Smart Contract Audit
BXR Token
1. Executive Summary

2. Assessment and Scope

3. Summary of Findings

4. Detailed Findings
   - Compromise of a single account leads to total takeover

5. Disclaimer
1. Executive Summary

Coinspect performed an assessment of the BXR Token contract deployed at address 0x97A3BD8a445cC187c6A751F392e15C3B2134D695 on the Ethereum mainchain, which was deployed on May 15, 2021.

The following issues were identified during the assessment:

<table>
<thead>
<tr>
<th>Risk Level</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Risk</td>
<td>0</td>
</tr>
<tr>
<td>Medium Risk</td>
<td>1</td>
</tr>
<tr>
<td>Low Risk</td>
<td>0</td>
</tr>
</tbody>
</table>
2. Assessment and Scope

The audit started on May 17 and was conducted on the BXR Token contract deployed at address 0x97A3BD8a445c187c6A751F392e15C3B2134D695 on the Ethereum mainchain with a capped supply of 100,000,000 tokens with 18 decimals.

The SHA256 of the contract source code analyzed is as shown below:

60b31332474874316d257f2b5e8d8fe536f92fcbea1aaab89da7adec6dac985 ./BXRToken.sol

Coinspect verified that most of the BXR Token source code available at Etherscan matches exactly with OpenZeppelin Contracts version 3.4.0, on which BXR Token is based.

Besides the vulnerability described in this document, Coinspect has general recommendations for the smart contract which would enhance its usability and ease of audit in the future:

- Make `pause()`, `unpause()` and `mint()` external, which would make them cheaper to call than the current public modifier.
- Maintain a public repository including the contract code, tests, documentation, and deployment scripts.

Without its dependencies, the source code of the contract analyzed is as follows:

```solidity
contract BXRToken is ERC20Burnable, ERC20Capped, ERC20Pausable, AccessControl {
    bytes32 public constant PAUSER_ROLE = keccak256("PAUSER_ROLE");
    bytes32 public constant MINTER_ROLE = keccak256("MINTER_ROLE");

    constructor() public ERC20Capped(100 * 10**6 * 10**18) ERC20("Blockster", "BXR") {
        _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
        _setupRole(PAUSER_ROLE, msg.sender);
        _setupRole(MINTER_ROLE, msg.sender);
    }

    function pause() public {
        require(hasRole(PAUSER_ROLE, msg.sender));
    }
}
```

© 2021 Coinspect
function unpause() public {
    require(hasRole(PAUSER_ROLE, msg.sender));
    _unpause();
}

function mint(address to, uint256 amount) public {
    require(hasRole(MINTER_ROLE, msg.sender));
    _mint(to, amount);
}

function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual override(ERC20, ERC20Pausable, ERC20Capped) {
    super._beforeTokenTransfer(from, to, amount);
}
### 3. Summary of Findings

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Risk</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>BXR-001</td>
<td>Compromise of a single account leads to total takeover</td>
<td>Medium</td>
<td>✘</td>
</tr>
</tbody>
</table>
4. Detailed Findings

**BXR-001  Compromise of a single account leads to total takeover**

<table>
<thead>
<tr>
<th>Total Risk</th>
<th>Impact</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>High</td>
<td>./BXRToken.sol</td>
</tr>
<tr>
<td>Fixed</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

Compromising the private key of the externally owned account 0x2B9AF0bd212BF9969Ed7308F7144ff281f9b8d42 would grant the adversary control over all aspects of the token, as that account is Admin, Minter and Pauser of the contract.

```solidity
constructor() public ERC20Capped(100 * 10**6 * 10**18) ERC20("Blockster", "BXR") {
    _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
    _setupRole(PAUSER_ROLE, msg.sender);
    _setupRole(MINTER_ROLE, msg.sender);
}
```

Coinspect confirmed the roles have not been segregated into different accounts since deployment.

**Recommendation**

Segregate the roles into three different accounts.

As a further defense mechanism, it is advisable to assign the Admin, Pauser and Minter roles to a multisig contract, so no single set of keys has control over the contract.
5. Disclaimer

The information presented in this document is provided "as is" and without warranty. The present security audit does not cover any off-chain systems and frontends that communicate with the contracts, nor the general operational security of the organization that developed the code.