Secure Code Review of Project Blackbird

Shell International Petroleum Ltd

May 2022

Version 2.1

Presented by:

BTblock LLC

Corporate Headquarters

FYEO Inc.

PO Box 147044
Lakewood CO 80214
United States

Security Level:
Strictly Confidential

Table of Contents

List of Figures

List of Tables

Executive Summary

Overview

Shell International Petroleum Ltd engaged BTblock LLC to perform a Secure Code Review of Project Blackbird.

The assessment was conducted remotely by the BTblock Security Team. Testing took place on March 31 - April 14, 2022, and focused on the following objectives:

- Provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the result of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the BTblock Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following are the issues identified during the testing period. These should be prioritized for remediation to reduce to the risk they pose:

- BT-SHELL-01 – AccessManager.sol - Platform owner can assign more platform owners
- BT-SHELL-02 – AccessManager.sol - Roles can be manipulated from interface functions
- BT-SHELL-03 – AcccessManager.sol - Similar behavior between functions
- BT-SHELL-04 – InventoryManagement.sol - Inaccurate permission checks during indirect transfers
- BT-SHELL-05 – Code Quality
- BT-SHELL-06 – InventoryManagement.sol - Token transfers can be simplified
- BT-SHELL-07 – Missing Testing Instructions

During the test, the following positive observations were noted regarding the scope of the engagement:

- The Shell team was very supportive and open to discuss the design choices made. They were hands on and the collaboration between the Shell development team and BTblock's security team was very productive and efficient. They were a pleasure to work with.

Based on formal verification we conclude that the reviewed code implements the documented functionality.

Scope and Rules of Engagement

BTblock performed a Secure Code Review of Project Blackbird. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a private repository at https://github.com/sede-x/saf-dea-transfer with the commit hash b054d61cb5390a60d2b92da52859cabb7f3d9f63. A re-review of the fixes was performed on April 27, 2022, with the commit hash 16ef2b1024291b2eed52353ac7f0f3755de58bc0.

Files included in the code review

```
saf-dea-transfer/
├── blockchain/
│   ├── contracts/
│   │   ├── test/
│   │   │   ├── AccessManager.sol
│   │   │   ├── CalculatorConstants.sol
│   │   │   └── InventoryManagement.sol
│   │   ├── AccessManager.sol
│   │   ├── CalculatorConstants.sol
│   │   └── InventoryManagement.sol
│   ├── scripts/
│   │   ├── add-customer-script.js
│   │   ├── deploy-script.js
│   │   ├── upgradable-script_access_manager.js
│   │   ├── upgradable-script_calculator.js
│   │   └── upgradable-script_inventory.js
│   ├── test/
│   │   ├── AccessManager_Test.js
│   │   ├── CalculatorConstants_Test.js
│   │   └── InventoryManagement_Test.js
│   ├── typechain-types/
│   │   ├── factories/
│   │   │   ├── AccessManager__factory.ts
│   │   │   ├── CalculatorConstants__factory.ts
│   │   │   └── InventoryManagement__factory.ts
│   │   ├── AccessManager.ts
│   │   ├── CalculatorConstants.ts
│   │   ├── InventoryManagement.ts
│   │   ├── common.ts
│   │   └── index.ts
│   ├── README.md
│   ├── hardhat.config.js
│   ├── package-lock.json
│   └── package.json
├── common/
│   ├── config/
│   │   └── rush/
│   │       ├── artifactory.json
│   │       ├── build-cache.json
│   │       ├── command-line.json
│   │       ├── common-versions.json
│   │       ├── deploy.json
│   │       ├── experiments.json
│   │       ├── pnpm-lock.yaml
│   │       ├── rush-plugins.json
│   │       └── version-policies.json
│   ├── env/
│   │   ├── README.md
│   │   ├── batchfuel.env
│   │   ├── calculator.env
│   │   ├── fetch-secrets.sh
│   │   ├── inventory-management.env
│   │   ├── saf-frontend.env
│   │   ├── update-secrets.sh
│   │   └── user-authentication.env
│   ├── git-hooks/
│   │   └── commit-msg.sample
│   └── scripts/
│       ├── install-run-rush.js
│       ├── install-run-rushx.js
│       └── install-run.js
├── scripts/
│   └── check_files_changed.sh
├── DockerfileBackend
├── README.md
├── rush.json
└── wss-unified-agent.config
```

Table 1: Scope

Technical Analyses and Findings

During the Secure Code Review of Project Blackbird, we discovered:

- 2 findings with MEDIUM severity rating.
- 2 findings with LOW severity rating.
- 3 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

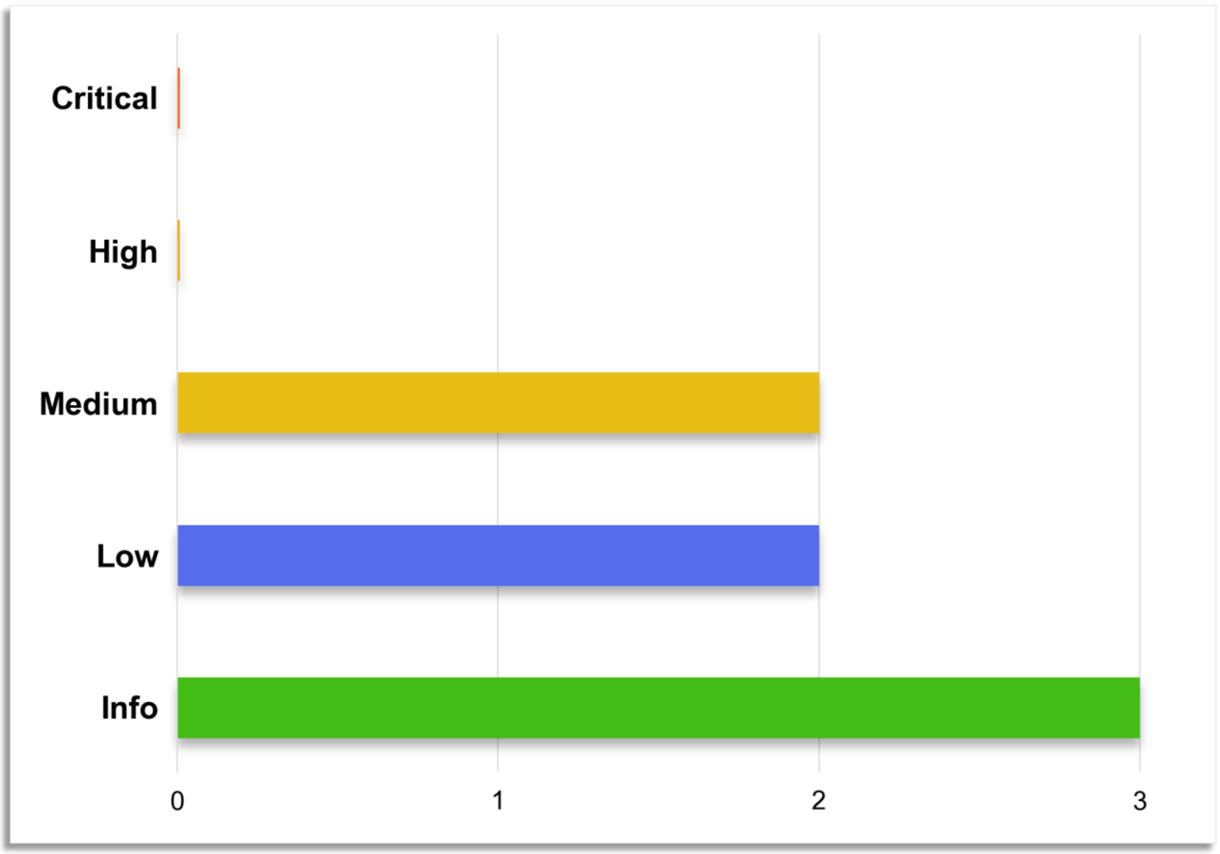

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Status	Description
BT-SHELL-01	Medium	Remediated	AccessManager.sol - Platform owner can assign more platform owners
BT-SHELL-02	Medium	Remediated	AccessManager.sol - Roles can be manipulated from interface functions
BT-SHELL-03	Low	Remediated	AcccessManager.sol - Similar behavior between functions
BT-SHELL-04	Low	Remediated	InventoryManagement.sol - Inaccurate permission checks during indirect transfers
BT-SHELL-05	Informational	Remediated	Code Quality
BT-SHELL-06	Informational	Open	InventoryManagement.sol - Token transfers can be simplified
BT-SHELL-07	Informational	Remediated	Missing Testing Instructions

Table 2: Findings Overview

Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality. Based on formal verification, we conclude that the code implements the documented functionality to the extent of the reviewed code.

Technical Findings

General Observations

Shell's development team was very communicative, quickly providing responses to the auditing team. The code was well written and heavily documented. The usage of existing standards made the project quite concrete, while the extensions implemented cannot be abused by attackers. In regards to the technical findings, they mostly include cases where the intended users might misuse the provided functionality. The team also provided extensive testing for the implemented functionality.

AccessManager.sol - Platform owner can assign more platform owners

Finding ID: BT-SHELL-01
Severity: Medium
Status: Remediated

Description

The Platform Owner is described as a special role in the `AccessManager.sol` contract. It has the privilege to add new roles and assign roles to addresses. The development team let us know that the Platform Owner is intended to be a single entity. This is not guaranteed by the contract as the platform owner is allowed to assign its role to other addresses as well. The same applies for the `DEFAULT_ADMIN_ROLE`.

Proof of Issue

File name: blockchain/contracts/AccessManager.sol
Line number: 102

```
    function addCompanyType(bytes32 _companyType, address _companyAccount)
        public
    {
        grantRole(_companyType, _companyAccount);
        ...
    }

    function addCompanyTypeAccount(
        bytes32 _companyType,
        address _companyAccount
    ) public {
        require(
            getRoleAdmin(_companyType) != DEFAULT_ADMIN_ROLE,
            "AM: Company type doesn't exist"
        );
        grantRole(_companyType, _companyAccount);
        ...
    }
```

The two functions allow the `PLATFORM_OWNER` to assign the same role to an address.

Severity and Impact Summary

The AccessManager contract does not restrict the platform owner to assign more owners.

Recommendation

Require that `companyType` is never `PLATFORM_OWNER` nor `DEFAULT_ADMIN_ROLE`.

AccessManager.sol - Roles can be manipulated from interface functions

Finding ID: BT-SHELL-02
Severity: Medium
Status: Remediated

Description

The `AccessManager` contract extends OpenZeppelin's `AccessControlUpgradeable`. Thus, all functions of the parent contract are also available to use. This includes `grantRole` and `revokeRole`, which can be used by the platform owner to grant roles without updating the internal state or emitting events. This could also lead to the platform owner renouncing its roles.

Severity and Impact Summary

Interface functions can be used to manipulate roles, bypassing checks and without updating internal state.

Recommendation

Implement `grantRole` and `revokeRole` with revert functionality and ensure that their requirements are met when they are used.

AcccessManager.sol - Similar behavior between functions

Finding ID: BT-SHELL-03
Severity: Low
Status: Remediated

Description

The `AccessManager` contract provides functions `addCompanyType` and `addCompanyTypeAccount`. Both subsequently call `grantRole` to give a `_companyType` role to a `_companyAccount` address:

Proof of Issue

File name: blockchain/contracts/AccessManager.sol
Line number: 120

```
    function addCompanyType(bytes32 _companyType, address _companyAccount)
        public
    {
        grantRole(_companyType, _companyAccount);
        emit AccessManager_ActionDetails(
            ActionType.ADD_COMPANY_TYPE,
            _companyType,
            _companyAccount,
            address(0),
            msg.sender,
            block.number
        );
    }

    function addCompanyTypeAccount(
        bytes32 _companyType,
        address _companyAccount
    ) public {
        require(
            getRoleAdmin(_companyType) != DEFAULT_ADMIN_ROLE,
            "AM: Company type doesn't exist"
        );
        grantRole(_companyType, _companyAccount);
        emit AccessManager_ActionDetails(
            ActionType.ADD_COMPANY_TYPE_ACCOUNT,
            _companyType,
            _companyAccount,
            address(0),
            msg.sender,
            block.number
        );
    }
```

For `grantRole` to succeed, the sender needs to be an admin of that role. When `addCompanyTypeAccount` checks that `getRoleAdmin(_companyType)!= DEFAULT_ADMIN_ROLE`, it restricts `_companyType` to be one of the contract's constants (to which an admin is assigned during contract initialization). This restriction is not present in `addCompanyType`, which therefore can be used in all cases, whether the role has an admin or not.

Severity and Impact Summary

Functions having the same functionality could lead to misuse of the contract.

Recommendation

Require `getRoleAdmin(_companyType) == DEFAULT_ADMIN_ROLE` to differentiate the functionality of the two functions

InventoryManagement.sol - Inaccurate permission checks during indirect transfers

Finding ID: BT-SHELL-04
Severity: Low
Status: Remediated

Description

ERC1155 allows for indirect transfers, where a user can approve transfers to be performed on their behalf. This could lead to the sender being different from the token holder. This procedure is implemented in `safeTransferFrom`. During transfers, a check is performed on the access of message's sender and the type of token. Performing the transaction on behalf of somebody else would lead to inaccurate checks.

Proof of Issue

File name: blockchain/contracts/InventoryManagement.sol
Line number: 205

```
    function safeTransferFrom(
        address from,
        address to,
        uint256 id,
        uint256 amount,
        bytes calldata data
    ) public override checkPermissions(SUPPLIER) {
        require(
            ((from == _msgSender() || isApprovedForAll(from, _msgSender())) &&
                amount > 0),
            "IM: caller is not owner nor approved or invalid amount"
        );
        _checkTokenTypeAndSupplierGroup(id, to);
```

Line number: 315

```
function _checkTokenTypeAndSupplierGroup(uint256 _tokenId, address _to)
        internal
        view
    {
        if (
            !((_mapTokenIdToTokenDetail[_tokenId].tokenType ==
                TokenType.Scope1 &&
                AccessManager(accessManagerAddress)
                    .checkSupplierCustomerAndCompanyTypeAccess(
                        AIRLINE,
                        _to,
                        msg.sender
                    ) &&
                _mapTokenIdToTokenDetail[_tokenId].owner == msg.sender) ||
                (_mapTokenIdToTokenDetail[_tokenId].tokenType ==
                    TokenType.Scope3 &&
                    AccessManager(accessManagerAddress)
                        .checkSupplierCustomerAndCompanyTypeAccess(
                            CORPORATE,
                            _to,
                            msg.sender
                        ) &&
                    _mapTokenIdToTokenDetail[_tokenId].owner == msg.sender))
        ) {
            revert("IM: Invalid Owner/Token type/Supplier Group");
        }
    }
```

Severity and Impact Summary

Indirect transfers could check for wrong permissions.

Recommendation

Check the transfer's `from` address for the required permissions instead of `msg.sender`

Code Quality

Finding ID: BT-SHELL-05
Severity: Informational
Status: Remediated

Description

This issue was added after it was brought to BTblock's attention by the development team. Overall, the quality of the code is high with extensive documentation and clear functionality. However the following points can be improved upon:

- Functions can be marked as `external`. This descriptor can be added in functions that are not called within the same contract.
- Extensive and sometimes unnecessary usage of `nonReentrant`. The modifier should be used in cases where there's room for re-entrancy. In some cases, e.g., in the setters of `CalculatorConstants.sol`, this wouldn't be possible. Reentrancy typically needs to be avoided in ETH transfers, or in functions that change the state while interacting with entities outside the contract.

Severity and Impact Summary

Minor improvements can improve functionality and reduce amount of instructions.

Recommendation

Mark `external` functions as such, especially if they are intended to be so, and remove `nonReentrant` to reduce costs.

InventoryManagement.sol - Token transfers can be simplified

Finding ID: BT-SHELL-06
Severity: Informational
Status: Open

Description

After batches are minted, `InventoryManagement` uses the following procedure to transfer tokens: it first transfers the amount required, then burns that amount from the destination while it internally records the state change. This could be simplified by immediately burning the tokens on the sender's side since a custom is emitted.

Proof of Issue

File name: blockchain/contracts/InventoryManagement.sol
Line number: 182

```
            _safeTransferFrom(msg.sender, _to[i], _mapBatchIdToTokenIds[_batchId][i],
_amount, "");
            _nonTransferableClaim(_to[i], _mapBatchIdToTokenIds[_batchId][i],
_amount);
```

Line number: 218

```
        _safeTransferFrom(from, to, id, amount, data);
        _nonTransferableClaim(to, id, amount);
```

Line number: 347

```
    function _nonTransferableClaim(
        address _from,
        uint256 _tokenId,
        uint256 _amount
    ) internal {
        _burn(_from, _tokenId, _amount);
    }
```

Severity and Impact Summary

Token transfers could be cheaper and faster.

Recommendation

Burn the tokens on the sender's side and update the internal state accordingly. A caveat would be that the transfer event will be omitted which, however, is negated since a custom event is emitted nevertheless.

Missing Testing Instructions

Finding ID: BT-SHELL-07
Severity: Informational
Status: Remediated

Description

Testing contract functionality includes upgrading the contracts. This causes issues when trying to run the tests. Although, the team was quick to point out the correct way to run the tests, we suggest adding these instructions in the README file to facilitate future developers and auditors. We should also mention that the tests ran successfully after the correct setup.

Severity and Impact Summary

Missing documentation for testing introduces some lag when familiarizing with the project.

Recommendation

Add testing instructions to the README file in order to smoothly run the test.