



Smart Contract Security Audit Report



Table Of Contents

1 Executive Summary	_____
2 Audit Methodology	_____
3 Project Overview	_____
3.1 Project Introduction	_____
3.2 Vulnerability Information	_____
4 Code Overview	_____
4.1 Contracts Description	_____
4.2 Visibility Description	_____
4.3 Vulnerability Summary	_____
5 Audit Result	_____
6 Statement	_____

1 Executive Summary

On 2026.09.23, the SlowMist security team received the TokenPocket team's security audit application for TokenPocket Inheritance Delegator, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Level	Description
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit
		Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

TokenPocket Asset Inheritance Protection is an on-chain estate planning module for self-custody wallets: the wallet owner may designate an heir in advance during their lifetime and set an inactivity threshold for claim eligibility. Once the wallet remains inactive beyond this specified duration, the heir can submit a claim application. After a subsequent waiting period (during which the owner may cancel the claim at any time), the heir gains operational authority over the wallet's assets. This ensures the assets can be taken over when the private key becomes permanently unavailable due to loss or the owner's passing.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Missing account activity check in completeInheritance function	Design Logic Audit	Low	Acknowledged
N2	delegationManager May Invoke InheritanceManager and Interfere with Inheritance Flow	Design Logic Audit	Medium	Acknowledged
N3	ERC-7710 Calls Do Not Update Account Activity Records	Design Logic Audit	Suggestion	Acknowledged
N4	executeFromExecutor Lacks Validation for the mode Parameter	Others	Suggestion	Acknowledged
N5	cancelInheritance Fails to Clear Residual Token Approvals	Others	Information	Acknowledged
N6	executeBatchAndSkipF ailures Rolls Back Entire Batch on Gas	Gas Optimization Audit	Suggestion	Acknowledged

NO	Title	Category	Level	Status
	Exhaustion Instead of Skipping			

4 Code Overview

4.1 Contracts Description

Audit Version:

<https://github.com/TP-Lab/tokenpocket-inheritance-contracts>

commit: 4c763d1febc96795d7f556e47e53b3f58c683cea

Audit Scope:

```
./src
├── 7710
│   ├── ExecutionHelper.sol
│   ├── ExecutionLib.sol
│   ├── ModeLib.sol
│   └── Simple7710Delegation.sol
├── TokenPocketInheritanceDelegator.sol
└── inheritance
    ├── IInheritanceDelegator.sol
    ├── IInheritanceManager.sol
    ├── InheritanceManager.sol
    └── SimpleInheritanceDelegation.sol
```

The main network address of the contract is as follows:

InheritanceManager:

<https://etherscan.io/address/0xf362080ff1c17bc9f7a00acc3c091806e0cb98e1>

<https://basescan.org/address/0xf362080FF1C17bc9f7A00ACc3c091806E0cB98e1>

<https://bscscan.com/address/0xf362080ff1c17bc9f7a00acc3c091806e0cb98e1>

<https://polygonscan.com/address/0xf362080ff1c17bc9f7a00acc3c091806e0cb98e1>

TokenPocketInheritanceDelegator:

<https://etherscan.io/address/0x8da00456ffc3a0ba1a69d359ad1447b88a4285b0>

<https://basescan.org/address/0x8da00456ffc3a0ba1a69d359ad1447b88a4285b0>

<https://bscscan.com/address/0x8da00456ffc3a0ba1a69d359ad1447b88a4285b0>

<https://polygonscan.com/address/0x8da00456ffc3a0ba1a69d359ad1447b88a4285b0>

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

TokenPocketInheritanceDelegator			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Simple7710Delegation SimpleInheritanceDelegation
execute	External	Can Modify State	-
executeBatch	External	Can Modify State	-
executeBatchAndSkipFailures	External	Can Modify State	-

ExecutionHelper			
Function Name	Visibility	Mutability	Modifiers
_execute	Internal	Can Modify State	-
_tryExecute	Internal	Can Modify State	-
_execute	Internal	Can Modify State	-

ExecutionHelper			
_tryExecute	Internal	Can Modify State	-
_executeDelegatecall	Internal	Can Modify State	-
_tryExecuteDelegatecall	Internal	Can Modify State	-

Simple7710Delegation			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
executeFromExecutor	External	Payable	onlyDelegationManager
supportsExecutionMode	External	-	-

InheritanceManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	Ownable
getState	External	-	-
configureInheritance	External	Can Modify State	whenNotPaused
touch	External	Can Modify State	-
requestInheritance	External	Can Modify State	whenNotPaused
completeInheritance	External	Can Modify State	whenNotPaused
cancelInheritanceRequest	External	Can Modify State	-
cancelInheritance	External	Can Modify State	-
executeAsHeir	External	Can Modify State	whenNotPaused

InheritanceManager			
executeBatchAsHeir	External	Can Modify State	whenNotPaused
isHeirAuthorized	External	-	-
paused	Public	-	-
pause	External	Can Modify State	onlyPauseAuthority
unpause	External	Can Modify State	onlyPauseAuthority
transferPauseAuthority	External	Can Modify State	onlyPauseAuthority
acceptPauseAuthority	External	Can Modify State	-
renouncePauseAuthority	External	Can Modify State	onlyPauseAuthority
_requireValidInheritanceAccount	Internal	-	-
_isValidInheritanceAccount	Internal	-	-
_isEip7702Delegated	Internal	-	-
_supportsInheritanceDelegation	Internal	-	-
_requireConfigured	Internal	-	-
_requireEnabledHeir	Internal	-	-
_isInactive	Internal	-	-
_now	Internal	-	-
rescueERC20	Public	Can Modify State	onlyOwner

SimpleInheritanceDelegation			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

SimpleInheritanceDelegation			
supportsInheritanceDelegation	External	-	-
executeFromHeir	External	Can Modify State	onlyInheritanceManager
executeBatchFromHeir	External	Can Modify State	onlyInheritanceManager
_executeSimple	Internal	Can Modify State	-
_executeBatch	Internal	Can Modify State	-
_touchInheritance	Internal	Can Modify State	-

4.3 Vulnerability Summary

[N1] [Low] Missing account activity check in completeInheritance function

Category: Design Logic Audit

Content

In the InheritanceManager contract, only the `requestInheritance` function calls `_isInactive()` to verify whether an account is still active. The `completeInheritance` function, which actually determines asset ownership, only checks whether an inheritance application exists and whether the waiting period has elapsed. It does not verify if the account owner remains active at all.

Once the heir submits an application after the owner's last activity exceeds `inactivityDuration`, no matter how many genuine transactions the owner subsequently performs (`execute` refreshes `lastActiveTime`), as long as the application is not cancelled, the inheritance will finalize once the waiting period expires, and the account assets can be transferred out.

Code Location:

src/inheritance/InheritanceManager.sol:102-118

```
function completeInheritance(address account) external override whenNotPaused {
    _requireValidInheritanceAccount(account);
    InheritanceState storage state = states[account];
    _requireConfigured(state);
    if (msg.sender != state.heir) revert NotHeir();
    if (state.inheritRequestedAt == 0) revert NoPendingInheritance();
    if (block.timestamp < uint256(state.inheritRequestedAt) +
uint256(state.inheritTimelock)) {
        revert TimelockNotExpired();
    }

    ...
}
```

Solution

It is recommended to add the `_isInactive` check in the `completeInheritance` function, and modify the `touch` function to invalidate pending inheritance applications (`state.inheritRequestedAt = 0`).

Status

Acknowledged; The project team responded: It meets expectations; no changes are required.

[N2] [Medium] delegationManager May Invoke InheritanceManager and Interfere with Inheritance

Flow

Category: Design Logic Audit

Content

Within the Simple7710Delegation contract, `executeFromExecutor()` decodes the `Execution` payload supplied by the executor directly as actions of the account itself without restricting targets. This allows the executor to: pass InheritanceManager as the target parameter to invoke the `configureInheritance` function and set themselves as the heir, then take over the account through the normal workflow; or invoke `cancelInheritanceRequest` / `cancelInheritance` functions to disrupt the regular inheritance process.

Code Location:

src/7710/Simple7710Delegation.sol:53-89

```
function executeFromExecutor(
    ModeCode _mode,
    bytes calldata _executionCalldata
)
external
payable
virtual
onlyDelegationManager
returns (bytes[] memory returnData_) {
    ...
}
```

Solution

It is recommended that `executeFromExecutor()` reject sensitive targets supplied in external calls, such as the `InheritanceManager` contract. Alternatively, validate function selectors for external calls.

Status

Acknowledged; The project team responded: It meets expectations; no changes are required.

[N3] [Suggestion] ERC-7710 Calls Do Not Update Account Activity Records

Category: Design Logic Audit

Content

In the `TokenPocketInheritanceDelegator` contract, `_touchInheritance()` is only invoked at the end of three entry points: `execute`, `executeBatch`, and `executeBatchAndSkipFailures`. When `delegationManager` operates the account via the ERC-7710 path (`executeFromExecutor`), `lastActiveTime` is not refreshed. On-chain, the account appears fully inactive; it may be prematurely marked as inactive, triggering the inheritance workflow.

Code Location:

src/7710/Simple7710Delegation.sol:53-89

```
function executeFromExecutor(
    ModeCode _mode,
    bytes calldata _executionCalldata
)
```

```
external
payable
virtual
onlyDelegationManager
returns (bytes[] memory returnData_) {
    ...
}
```

Solution

Call the `_touchInheritance` function after successful execution inside `executeFromExecutor()` to update the account's last active timestamp.

Status

Acknowledged; The project team responded: It meets expectations; no changes are required.

[N4] [Suggestion] executeFromExecutor Lacks Validation for the mode Parameter

Category: Others

Content

In the Simple7710Delegation contract, the account externally declares support only for MODE_DEFAULT (selector=0, payload=0). However, the execution path within `executeFromExecutor` only decodes callType/execType while ignoring and accepting arbitrary selector/payload values. This creates an inconsistency between the advertised and actual supported modes, violating the ERC-7579 principle that unsupported modes must revert.

Code Location:

src/7710/Simple7710Delegation.sol:53-89

```
function executeFromExecutor(
    ModeCode _mode,
    bytes calldata _executionCalldata
)
external
payable
virtual
onlyDelegationManager
returns (bytes[] memory returnData_) {
```

```
(CallType callType_, ExecType execType_,) = _mode.decode();  
  
...  
}
```

Solution

It is recommended to invoke the `supportsExecutionMode` function within `executeFromExecutor` to perform the check.

Status

Acknowledged; The project team responded: It meets expectations; no changes are required.

[N5] [Information] `cancelInheritance` Fails to Clear Residual Token Approvals

Category: Others

Content

An heir may call `approve(heir, max)` on any ERC-20 token via the `executeAsHeir` function. This creates persistent approvals within the window period that do **not** get revoked when the inheritance right is cancelled. The reason is that `cancelInheritance()` only deletes the account's inheritance state but does not clear the allowances. Consequently, even after the account owner takes remedial action and revokes the heir's inheritance privileges, the formerly-disqualified heir may still drain the account assets. This behavior should be documented in the product documentation.

Code Location:

src/inheritance/InheritanceManager.sol:133-140

```
function cancelInheritance() external override {  
    _requireValidInheritanceAccount(msg.sender);  
    delete states[msg.sender];  
    emit InheritanceCancelled(msg.sender);  
}
```

Solution

N/A

Status

Acknowledged; The project team responded: It meets expectations; no changes are required.

[N6] [Suggestion] executeBatchAndSkipFailures Rolls Back Entire Batch on Gas Exhaustion Instead of Skipping**Category: Gas Optimization Audit****Content**

The `executeBatchAndSkipFailures` function is designed to emit `ExecuteFailures(i)` for failed subcalls and continue execution, keeping the overall transaction successful. However, there exists a critical gas edge case: no gas is reserved for the trailing `_touchInheritance()` call. Each subcall receives the full `gasleft()`. A gas-consuming subcall can exhaust all available gas, triggering an OOG error on the subsequent trailing call. This causes the entire batch to revert. Precisely the class of failures that should be skipped cannot be skipped.

Code Location:

src/TokenPocketInheritanceDelegator.sol:30-43

```
function executeBatchAndSkipFailures(Call[] calldata calls) virtual external {
    _requireForExecute();

    uint256 callsLength = calls.length;
    for (uint256 i = 0; i < callsLength; i++) {
        Call calldata call = calls[i];
        bool success = Exec.call(call.target, call.value, call.data, gasleft());
        if (!success) {
            emit ExecuteFailures(i);
        }
    }

    _touchInheritance();
}
```

Solution

Forward a limited amount of gas to each subcall (e.g., `gasleft() * 63 / 64`) to reserve gas for the trailing `_touchInheritance`.

Status

Acknowledged; The project team responded: It meets expectations; no changes are required.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002609240001	SlowMist Security Team	2026.09.23 - 2026.09.24	Passed

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 1 low risk vulnerabilities, 3 suggestions and 1 information. And all findings were acknowledged.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.



Official Website
www.slowmist.com



E-mail
team@slowmist.com



Twitter
[@SlowMist_Team](https://twitter.com/SlowMist_Team)



Github
<https://github.com/slowmist>