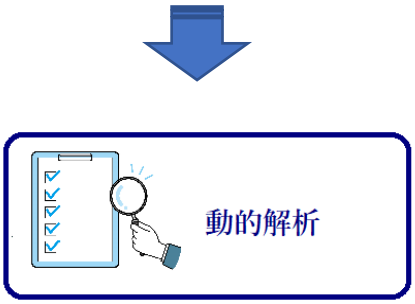
1.1 監査対象

監査対象	yamawake
対象コントラクト	GitHub - DeFiGeek-Community/yamawake
ブランチ	bc813ed
監査完了日	2024/03/07

2.監査手法

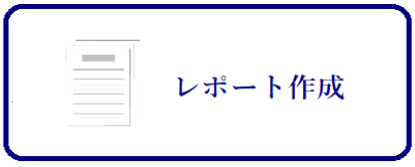
2.1 スマートコントラクト監査手法概要

コントラクトの目的や概要を把握し、
設計上のリスクを分析します。



手動による、正常系/異常系テスト及び
SWC、ConsensysBestPractice を基にした定型テストを行います。

ツール診断(Slither など)を用いて、
コード分析を行います。



発見された脆弱性の説明、関連するリスク、及び推奨対策について記載した報告書を作成いたします。

3.脆弱性評価基準

本監査は、CWE(Common Weakness Enumeration)、SWC(Smart Contact Weakness Classification and Test Cases)、Ethereum Smart Contract security Best Practice 等の脆弱性基準を基にして、弊社独自の脆弱性評価基準を採用しランク付けを行っております。

重大度	説明
High	コントラクトの機能停止やコントラクトの破損に繋がる脆弱性、及びコントラクトやユーザーの資産の損失、データの損失、または資産やデータ不正な操作に繋がる恐れのある脆弱性が該当します。 メインネットに移行する前に修正する必要があります。
Medium	スマートコントラクトによる、サービスの提供や可用性に影響を及ぼす可能性がある脆弱性が該当します。 このレベルの脆弱性は潜在的な問題を引き起こす危険性がある為、修正することを推奨致します。
Low	こちらの脆弱性はサービスの提供や可用性に影響を及ぼすことはありませんが、将来的に問題となり得る可能性がある脆弱性が含まれます。 このレベルの脆弱性は、可能な限り修正されることを推奨致します。
Info	こちらで指摘されている項目は脆弱性ではありませんが、コードの品質向上やセキュリティ向上の為の推奨事項等が含まれております。

4.監査結果概要

4.1 動的解析（マニュアルレビュー）

対象のスマートコントラクトにおいて、以下を重点にマニュアルレビューを行っています。

・コード全体の品質

・Best Practice に沿っているか

・ドキュメント、コメントから期待される動作とコードのロジックが一致しているか

※適宜必要に応じて fuzz test 等も行っております。

```
function testAddScore(uint256 x↑) public {
    factory.addTemplate(templateName, implementionAddr, initializeSignature, transferSignature);

    uint256 value = 100;

    bytes memory args = abi.encode(value);
    address target_ = address(0x123);

    address deployAuctionAddr = factory.deployAuction(templateName,args);

    factory.auctions(deployAuctionAddr);

    vm.prank(deployAuctionAddr);

    _distributor.addScore(target_,x↑);
```

```
[PASS] testAddScore(uint256) (runs: 256, μ: 165844, ~: 166777)
```

4.2 SWC レジストリ

以下はスマートコントラクトの弱点分類(SmartContract Weakness Classification and Test Cases)に登録されております脆弱性のリストとなっており、コード内にリスト内の脆弱性が含まれていないかの確認となります。

ID	概要	実施
SWC-100	Function Default Visibility	☑
SWC-101	Integer Overflow and Underflow	☑
SWC-102	Outdated Compiler Version	☑
SWC-103	Floating Pragma	☑
SWC-104	Unchecked Call Return Value	☑
SWC-105	Unprotected Ether Withdrawal	☑
SWC-106	Unprotected SELFDESTRUCT Instruction	☑
SWC-107	Reentrancy	☑
SWC-108	State Variable Default Visibility	☑
SWC-109	Uninitialized Storage Pointer	☑
SWC-110	Assert Violation	☑
SWC-111	Use of Deprecated Solidity Functions	☑
SWC-112	Delegatecall to Untrusted Callee	☑
SWC-113	DoS with Failed Call	☑
SWC-114	Transaction Order Dependence	☑
SWC-115	Authorization through tx.origin	☑
SWC-116	Block values as a proxy for time	☑
SWC-117	Signature Malleability	☑
SWC-118	Incorrect Constructor Name	☑
SWC-119	Shadowing State Variables	☑
SWC-120	Weak Sources of Randomness from Chain Attributes	☑
SWC-121	Missing Protection against Signature Replay Attacks	☑
SWC-122	Lack of Proper Signature Verification	☑
SWC-123	Requirement Violation	☑
SWC-124	Write to Arbitrary Storage Location	☑

SWC-125	Incorrect Inheritance Order	☑
SWC-126	Insufficient Gas Griefing	☑
SWC-127	Arbitrary Jump with Function Type Variable	☑
SWC-128	DoS With Block Gas Limit	☑
SWC-129	Typographical Error	☑
SWC-130	Right-To-Left-Override control character (U+202E)	☑
SWC-131	Presence of unused variables	☑
SWC-132	Unexpected Ether balance	☑
SWC-133	Hash Collisions With Multiple Variable Length Arguments	☑
SWC-134	Message call with hardcoded gas amount	☑
SWC-135	Code With No Effects	☑
SWC-136	Unencrypted Private Data On-Chain	☑

4.3 検出脆弱性一覧

当監査において検出された脆弱性/指摘項目は下記の表の通りです。

監査対象	リスクレベル及び検出件数			
	High	Medium	Low	Info
全般	0	0	0	2
BaseTemplate.sol	0	0	0	0
Distributor.sol	0	0	0	0
Factory.sol	0	0	0	1
FeePool.sol	0	0	0	0
SampleToken.sol	0	0	0	0
TemplateV1.sol	0	0	0	0
TemplateV1WithCreationFee.sol	0	0	0	0
YMWK.sol	0	0	0	0
合計	0	0	0	3

No	脆弱性詳細	重大度	ステータス
A.1	Solidity Compiler のバージョン	Info	acknowledged
A.2	コード内の適切なコメントの不足	Info	fixed
A.3	Memory の参照ズレの懸念	Info	acknowledged

4.4 脆弱性詳細

A.1 solidity compiler のバージョン

▼概要

コントラクトコードは最もテストされたものと同じコンパイラのバージョンでデプロイされることが推奨されます。

pragma を固定することによって、未知のバグリスクを抑えることができます。

pragma solidity ^0.8.18;と指定されていますが、pragma を最新バージョンに固定されることを推奨致します。

A.2 コード内の適切なコメントの不足

▼概要

このスマートコントラクトにおいて、以下の点で NatSpec(Natural Specification)が不足しています。

1. コントラクトの説明:コントラクトの目的や機能についての説明
2. 関数の説明:各関数の目的や引数、戻り値についての説明
3. イベントの説明:イベントの目的や発生条件についての説明
4. 修飾子の説明:修飾子がどのような条件をチェックしているのか、どのような役割を果たしているかの説明　例:onlyAuction 修飾子

NatSpec を追加しドキュメンテーションを充実させることで、コードの可読性や理解度を上げ、保守性を向上させることができます。

A.3 memory 参照のズレの懸念

▼概要

インラインアセンブリ内でのメモリ操作での参照のズレによるクローンされるアドレスが予想されるアドレスと一致しない場合の懸念がございます。
実際に発生するケースはほぼございませんが、セキュリティを重視する場合は、ソルトの生成方法や実装アドレスの正確さに注意した上で実装を検討されることを推奨致します。

▼該当コード(Factory.sol>Line:103)

```
/// @dev Deploy implemention's minimal proxy by create2
ftrace | funcSig
function _createClone(    Factory._createClone(address) (src/Factory.sol#103-124) uses assembly INLINE ASM (src/Factory.sol#109-122)
    address implementation_
) internal returns (address result) {
    nonce += 1;
    bytes32 salt = keccak256(abi.encodePacked(implementation_, nonce));
    // OpenZeppelin Contracts (last updated v4.8.0) (proxy/Clones.sol)
    assembly {
        mstore(
            0x00,
            or(
                shr(0xe8, shl(0x60, implementation_)),
                0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000
            )
        )
        mstore(
            0x20,
            or(shl(0x78, implementation_), 0x5af43d82803e903d91602b57fd5bf3)
        )
        result := create2(0, 0x09, 0x37, salt)
    }
    require(result != address(0), "ERC1167: create2 failed");
}
```

4.5 静的解析詳細

以下はコード解析ツールによるコード分析結果となります。

① Slither

▼HIGH リスク検出一覧(10 件)

```
TemplateV1.withdrawRaisedETH() (src/TemplateV1.sol#164-190) sends eth to arbitrary user
        Dangerous calls:
        - address(owner).transfer(address(this).balance) (src/TemplateV1.sol#189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbi
trary-destinations

SampleToken is re-used:
        - SampleToken (src/SampleToken.sol#11-29)
        - SampleToken (src/sampleToken.sol#11-29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused

StdCheats.vm (lib/forge-std/src/StdCheats.sol#480) shadows:
        - StdCheatsSafe.vm (lib/forge-std/src/StdCheats.sol#10)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

Distributor.claim(address) (src/Distributor.sol#33-47) ignores return value by token.transfer(target_,_sco
re) (src/Distributor.sol#45)
FeePool.withdrawToken(address,address[]) (src/FeePool.sol#24-38) ignores return value by IERC20(token_[i])
.transfer(to_,amount) (src/FeePool.sol#32)
TemplateV1.claim(address,address) (src/TemplateV1.sol#114-148) ignores return value by erc20onsale.transfe
r(recipient,erc20allocation) (src/TemplateV1.sol#135)
TemplateV1.withdrawERC20Onsale() (src/TemplateV1.sol#198-205) ignores return value by erc20onsale.transfer
(owner,allocatedAmount) (src/TemplateV1.sol#204)
TemplateV1.initializeTransfer(address,uint256,address) (src/TemplateV1.sol#207-213) ignores return value b
y IERC20(token_).transferFrom(msg.sender,to_,amount_) (src/TemplateV1.sol#212)
TemplateV1WithCreationFee.claim(address,address) (src/TemplateV1WithCreationFee.sol#115-149) ignores retur
n value by erc20onsale.transfer(recipient,erc20allocation) (src/TemplateV1WithCreationFee.sol#136)
TemplateV1WithCreationFee.withdrawERC20Onsale() (src/TemplateV1WithCreationFee.sol#202-209) ignores return
 value by erc20onsale.transfer(owner,allocatedAmount) (src/TemplateV1WithCreationFee.sol#208)
TemplateV1WithCreationFee.initializeTransfer(address,uint256,address) (src/TemplateV1WithCreationFee.sol#2
11-217) ignores return value by IERC20(token_).transferFrom(msg.sender,to_,amount_) (src/TemplateV1WithCre
ationFee.sol#216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

▼解析内容詳細

1.TemplateV1.withdrawRaisedETH()関数は、任意のユーザーに Ether を送信する可能性があります。これは、攻撃者がコントラクトを悪用して Ether を不正に引き出す可能性を秘めています。この関数の実装を検討し、Ether の送信先を厳密に制御する必要があります。

2. StdCheats.vm では、StdCheatsSafe.vm を上書きしています。これは変数の名前が同じであり、影響範囲が広がることによる予期しないバグを引き起こす可能性があります。

変数名を一意にすることで、このような混乱を避けることが重要です。

3. 下記の関数は、返り値を無視しています。これは、トランザクションの結果が正しく処理されない可能性があります。必要に応じて、返り値を適切に処理するように修正することが望まれます。

・Distributor.claim

・FeePool.withdrawToken

・TemplateV1.claim

・TemplateV1.withdrawERC20Onsale

・TemplateV1.initializeTransfer

・TemplateV1WithCreationFee.claim

・TemplateV1WithCreationFee.withdrawERC20Onsale

・TemplateV1WithCreationFee.initializeTransfer

▼MEDIUM リスク検出一覧(24 件)

```
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-131) per
forms a multiplication on the result of a division:
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#98)
        - inverse = (3 * denominator) ^ 2 (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#113)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-131) per
forms a multiplication on the result of a division:
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#98)
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#1
17)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-131) per
forms a multiplication on the result of a division:
        - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#98)
        - inverse *= 2 - denominator * inverse (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#1
18)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-131) per
forms a multiplication on the result of a division:
20)
        - current_rate = (current_rate * RATE_REDUCTION_COEFFICIENT) / RATE_DENOMINATOR (src/YMWK.sol#171-
173)
YMWK.mintable_in_timeframe(uint256,uint256) (src/YMWK.sol#131-182) performs a multiplication on the result
 of a division:
        - to_mint += current_rate * (current_end - current_start) (src/YMWK.sol#164)
        - current_rate = (current_rate * RATE_REDUCTION_COEFFICIENT) / RATE_DENOMINATOR (src/YMWK.sol#171-
173)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

StdCheatsSafe.rawToConvertedEIPTx1559s(StdCheatsSafe.RawTx1559[]).i (lib/forge-std/src/StdCheats.sol#245)
is a local variable never initialized
StdCheatsSafe.readEIP1559ScriptArtifact(string).artifact (lib/forge-std/src/StdCheats.sol#232) is a local
variable never initialized
StdCheatsSafe.rawToConvertedReceipts(StdCheatsSafe.RawReceipt[]).i (lib/forge-std/src/StdCheats.sol#313) i
s a local variable never initialized
StdCheatsSafe.rawToConvertedEIPTx1559(StdCheatsSafe.RawTx1559).transaction (lib/forge-std/src/StdCheats.so
l#252) is a local variable never initialized
StdCheatsSafe.rawToConvertedEIP1559Detail(StdCheatsSafe.RawTx1559Detail).txDetail (lib/forge-std/src/StdCh
eats.sol#268) is a local variable never initialized
YMWK.mintable_in_timeframe(uint256,uint256).i (src/YMWK.sol#151) is a local variable never initialized
StdCheatsSafe.rawToConvertedReceipt(StdCheatsSafe.RawReceipt).receipt (lib/forge-std/src/StdCheats.sol#320
) is a local variable never initialized
StdCheatsSafe.rawToConvertedReceiptLogs(StdCheatsSafe.RawReceiptLog[]).i (lib/forge-std/src/StdCheats.sol#
344) is a local variable never initialized
```

```
FeePool.withdrawToken(address,address[]).i (src/FeePool.sol#30) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

StdChains.getChainWithUpdatedRpcUrl(string,StdChains.Chain) (lib/forge-std/src/StdChains.sol#153-172) igno
res return value by vm.rpcUrl(chainAlias) (lib/forge-std/src/StdChains.sol#155-169)
StdCheatsSafe.isFork() (lib/forge-std/src/StdCheats.sol#428-432) ignores return value by vm.activeFork() (
lib/forge-std/src/StdCheats.sol#429-431)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (lib/openzeppelin-contracts/contracts/token/E
RC721/ERC721.sol#399-421) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,t
okenId,data) (lib/openzeppelin-contracts/contracts/token/ERC721/ERC721.sol#406-417)
FactoryTest.testAddScore(uint256) (test/Factory.t.sol#33-50) ignores return value by factory.auctions(depl
oyAuctionAddr) (test/Factory.t.sol#43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

▼解析内容詳細

1. Math.mulDiv(uint256,uint256,uint256)関数は、除算の結果に対して乗算を実行しています。この操作は、不正確な計算結果を引き起こす可能性があります。

2.TemplateV1._calculateAllocation(uint256,uint256,uint256),TemplateV1WithCreationFee._calculateAllocation(uint256,uint256,uint256)関数も、除算の結果に対して乗算を実行しています。こちらも計算結果が不正確になる可能性があります。

3. YMWK.mintable_in_timeframe(uint256,uint256)関数も、除算の結果に対して乗算を実行しています。同様に、計算結果が不正確になる可能性があります。

4. 未初期化のローカル変数がいくつか検出されております。これらの変数は使用されていない為、コードの可読性やメンテナンス性に影響を与える可能性があります。これらの変数を適切に初期化するか、不要な場合は削除することを検討してください。

■分析結果概要

静的解析の結果を分析した所、いくつかの脆弱性が検出されましたが、現実点でサービスの提供や可用性に影響を及ぼす懸念のある脆弱性は見受けられませんでした。
その他結果に関してはライブラリに対する内容が多く、大きな脆弱性はないと判断しております。
検出された HIGH リスク、MEDIUM リスクの脆弱性(LOW 以下は割愛させて頂いております)については上記に掲載しておりますので、ご確認の上必要に応じて更なるテストや対策を講じることを推奨致します。

5.免責事項

このスマートコントラクト監査は弊社の監査メソッドに基づき脆弱性を検出する物であって、このスマートコントラクトに関連する全てのセキュリティ問題を検出することを保証するものでは御座いません。

さらなるセキュリティ保証をするためには複数の監査会社、及びバグ報奨金プログラム等を実施することを推奨致します。