SAMPLE REPORT

Contract audit | PRO | ANALYSIS REPORT

RESULT: highest finding severity HIGH, to be verified — Unprotected privileged write (anyone can change owner or role) — function DemonstrationOnly.setOwner. HIGH

What to do now: apply the suggested fix (the diff in the key findings section), compile the contract, add a test that an unprivileged account cannot call this function, and run the analysis again before deploying.

Demonstration data: a self-authored test contract, never deployed on a network. This is a short version of an Audit Pro report — the full report also includes execution scope, measurements, data source and the document identifier.

KEY FINDINGS

HIGH / To verify: Unbound-source heuristic candidate: Unprotected privileged write (anyone can change owner or role) — function DemonstrationOnly.setOwner

Summary: Static source signal Unprotected privileged write (anyone can change owner or role) — function DemonstrationOnly.setOwner (UNPROTECTED-PRIVILEGED-WRITE) in DemonstrationOnly.sol, lines 7-7. This is a review lead; it confirms neither a defect nor reachability, supplied-bytecode identity, or an exploit.

Remediation recommendation: restrict the function with access control (for example an onlyOwner modifier from Ownable, two-step ownership transfer with Ownable2Step, or an AccessControl role), reject the zero address (require(next != address(0))) and add a test that an unauthorized account cannot change the owner or role.

Suggested fix — a diff against your file (compile and test before deploying)

```
--- a/DemonstrationOnly.sol
+++ b/DemonstrationOnly.sol
@@ -5,6 +5,7 @@
  address public owner;
  constructor() { owner = msg.sender; }
  function setOwner(address next) external {
+ require(msg.sender == owner);
  owner = next;
  }
  }
```

A caller check was added as the first statement of the function, using the existing state variable "owner" of this contract. Confirm that this is the intended role, compile the code and add a test with an unauthorized account.

SAMPLE REPORT

Contract audit | PRO | ANALYSIS REPORT

WHAT TO CHECK NEXT

To verify: Next step 1 (analyst or contract owner): Verify this candidate against the exact code, state and conditions referenced by its evidence before treating it as a confirmed issue.
To verify: External interactions: Review the recorded scope and evidence for this check. An unresolved status does not establish a problem or guarantee that additional data will resolve it.

LIMITS OF THIS REPORT

1. This is an automated analysis of the supplied code, not a manual audit or a security certificate.
2. We did not compare the code with a version deployed on a network; if the contract is deployed, compare the code before deciding.
3. No findings in an area do not prove the code is safe — the rules detect known error patterns.
4. No attack was run against a deployed contract and no financial impact was computed.
5. The complete list of recorded limitations (26) is in the technical appendix of the TXT file and in the JSON file. Full findings, evidence, measurements and scope rows remain in the private JSON. TXT contains a detailed account.