Pépouze

SMART CONTRACT SECURITY AUDIT REPORT

Smart Contract Security Audit Report

Prepared for: Compound V2 (sample)

Report ID: PEPOUZE-SAMPLE-COMPOUND-V2

Audit date: May 1, 2026, 12:00 AM UTC

Pépouze Audit Engine v2.0 - Confidential

Table of Contents

01 Contextual Introduction 3

02 Scope 4

03 Smart Contract Architecture Overview 5

04 Methodology 6

05 Executive Summary 7

06 Detailed Findings 8

07 Gas Optimization Findings 11

08 Informational Findings 12

09 Recommendations Summary 13

10 Conclusion 14

11 Appendix 15

Contextual Introduction

P?pouze performed a security audit of the submitted smart contract security audit scope for Compound V2 (sample). The assessment focuses on deployed contract behavior, source-level indicators when available, bytecode-level risk patterns, token-safety intelligence, and operational assumptions visible from the submitted order metadata.

The purpose of this audit is to identify vulnerabilities, misconfigurations, trust assumptions, and code-quality risks before users rely on the contracts with real funds. Smart contract security matters because deployed EVM code is transparent, adversarially accessible, and difficult or impossible to patch without carefully designed upgrade mechanisms. A small authorization, oracle, accounting, or external-call mistake can become a permanent loss event.

P?pouze runs a fully automated pipeline: the Slither static analyzer, EVM bytecode heuristics, and token-safety intelligence - with no manual review step. This report should be treated as a professional remediation roadmap: it highlights issues observed by tooling, explains likely impact, and identifies areas where the client team should perform deeper manual validation of their own.

Scope

Compound V2 - Comptroller & cToken (sample)

Smart Contract - Generated May 1, 2026, 12:00 AM UTC - Score 78/100 - Overall risk: Medium

Explorer: https://etherscan.io/address/0x3d9819210A31b4961b30EF54bE2aeD79B9c9Cd3B

Chain |
Ethereum |

Chain ID |
1 |

Address |
0x3d9819210A31b4961b30EF54bE2aeD79B9c9Cd3B |

Source origin |
sourcify |

Source available |
true |

Contract name |
Comptroller |

Compiler version |
0.8.26 |

Hash algorithm |
SHA-256 source |

Architecture notes: Automated analysis of the Compound V2 Comptroller and cToken scope completed. The overall risk is medium: no unauthenticated fund-loss path was observed, but oracle-dependency, privileged-admin, and external-call ordering items warrant remediation review before relying on the deployment. Key externally relevant paths are inferred from verified source when available, and from bytecode or token-safety indicators when source is unavailable.

In Scope

- Submitted contract addresses, verified source where available, runtime bytecode, token-safety metadata, and public explorer context.

- Common EVM vulnerability classes including access control, reentrancy, oracle handling, unsafe calls, proxy risks, lifecycle controls, gas-driven denial of service, and token-specific hazards.
Out of Scope

- Private keys, off-chain backend systems, hosted frontends, undisclosed deployment scripts, governance processes not visible on-chain, and economic modeling beyond observable code paths.

- Formal verification, exhaustive fuzzing, authenticated infrastructure testing, and legal/compliance review.
Limitations and Caveats

- This is an illustrative sample; findings are representative and not a live engagement.

- Economic modelling and governance-process review are outside automated scope.

Smart Contract Architecture Overview

The audited scope consists of 1 target. The architecture is summarized from verified source metadata where available; when source is unavailable, P?pouze infers risk-relevant behavior from runtime bytecode, token-safety feeds, and detected bytecode patterns.

```
P?pouze Audit Intake
?? Order PEPOUZE-SAMPLE-COMPOUND-V2
?? Smart Contract Security Audit
?? Target 1: 0x3d9819210A31b4961b30EF54bE2aeD79B9c9Cd3B
? ?? Chain: Ethereum
? ?? Source: sourcify
? ?? Risk: Medium / Score 78
?? Outputs: findings, gas notes, recommendations, PDF delivery
```

Compound V2 - Comptroller & cToken (sample)

Smart Contract - Generated May 1, 2026, 12:00 AM UTC - Score 78/100 - Overall risk: Medium

Explorer: https://etherscan.io/address/0x3d9819210A31b4961b30EF54bE2aeD79B9c9Cd3B

Chain |
Ethereum |

Chain ID |
1 |

Address |
0x3d9819210A31b4961b30EF54bE2aeD79B9c9Cd3B |

Source origin |
sourcify |

Source available |
true |

Contract name |
Comptroller |

Compiler version |
0.8.26 |

Hash algorithm |
SHA-256 source |

Architecture notes: Automated analysis of the Compound V2 Comptroller and cToken scope completed. The overall risk is medium: no unauthenticated fund-loss path was observed, but oracle-dependency, privileged-admin, and external-call ordering items warrant remediation review before relying on the deployment. Key externally relevant paths are inferred from verified source when available, and from bytecode or token-safety indicators when source is unavailable.

Key Functions, State, and Data Flow

Key functions are represented in the detailed findings by exact source function/line when source is available or by the bytecode/feed pattern when source is unavailable. State flow review focused on privileged mutations, external calls, token transfer behavior, upgrade pathways, oracle reads, replay domains, and gas-sensitive loops.

External Dependencies and Inheritance

External calls, token feeds, inherited modules, and interfaces are identified from source imports or bytecode heuristics. If the source is not verified, inheritance and interface details remain a manual-review item and should be confirmed by the client team.

Methodology

Static Analysis Tools Used

- Slither static analyzer (100+ detectors) compiled against a matching solc version for verified source

- P?pouze EVM heuristic engine (P?pouze Audit Engine v2.0) with Solidity parser AST checks

- Runtime bytecode heuristic scanner for unverified contracts

- Token-safety enrichment where chain and address support it
Analysis Engines Executed
Every finding in this report is produced by automated tooling. The exact engines, versions, and a hash of the analyzed source are logged below for reproducibility - this is a fully automated audit with no manual review step.

Target | Engine | Version | Status | Findings | Detail |
Compound V2 - Comptroller & cToken (sample) | P?pouze heuristic | pepouze-heuristic-1.0 | ran | 2 | |
Compound V2 - Comptroller & cToken (sample) | Slither | 0.11.5 | ran | 2 | solc 0.8.26, 2 detectors, source hash 4a1c9e0b2f77 |
Compound V2 - Comptroller & cToken (sample) | Mythril | 0.24.8 | ran | 1 | solc 0.8.26, 1 issue, source hash 4a1c9e0b2f77 |
Automated Result Processing

- Correlate and de-duplicate engine output, then rank by severity, exploitability, and confidence.

- Map each finding to business impact and a concrete remediation / code-fix suggestion.

- Flag assumptions that require client-side confirmation, especially privileged roles and upgrade ownership.
Threat Modeling Approach
P?pouze models attackers as unauthenticated public callers, malicious token receivers, MEV searchers, compromised privileged keys, oracle manipulators, and governance/upgrade adversaries. Detector logic targets the invariant failure behind each class and pairs it with a remediation the client can verify with regression tests.

Severity Rating Scale

Critical | Direct or near-direct path to loss of funds, permanent contract takeover, or protocol insolvency. |
High | Material exploit path that can cause significant financial, governance, or availability impact. |
Medium | Security weakness that depends on assumptions, privileged mistakes, market conditions, or chained issues. |
Low | Limited-impact weakness, defense-in-depth issue, or robustness gap. |
Informational | Best-practice, maintainability, documentation, or operational observation. |

Executive Summary

P?pouze reviewed 1 target for order PEPOUZE-SAMPLE-COMPOUND-V2. The overall automated risk rating is Medium, with an average security score of 78/100.

Critical0 | High1 | Medium1 | Low2 | Informational0 |
Findings Summary

ID | Severity | Finding | Target | Code Reference | Confidence |

NL-001 |
High |
External call precedes state update in redeem path
Reentrancy |
Compound V2 - Comptroller & cToken (sample) |
redeemFresh() @ L640 |
medium |

NL-002 |
Medium |
Price dependency may be manipulable within a block
Oracle / Flash Loan |
Compound V2 - Comptroller & cToken (sample) |
getUnderlyingPrice() @ L212 |
medium |

NL-003 |
Low |
Privileged admin can change market parameters
Governance / Centralization |
Compound V2 - Comptroller & cToken (sample) |
_setCollateralFactor() @ L388 |
medium |

NL-004 |
Low |
Return value of token transfer not consistently checked
External Calls |
Compound V2 - Comptroller & cToken (sample) |
doTransferOut() @ L1102 |
medium |

Key Recommendations at a Glance

- Fix NL-001 before deployment: External call precedes state update in redeem path.
- Verify all submitted contract addresses and source code on the relevant block explorer or Sourcify.
- Add regression tests for every remediated finding and re-run P?pouze before production deployment.

Detailed Findings

High NL-001 Status: Open

External call precedes state update in redeem path

Category | Reentrancy |

Target | Compound V2 - Comptroller & cToken (sample) |

Code Reference | redeemFresh() @ L640 |

Confidence | medium |

Description
In the token redemption flow, value is transferred to the caller before internal accounting is finalized. Without a reentrancy guard, this ordering increases classic and cross-function reentrancy exposure.

Evidence

- SWC references: SWC-107

Impact
Successful exploitation may materially affect user funds, privileged operations, market integrity, token transferability, or system availability.

Proof of Concept
A malicious contract calls the vulnerable function, receives control during the external call, and re-enters before balances or accounting state are finalized.

Recommendation / Code Fix
Apply checks-effects-interactions ordering and add a reentrancy guard around balance-mutating functions. Emit accounting events after state is settled.

```
function withdraw(uint256 amount) external nonReentrant {
uint256 balance = balances[msg.sender];
require(balance >= amount, "insufficient balance");
balances[msg.sender] = balance - amount;
(bool ok, ) = msg.sender.call{value: amount}("");
require(ok, "transfer failed");
}
```

Medium NL-002 Status: Open

Price dependency may be manipulable within a block

Category | Oracle / Flash Loan |

Target | Compound V2 - Comptroller & cToken (sample) |

Code Reference | getUnderlyingPrice() @ L212 |

Confidence | medium |

Description
Collateral valuation relies on an external price source. If a spot value can be moved within a single block, lending and liquidation logic built on it can be gamed via flash loans.

Evidence

- SWC references: SWC-120

Impact
The issue can weaken assumptions, create failure modes under adverse conditions, or become exploitable when combined with another weakness.

Proof of Concept
An attacker manipulates the referenced market or stale oracle response during the transaction window and executes a trade or liquidation against incorrect pricing.

Recommendation / Code Fix

Prefer time-weighted or sequenced oracle updates with staleness checks over raw spot values, and bound acceptable deviation per update.

```
function readPrice() internal view returns (uint256 price) {
(uint80 roundId, int256 answer,, uint256 updatedAt, uint80 answeredInRound) = feed.latestRoundData();
require(answer > 0, "invalid oracle answer");
require(answeredInRound >= roundId, "stale round");
require(block.timestamp - updatedAt <= MAX_ORACLE_DELAY, "stale price");
return uint256(answer);
}
```

Low NL-003 Status: Open

Privileged admin can change market parameters

Category | Governance / Centralization |

Target | Compound V2 - Comptroller & cToken (sample) |

Code Reference | _setCollateralFactor() @ L388 |

Confidence | medium |

Description

Administrative functions can adjust collateral factors, add markets, and pause activity. This is expected for the protocol design but concentrates operational trust in the admin role.

Evidence

- SWC references: SWC-105

Impact

The direct impact is limited, but remediation improves reliability, incident response, maintainability, or defense in depth.

Proof of Concept

An unprivileged address calls the exposed administrative entrypoint directly and changes a sensitive parameter or ownership-dependent state.

Recommendation / Code Fix

Route privileged parameter changes through a timelock and multisig, and emit detailed events for every configuration change.

```
function setCriticalParameter(uint256 value) external onlyRole(GOVERNANCE_ROLE) {
require(value <= MAX_ALLOWED_VALUE, "value out of bounds");
criticalParameter = value;
emit CriticalParameterUpdated(value);
}
```

Low NL-004 Status: Open

Return value of token transfer not consistently checked

Category | External Calls |

Target | Compound V2 - Comptroller & cToken (sample) |

Code Reference | doTransferOut() @ L1102 |

Confidence | medium |

Description

Some ERC-20 interactions do not validate the boolean return value. Non-standard tokens that return false instead of reverting could leave state partially applied.

Evidence

- SWC references: SWC-104

Impact

The direct impact is limited, but remediation improves reliability, incident response, maintainability, or defense in depth.

Proof of Concept

Reviewers should construct a targeted unit test around the referenced code path, assert the expected invariant, and then demonstrate that the invariant fails before remediation and passes after the fix.

Recommendation / Code Fix

Use a safe-transfer wrapper that reverts on a false return, and validate low-level call results explicitly.

```
// Apply the project-specific remediation at the referenced code path.
// Add explicit validation, emit an event for sensitive state changes,
// and cover the fixed invariant with a regression test before deployment.
require(validatedCondition, "invalid state");
```

Gas Optimization Findings

The following opportunities should be reviewed after security remediations are complete. Estimated savings vary with compiler version, optimizer settings, calldata shape, and runtime path frequency.

- Cache frequently-read storage variables in memory inside loops - estimated 100-2,100 gas saved per repeated SLOAD.
- Prefer calldata for external read-only array/string parameters - estimated 500-5,000 gas saved depending on payload size.
- Use custom errors for revert-heavy validation paths - estimated deployment bytecode reduction and 50-200 gas saved per revert path.
- Mark constructor-set addresses and limits as immutable where business logic permits - estimated 100-2,100 gas saved per read.
- Use unchecked increments only in provably bounded loops - estimated 30-60 gas saved per iteration after review.

Informational Findings

- NL-003: Privileged admin can change market parameters (Governance / Centralization).
- NL-004: Return value of token transfer not consistently checked (External Calls).
- Compound V2 - Comptroller & cToken (sample): This is an illustrative sample; findings are representative and not a live engagement.
- Compound V2 - Comptroller & cToken (sample): Economic modelling and governance-process review are outside automated scope.
- Document privileged roles, emergency contacts, upgrade runbooks, and monitoring ownership before production launch.

Recommendations Summary

Short-Term: Fix Before Deployment

- Fix NL-001 before deployment: External call precedes state update in redeem path.
- Verify all submitted contract addresses and source code on the relevant block explorer or Sourcify.
- Add regression tests for every remediated finding and re-run P?pouze before production deployment.
Long-Term Improvements
- Maintain a monitored security runbook covering upgrades, signer rotation, pause authority, and incident communication.
- Schedule follow-up review after each material code, role, oracle, bridge, or dependency change.
- Adopt continuous monitoring for privileged transactions, abnormal transfer patterns, and oracle freshness.

Conclusion

P?pouze identified one high-impact and several lower-severity automated findings in the submitted scope. Treat the reentrancy-ordering and oracle-dependency items as priority remediation work, validate the privileged-admin surface against your governance model, and re-run the audit after fixes are complete.

Deployment readiness depends on closing Critical and High items, validating Medium items against business assumptions, and the client completing their own review of owner/admin operations. P?pouze recommends re-running the audit after remediation and before any production launch or material upgrade.

P?pouze sign-off: This report was produced by P?pouze's fully automated security analysis pipeline and is provided for client remediation planning. P?pouze is available for follow-up scans and clarification.

Appendix

Tools and Versions

- Slither static analyzer (per-run version logged in the Methodology section)
- P?pouze Audit Engine v2.0
- Solidity parser checks via @solidity-parser/parser
- Runtime bytecode heuristics and token-safety enrichment
- PDF rendering via pdf-lib

Glossary

- Reentrancy: External control returns to the contract before state is finalized.
- Oracle: A data source used for pricing, valuation, or external facts.
- Proxy: A contract pattern that delegates logic to an implementation address.
- MEV: Miner/validator/searcher extractable value from transaction ordering.
- Bytecode-only: Analysis mode used when verified Solidity source is unavailable.

Disclaimer and Terms Excerpt

This report is produced by an exclusively automated analysis pipeline and is provided "as is", without warranty of any kind, express or implied. It is not a guarantee that the analyzed contracts or systems are free from vulnerabilities, secure, or safe to deploy. Automated analysis cannot detect every vulnerability; a clean or high-scoring result does not imply the absence of risk.

Any fix, patch, code, or architecture recommendation in this report is illustrative guidance only - not deployable production code, and not professional engineering, security, legal, financial, or investment advice. The client is solely responsible for independently reviewing, testing, validating, and integrating any suggestion, and for re-verifying the corrected code before deployment.

To the maximum extent permitted by applicable law, P?pouze assumes no liability for any loss, damage, exploit, or loss of funds arising from the use of this report or from any system it analyzed, and its aggregate liability is limited to the amount paid for this report. The client remains solely responsible for deployment decisions, key management, monitoring, and incident response. This report does not constitute financial, investment, or legal advice. Full terms: https://pepouze.app/terms

Contact

P?pouze - https://pepouze.app - Reply to the delivery email for questions, remediation review, or follow-up scans.

Appendix Continued - Verification Checklist

Confirm every deployed address, proxy implementation, admin, owner, pauser, oracle, bridge endpoint, treasury wallet, and signer set before launch.

For each remediation, require a pull request, unit/regression test, reviewer sign-off, deployment note, and post-deployment monitoring alert.

Compound V2 - Comptroller & cToken (sample): Medium risk, 78/100 score, source origin sourcify.

Appendix Continued - Threat Model Matrix

Public callers: validate authorization, input bounds, replay protection, and payable flows.

Privileged operators: validate multisig ownership, timelocks, emergency pause scope, signer rotation, and admin event coverage.

Economic adversaries: validate oracle freshness, slippage bounds, liquidation assumptions, MEV exposure, and market manipulation resistance.

Composability adversaries: validate token callbacks, ERC-777-style hooks, flash loans, malicious receivers, and non-standard ERC-20 behavior.

Appendix Continued - Remediation Worksheet

Priority 1: remediate Critical and High findings, then rerun the audit and compare the severity distribution.

Priority 2: validate Medium findings against business logic, protocol economics, and deployment configuration.

Priority 3: schedule Low, Informational, and gas items into the next hardening sprint after security fixes are merged.

Definition of done: code fix merged, tests added, deployment configuration reviewed, monitoring updated, and owner sign-off recorded.

Appendix Continued - Operational Sign-off

P?pouze recommends deployment only after the client confirms all privileged roles, source verification, upgrade controls, emergency contacts, and monitoring alerts.

If any target remains bytecode-only, obtain verified source from the development team or reproduce the deployed bytecode from the exact compiler settings before relying on a clean result.

Store this report with deployment artifacts and rerun P?pouze after material code, ownership, oracle, bridge, or dependency changes.

Appendix Continued - Additional Review Notes

This intentionally reserved review page supports client annotations, remediation sign-off, and post-audit deployment notes.

Record owner, reviewer, date, deployed address, commit hash, and monitoring updates before production use.