Approved

SMART CONTRACT SECURITY AUDIT

Gamifly

Scan and check this report was posted at Approved Github

November, 2023

Website: https://approved.ltd

Table of Contents

Disclaimer

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws of the project's smart contract.

Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it.

Before making any judgments, you have to conduct your own independent research.

We will discuss this in more depth in the following disclaimer - please read it fully.

DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. Scan and verify report's presence in the GitHub repository by a qr-code at the title page. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Approved and its affiliates shall not be held responsible to you or anyone else, nor shall Approved provide any guarantee or representation to any person with regard to the accuracy or integrity of the report.

Without any terms, warranties or other conditions other than as set forth in that exclusion and Approved excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills).

The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Approved disclaims all responsibility and responsibilities and no claim against Approved is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent).

Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Our analysis contains following steps:

1. Project Analysis;

2. Manual analysis of smart contracts:
   - Deploying smart contracts on any of the network(Ropsten/Rinkeby) using Remix IDE
   - Hashes of all transaction will be recorded
   - Behaviour of functions and gas consumption is noted, as well.

3. Unit Testing:
   - Smart contract functions will be unit tested on multiple parameters and under multiple conditions to ensure that all paths of functions are functioning as intended.
   - In this phase intended behaviour of smart contract is verified.
   - In this phase, we would also ensure that smart contract functions are not consuming unnecessary gas.
   - Gas limits of functions will be verified in this stage.

4. Automated Testing:
   - Mythril
   - Oyente
   - Manticore
   - Solgraph

Terminology

We categorize the finding into 4 categories based on their vulnerability:

- Low-severity issue — less important, must be analyzed
- Medium-severity issue — important, needs to be analyzed and fixed
- High-severity issue —important, might cause vulnerabilities, must be analyzed and fixed
- Critical-severity issue —serious bug causes, must be analyzed and fixed.

Limitations

The security audit of Smart Contract cannot cover all vulnerabilities. Even if no vulnerabilities are detected in the audit, there is no guarantee that future smart contracts are safe. Smart contracts are in most cases safeguarded against specific sorts of attacks. In order to find as many flaws as possible, we carried out a comprehensive smart contract audit. Audit is a document that is not legally binding and guarantees nothing.

Basic Security Recommendation

Unlike hardware and paper wallets, hot wallets are connected to the internet and store private keys online, which exposes them to greater risk. If a company or an individual holds significant amounts of cryptocurrency in a hot wallet, they should consider using MultiSig addresses. Wallet security is enhanced when private keys are stored in different locations and are not controlled by a single entity.

Token Contract Details for 07.11.2023

Contract Name: **GameTournamentTrophy**

Deployed address: **0xf5edA6C581F4373B07CE111BAF8d1C4Fc21cbAa1**

Token Tracker: **GTT**

Audit Details

Project Name: **Gamifly**

Language: **Solidity**

Compiler Version: **v0.8.18**

Blockchain: **Etherscan**

Social Profiles

Project Website: **https://gamifly.co/**

Project Twitter: **https://twitter.com/Gamiflyco**

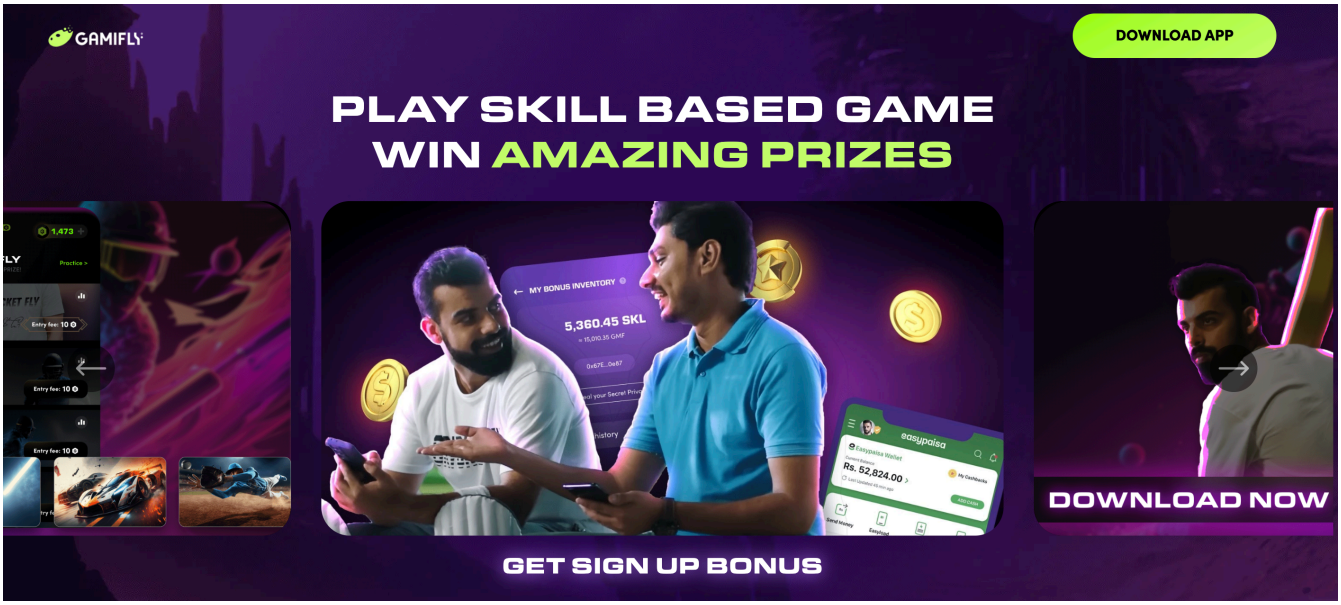
Project Telegram: **https://t.me/gamifly**

Project Medium: **https://gamifly.medium.com/**

Project Instagram: **https://www.instagram.com/gamifly/**

Project Youtube: **https://www.youtube.com/@gamifly**

Project Website Overview



✓JavaScript errors hasn't been found.
✓Malware pop-up windows hasn't been detected.
✓No issues with loading elements, code, or stylesheets.

Vulnerabilities checking

Issue Description	Checking Status
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Timestamp Dependence	Completed
Shadowing State Variables	Completed
DoS with Failed Call	Completed
DoS with Block Gas Limit	Completed
Outdated Complier Version	Completed
Assert Violation	Completed
Use of Deprecated Solidity Functions	Completed
Integer Overflow and Underflow	Completed
Function Default Visibility	Completed
Malicious Event Log	Completed
Math Accuracy	Completed
Design Logic	Completed
Fallback Function Security	Completed
Cross-function Race Conditions	Completed
Safe Zeppelin Module	Completed

Security Issues

1) Assembly Usage:

Configuration

Check: assembly

Severity: Informational

Confidence: **High**

```
function values(AddressSet storage set) internal view returns (address[] memory) {    undefined gas
    bytes32[] memory store = _values(set._inner);
    address[] memory result;

    assembly {
        result := store
    }

    return result;
}
```

```
function values(UintSet storage set) internal view returns (uint256[] memory) {
    bytes32[] memory store = _values(set._inner);
    uint256[] memory result;

    assembly {
        result := store
    }

    return result;
}
```

```
assembly {
    result := store
}
```

L274-L283, L347-356, L351-353

Description:

The use of assembly is error-prone and should be avoided.

Recommendation:

It is recommended that do not use EVM assembly.

2) Different pragma directives are used:

Configuration

Check: Pragma

Severity: Informational

Confidence: **High**

```
pragma solidity ^0.8.0;
```

L04

Description:

Detect whether different solidity versions are used.

Recommendation:

It is recommended to use one solidity version.

3) Dead-Code:

Configuration

Check: dead code

Severity: Informational

Confidence: **Medium**

```
function _setRoleAdmin(bytes32 role, bytes32 adminRole) internal virtual {
    bytes32 previousAdminRole = getRoleAdmin(role);
    _roles[role].adminRole = adminRole;
    emit RoleAdminChanged(role, previousAdminRole, adminRole);
}
```

```
function _setupRole(bytes32 role, address account) internal virtual {
    _grantRole(role, account);
}
```

```
function _msgData() internal view virtual returns (bytes calldata) {
    return msg.data;
}
```

```
function _values(Set storage set) private view returns (bytes32[] memory) {
    return set._values;
}
```

```
function at(Bytes32Set storage set, uint256 index) internal view returns (bytes32) {
    return _at(set._inner, index);
}
```

```
function at(UintSet storage set, uint256 index) internal view returns (uint256) {
    return uint256(_at(set._inner, index));
}
```

L194-L198, L185-L187, L21-L23, L142-L144, L196-L198, L335-L337

Description:

A function that is not sued.

Recommendation:

It is recommended to remove unused functions.

Conclusion for project owner

Medium and Informational severity issues exist within smart contracts.

NOTE: Please check the disclaimer above and note, that audit makes no statements or warranties on business model, investment attractiveness or code sustainability. Contract security report for community

Approved Contact Info

Website: https://approved.ltd

Telegram: @team_approved

GitHub: https://github.com/Approved-Audits/smart_contracts