BLOCKSEC

Security Audit
Report for Pause Guardian Contracts

Date: July 24, 2024 **Version:** 1.0
Contact: contact@blocksec.com

Contents

Report Manifest

Item	Description
Client	BlockSec
Target	Pause Guardian Contracts

Version History

Version	Date	Description
1.0	July 24, 2024	First release

Signature

About BlockSec BlockSec focuses on the security of the blockchain ecosystem and collaborates with leading DeFi projects to secure their products. BlockSec is founded by top-notch security researchers and experienced experts from both academia and industry. They have published multiple blockchain security papers in prestigious conferences, reported several zero-day attacks of DeFi applications, and successfully protected digital assets that are worth more than 14 million dollars by blocking multiple attacks. They can be reached at Email, Twitter and Medium.

Chapter 1   Introduction

1.1 About Target Contracts

Information	Description
Type	Smart Contract
Language	Solidity
Approach	Semi-automatic and manual verification

The target of this audit is the code repository of Pause Guardian Contracts[1] by BlockSec. It is a Safe Wallet module developed by BlockSec, allowing project owners (i.e., the Safe Wallet owners) to authorize specific accounts to activate pausing-related functions of designated contracts during emergencies to prevent further losses to the project.

The auditing process is iterative. Specifically, we would audit the commits that fix the discovered issues. If there are new issues, we will continue this process. The commit SHA values during the audit are shown in the following table. Our audit report is responsible for the code in the initial version (`Version 1`), as well as new code (in the following versions) to fix issues in the audit report.

Project	Version	Commit Hash
Pause Guardian Contracts	Version 1	31598b6205322be707547057e41f0a807e1791b7

Pause Guardian Contracts have been deployed and verified on several networks, including Ethereum, BSC, Arbitrum, Manta and Mantle. The deployed addresses are listed in the following table.

Contract Name	Deployed Address
PauseModuleFactory	0x49a7540C1F46Ca660Df527801a61448Cf7254421
PauseModuleHelper	0xde2fBAec7F747C769B02cA3ba5d399D426F1F284
PauseModule	0x82D804148Fad218f2B3a3CF0cb35BdDE7c6fCb84

1.2 Disclaimer

This audit report does not constitute investment advice or a personal recommendation. It does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Any entity should not rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset.

This audit report is not an endorsement of any particular project or team, and the report does not guarantee the security of any particular project. This audit does not give any warranties on discovering all security issues of the smart contracts, i.e., the evaluation result does

---

[1] https://github.com/blocksecteam/pause_guardian

not guarantee the nonexistence of any further findings of security issues. As one audit cannot be considered comprehensive, we always recommend proceeding with independent audits and a public bug bounty program to ensure the security of smart contracts.

The scope of this audit is limited to the code mentioned in Section 1.1. Unless explicitly specified, the security of the language itself (e.g., the solidity language), the underlying compiling toolchain and the computing infrastructure are out of the scope.

1.3 Procedure of Auditing

We perform the audit according to the following procedure.

- **Vulnerability Detection**   We first scan smart contracts with automatic code analyzers, and then manually verify (reject or confirm) the issues reported by them.
- **Semantic Analysis**   We study the business logic of smart contracts and conduct further investigation on the possible vulnerabilities using an automatic fuzzing tool (developed by our research team). We also manually analyze possible attack scenarios with independent auditors to cross-check the result.
- **Recommendation**   We provide some useful advice to developers from the perspective of good programming practice, including gas optimization, code style, and etc.

We show the main concrete checkpoints in the following.

1.3.1 Software Security

* Reentrancy
* DoS
* Access control
* Data handling and data flow
* Exception handling
* Untrusted external call and control flow
* Initialization consistency
* Events operation
* Error-prone randomness
* Improper use of the proxy system

1.3.2 DeFi Security

* Semantic consistency
* Functionality consistency
* Permission management
* Business logic
* Token operation
* Emergency mechanism
* Oracle security
* Whitelist and blacklist
* Economic impact

    ∗ Batch transfer

1.3.3 NFT Security

    ∗ Duplicated item
    ∗ Verification of the token receiver
    ∗ Off‑chain metadata security

1.3.4 Additional Recommendation

    ∗ Gas optimization
    ∗ Code quality and style

Note *The previous checkpoints are the main ones. We may use more checkpoints during the auditing process according to the functionality of the project.*

1.4 Security Model

To evaluate the risk, we follow the standards or suggestions that are widely adopted by both industry and academy, including OWASP Risk Rating Methodology [2] and Common Weakness Enumeration [3]. The overall *severity* of the risk is determined by *likelihood* and *impact*. Specifically, likelihood is used to estimate how likely a particular vulnerability can be uncovered and exploited by an attacker, while impact is used to measure the consequences of a successful exploit.

In this report, both likelihood and impact are categorized into two ratings, i.e., *high* and *low* respectively, and their combinations are shown in Table 1.1.
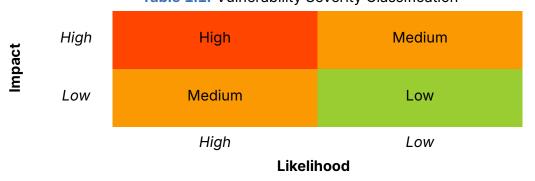
Table 1.1: Vulnerability Severity Classification

Impact			
High	High	Medium	
Low	Medium	Low	
	High	*Low*	
	Likelihood		

Accordingly, the severity measured in this report are classified into three categories: **High**, **Medium**, **Low**. For the sake of completeness, **Undetermined** is also used to cover circumstances when the risk cannot be well determined.

Furthermore, the status of a discovered item will fall into one of the following four categories:

---

[2] https://owasp.org/www‑community/OWASP_Risk_Rating_Methodology

[3] https://cwe.mitre.org/

- **Undetermined** No response yet.
- **Acknowledged** The item has been received by the client, but not confirmed yet.
- **Confirmed** The item has been recognized by the client, but not fixed yet.
- **Fixed** The item has been confirmed and fixed by the client.

Chapter 2   Findings

In total, we did not find potential security issues. Besides, we have **one** recommendation.
- Recommendation: 1

ID	Severity	Description	Category	Status
1	-	Unify the interface for executing transaction	Recommendation	Acknowledged

The details are provided in the following sections.

2.1 Additional Recommendation

2.1.1 Unify the interface for executing transaction

Status   Acknowledged

Introduced by   `Version 1`

Description   In the `PauseModule` contract, the interface for executing a single transaction (i.e., the `execTransaction`) spreads parameters in the signature, while the interface for multiple transaction execution uses the `TransactionData` structure to arrage the parameters. It is recommended to unify the interface for transaction execution for consistent code style.

```
303    function execTransaction(
304        address to,
305        uint256 value,
306        bytes calldata data,
307        uint256 flag
308    ) public onlyDelegate returns (TransactionResult memory result) {
309        result = _execTransaction(to, value, data, flag);
310    }
311
312    function execTransactions(TransactionData[] calldata txDataList) external onlyDelegate returns
             (TransactionResult[] memory resultList) {
```

Listing 2.1: PauseModule.sol

Impact   The code style and interfaces are not consistent.

Suggestion   Unify the interface for transaction execution for consistent code style.

BOOST WEB3 THROUGH NEXT-GENERATION SECURITY & USABILITY INNOVATIONS