Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: Backters
Website: https://backters.com
Platform: Polygon Network
Language: Solidity
Date: May 14th, 2022

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the Backters team to perform the Security audit of the BKD and USDBK smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on May 14th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- The BKD777 is the standard ERC777 token whose mint and burn are controlled by redemptionController.
- USDBK777 Contract is a smart contract, having functions like destroy, send, burn, batchTransfer, globalOperators, authorizeGlobalOperator, etc.
- The USDBK777 contract inherits the IERC20, ERC777, SafeMath standard smart contracts from the OpenZeppelin library.
- These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

Email: audit@EtherAuthority.io

Audit scope

Name	Code Review and Security Analysis Report for Backters Protocol Smart Contracts
Platform	**Polygon / Solidity**
File 1	BKD777Token.sol
File 1 MD5 Hash	998502EC75BC90E5A83C43C47AB3CF0B
Updated File 1 MD5 Hash	3E6AA51B2E25EE4189DD2F38555433EC
File 2	RedemptionController.sol
File 2 MD5 Hash	998502EC75BC90E5A83C43C47AB3CF0B
Updated File 2 MD5 Hash	4F09AE3D3EAD8D82EBEDBB678281D073
File 3	USDBK777Token.sol
File 3 MD5 Hash	C5F6130245CA46E9B24FA14F3A3176D5
Updated File 3 MD5 Hash	6819C3C9652FBDDC47F08A15DED5B138
Audit Date	May 14th, 2022
Revise Audit Date	December 13th, 2022

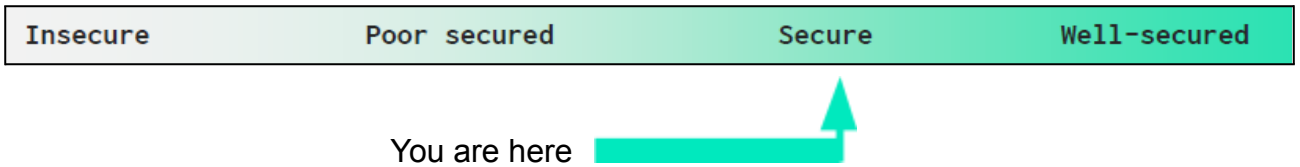
Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 BKD777Token.sol • Authorized operator can mint and burn tokens for wallets. • Authorized operator can destroy the smart contract.	**YES, This is valid.**
File 2 RedemptionController.sol • Manager can add tokens for wallets. • Redeemer can redeem their tokens • Manager can set an interval, period, reward token, reward from account, and redeem token. • Default Admin can destroy the smart contract.	**YES, This is valid.**
File 3 USDBK777Token.sol • Owner can destroy the smart contract. • Owner can burn someone else's tokens. • Open Zeppelin standard code is used.	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. This token contract does contain owner control, which does not make it fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 1 medium and 2 low and some very low level issues.

All the issues have been fixed/acknowledged in revised contract code.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderate
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderate
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 3 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the Backters Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Backters Protocol.

The Backters team has not provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **well** commented on smart contracts.

Documentation

We were given a Backters Protocol smart contract code in the form of a Files. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **well** commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://backters.com which provided rich information about the project architecture.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

BKD777Token.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	decimals	write	Passed	No Issue
5	granularity	read	Passed	No Issue
6	totalSupply	read	Passed	No Issue
7	balanceOf	read	Passed	No Issue
8	send	write	Passed	No Issue
9	transfer	write	Passed	No Issue
10	burn	write	Passed	No Issue
11	isOperatorFor	read	Passed	No Issue
12	authorizeOperator	write	Passed	No Issue
13	revokeOperator	write	Passed	No Issue
14	defaultOperators	read	Passed	No Issue
15	operatorSend	write	Passed	No Issue
16	operatorBurn	write	Passed	No Issue
17	allowance	read	Passed	No Issue
18	approve	write	Passed	No Issue
19	transferFrom	write	Passed	No Issue
20	_mint	internal	Passed	No Issue
21	_mint	internal	Passed	No Issue
22	_send	internal	Passed	No Issue
23	_burn	internal	Passed	No Issue
24	_move	write	Passed	No Issue
25	_approve	internal	Passed	No Issue
26	_callTokensToSend	write	Passed	No Issue
27	_callTokensReceived	write	Passed	No Issue
28	_spendAllowance	internal	Passed	No Issue
29	_beforeTokenTransfer	internal	Passed	No Issue
30	destroy	write	Passed	No Issue
31	getBurnReturnAccount	read	Passed	No Issue
32	getBurnReturnPercentage	read	Passed	No Issue
33	setBurnReturnAccount	write	Passed	No Issue
34	setBurnReturnPercentage	write	Burn Return Percentage limit is not set	Refer Audit Findings
35	getBurnReturnForwardAccount	read	Passed	No Issue
36	getBurnReturnForwardPercentage	read	Passed	No Issue
37	setBurnReturnForwardAccount	write	Passed	No Issue
38	setBurnReturnForwardPercentage	write	Passed	No Issue

39	operatorBurnReturn	write	Function input parameters lack of check	Refer Audit Findings
40	_burnReturn	internal	Passed	No Issue
41	operatorMint	write	Function input parameters lack of check	Refer Audit Findings
42	circulatingSupply	write	Passed	No Issue
43	operatorTransferAnyERC20Token	write	Function input parameters lack of check	Refer Audit Findings
44	batchBalanceOf	write	Passed	No Issue
45	operatorBatchTransfer	write	Infinite loop possibility,Function input parameters lack of check	Refer Audit Findings
46	operatorBatchMint	write	Infinite loop possibility	Refer Audit Findings
47	operatorBatchBurn	write	Infinite loop possibility	Refer Audit Findings

RedemptionController.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	supportsInterface	read	Passed	No Issue
3	getRoleMember	read	Passed	No Issue
4	getRoleMemberCount	read	Passed	No Issue
5	_grantRole	internal	Passed	No Issue
6	_revokeRole	internal	Passed	No Issue
7	destroy	write	Passed	No Issue
8	count	read	Passed	No Issue
9	startsAt	read	Passed	No Issue
10	startsAt	write	Passed	No Issue
11	interval	read	Passed	No Issue
12	periods	read	Passed	No Issue
13	redeemToken	read	Passed	No Issue
14	periods	write	Passed	No Issue
15	redeemToken	write	Passed	No Issue
16	rewardAccount	read	Passed	No Issue
17	rewardAccount	write	Passed	No Issue
18	rewardToken	write	Passed	No Issue
19	rewardToken	read	Passed	No Issue
20	redeemTokenTotalSupply	read	Passed	No Issue
21	rewardTokenBalance	read	Passed	No Issue
22	add	write	Passed	No Issue

23	batchAdd	write	Infinite loop possibility	Refer Audit Findings
24	_addRedemption	internal	Passed	No Issue
25	_add	internal	Passed	No Issue
26	get	read	Passed	No Issue
27	getAll	read	Passed	No Issue
28	operatorMigrateFrom	write	Passed	No Issue
29	getAccountAtOffset	read	Passed	No Issue
30	redeemableAt	read	Passed	No Issue
31	redeemableAt	read	Passed	No Issue
32	redeemable	read	Passed	No Issue
33	redeemable	read	Passed	No Issue
34	_redeemPlansAt	internal	Passed	No Issue
35	_redeemAt	internal	Passed	No Issue
36	_redemptionSchedule	internal	Passed	No Issue
37	redeem	write	Passed	No Issue
38	operatorTransferAnyERC20Token	write	Passed	No Issue
39	supportsInterface	read	Passed	No Issue
40	update	write	Passed	No Issue

USDBK777Token.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	destroy	write	Passed	No Issue
3	authorizeGlobalOperator	write	Passed	No Issue
4	revokeGlobalOperator	write	Passed	No Issue
5	isOperatorFor	read	Passed	No Issue
6	globalOperators	read	Passed	No Issue
7	getTransferEnabled	read	Passed	No Issue
8	setTransferEnabled	write	Passed	No Issue
9	getBurningEnabled	read	Passed	No Issue
10	setBurningEnabled	write	Passed	No Issue
11	send	write	Passed	No Issue
12	transfer	write	Passed	No Issue
13	burn	write	Passed	No Issue
14	transferFrom	write	Passed	No Issue
15	operatorSend	write	Passed	No Issue
16	batchBalanceOf	read	Passed	No Issue
17	batchTransfer	write	Infinite loop possibility	Refer Audit Findings
18	operatorBatchTransfer	write	Infinite loop possibility, Function input parameters lack of check	Refer Audit Findings

19	operatorBatchMint	write	Infinite loop possibility	Refer Audit Findings
20	operatorBatchBurn	write	Infinite loop possibility	Refer Audit Findings
21	operatorMint	write	Function input parameters lack of check	Refer Audit Findings
22	operatorBurn	write	Passed	No Issue
23	operatorTransferAnyERC20Token	write	Function input parameters lack of check	Refer Audit Findings
24	name	read	Passed	No Issue
25	symbol	read	Passed	No Issue
26	decimals	write	Passed	No Issue
27	granularity	read	Passed	No Issue
28	totalSupply	read	Passed	No Issue
29	balanceOf	read	Passed	No Issue
30	send	write	Passed	No Issue
31	transfer	write	Passed	No Issue
32	burn	write	Passed	No Issue
33	isOperatorFor	read	Passed	No Issue
34	authorizeOperator	write	Passed	No Issue
35	revokeOperator	write	Passed	No Issue
36	defaultOperators	read	Passed	No Issue
37	operatorSend	write	Passed	No Issue
38	operatorBurn	write	Passed	No Issue
39	allowance	read	Passed	No Issue
40	approve	write	Passed	No Issue
41	transferFrom	write	Passed	No Issue
42	_mint	internal	Passed	No Issue
43	_send	internal	Passed	No Issue
44	_burn	internal	Passed	No Issue
45	_move	write	Passed	No Issue
46	_approve	internal	Passed	No Issue
47	_callTokensToSend	write	Passed	No Issue
48	_callTokensReceived	write	Passed	No Issue
49	_spendAllowance	internal	Passed	No Issue
50	_beforeTokenTransfer	internal	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

(1) Burn Return Percentage limit is not set: **BKD777Token.sol**

```
function setBurnReturnPercentage(    23633 gas
    uint256 percentage
) public virtual override {
    require(isOperatorFor(_msgSender(), address(this)), "BKD777: caller is not an operator for this contract");
    _burnReturnPercentage = percentage;
}
```

Operators can set the individual Burn Return Percentage to any variable. This might deter investors as they could be wary that these percentages might one day be set to 100% which might affect the Return amount calculations.

Resolution: Consider adding an explicit limit while setting the setBurnReturnPercentage value.
Status: **This issue is fixed in the revised contract code.**

Low

(1) Infinite loop possibility:

Below functions allow the operator to input unlimited wallets. So, the operator must input limited wallets, as inputting excessive wallets might hit the block's gas limit. The operator can accept this risk and can execute this function using limited wallets only.

BKD777Token.sol

operatorBatchTransfer

```
function operatorBatchTransfer(
    address sender,
    address[] memory recipients,
    uint256[] memory amounts,
    bytes memory data,
    bytes memory operatorData
) public virtual override {
    require(isOperatorFor(_msgSender(), sender), "BKD777: caller is not an operator for sender");
    require(recipients.length == amounts.length, "BKD777: recipients and amounts length mismatch");

    for (uint256 i = 0; i < recipients.length; i++) {
        address recipient = recipients[i];
        uint256 amount = amounts[i];

        _send(sender, recipient, amount, data, operatorData, false);
    }
}
```

operatorBatchMint

```
function operatorBatchMint(
    address[] memory recipients,
    uint256[] memory amounts,
    bytes memory data,
    bytes memory operatorData
) public virtual override {
    require(recipients.length == amounts.length, "BKD777: recipients and amounts length mismatch");

    address operator = _msgSender();
    require(operator != address(0), "BKD777: batch mint using the zero address");

    for (uint256 i = 0; i < recipients.length; i++) {
        address recipient = recipients[i];
        uint256 amount = amounts[i];

        require(isOperatorFor(operator, recipient), "BKD777: caller is not an operator for recipient");

        _mint(recipient, amount, data, operatorData);
    }
}
```

operatorBatchBurn

```
function operatorBatchBurn(
    address[] memory holders,
    uint256[] memory amounts,
    bytes memory data,
    bytes memory operatorData
) public virtual override {
    require(holders.length == amounts.length, "BKD777: holders and amounts length mismatch");

    address operator = _msgSender();
    require(operator != address(0), "BKD777: batch burn using the zero address");

    for (uint256 i = 0; i < holders.length; i++) {
        address holder = holders[i];
        uint256 amount = amounts[i];

        require(isOperatorFor(operator, holder), "BKD777: caller is not an operator for holder");

        _burn(holder, amount, data, operatorData);
    }
}
```

RedemptionController.sol

batchAdd

```
function batchAdd(
    address[] memory accounts,
    uint256[] memory amounts,
    bytes[] memory data
) public override
//returns (uint256[] memory plans)
{
    require(hasRole(MANAGER_ROLE, _msgSender()), "RedemptionController: must have manager role");
    require(accounts.length == amounts.length, "RedemptionController: accounts and amounts length mismatch");

    uint64 timestamp = block.timestamp < startsAt() ? uint64(startsAt()): uint64(block.timestamp);

    for (uint256 i = 0; i < accounts.length; i++) {
        _add(accounts[i], amounts[i], timestamp, data[i]);
    }
}
```

BKD777Token.sol

- operatorBatchTransfer
- operatorBatchMint
- operatorBatchBurn
- batchTransfer

Resolution: We suggest specifying some limit on the number of wallets can be used. This will prevent any potential human error.

Status: **This issue is fixed in the revised contract code.**

(2) Function input parameters lack of check:

Variable validation is not performed in below functions:

BKD777Token.sol

- operatorBurnReturn = account
- operatorMint = account
- operatorTransferAnyERC20Token = token , recipient
- operatorBatchTransfer = sender

USDBK777Token.sol

- operatorMint = account
- operatorTransferAnyERC20Token = token , recipient
- operatorBatchTransfer = sender

Resolution: We advise to put validation: integer type variables should be greater than 0 and address type variables should not be address(0).
Status: **This issue is fixed in the revised contract code.**

Very Low / Informational / Best practices:

(1) “external” visibility over “public”: **BKD777Token.sol**

We suggest using “external” visibility instead of “public” if those functions are not being called internally. Although this does not raise any security issue, it is considered a best practice, and it saves some gas as well.

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

Status: Acknowledged.

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- destroy: RedemptionController owner can destroy smart contract.
- operatorMigrateFrom: RedemptionController operator can withdraw any ERC20 token received by the contract.
- destroy: BKD777Token owner can destroy smart contract.
- setBurnReturnAccount: BKD777Token owner can set burn return account address.
- setBurnReturnPercentage: BKD777Token owner can set burn return percentage value.
- setBurnReturnForwardAccount: BKD777Token owner can set burn return forward account address.
- setBurnReturnForwardPercentage: BKD777Token owner can set burn return forward percentage value.
- operatorBurnReturn: BKD777Token operator can burn return token.
- operatorMint: BKD777Token operator can mint and transfer tokens.
- operatorTransferAnyERC20Token: BKD777Token operator can withdraw any ERC20 token received by the contract.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

Conclusion

We were given a contract code in the form of files. And we have used all possible tests based on given objects as files. We have observed 1 medium issue, 2 low issues and some very low level issues in the smart contracts. All the issues have been fixed / acknowledged in the revised code. **So, it's good to go for the mainnet deployment.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Email: audit@EtherAuthority.io

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

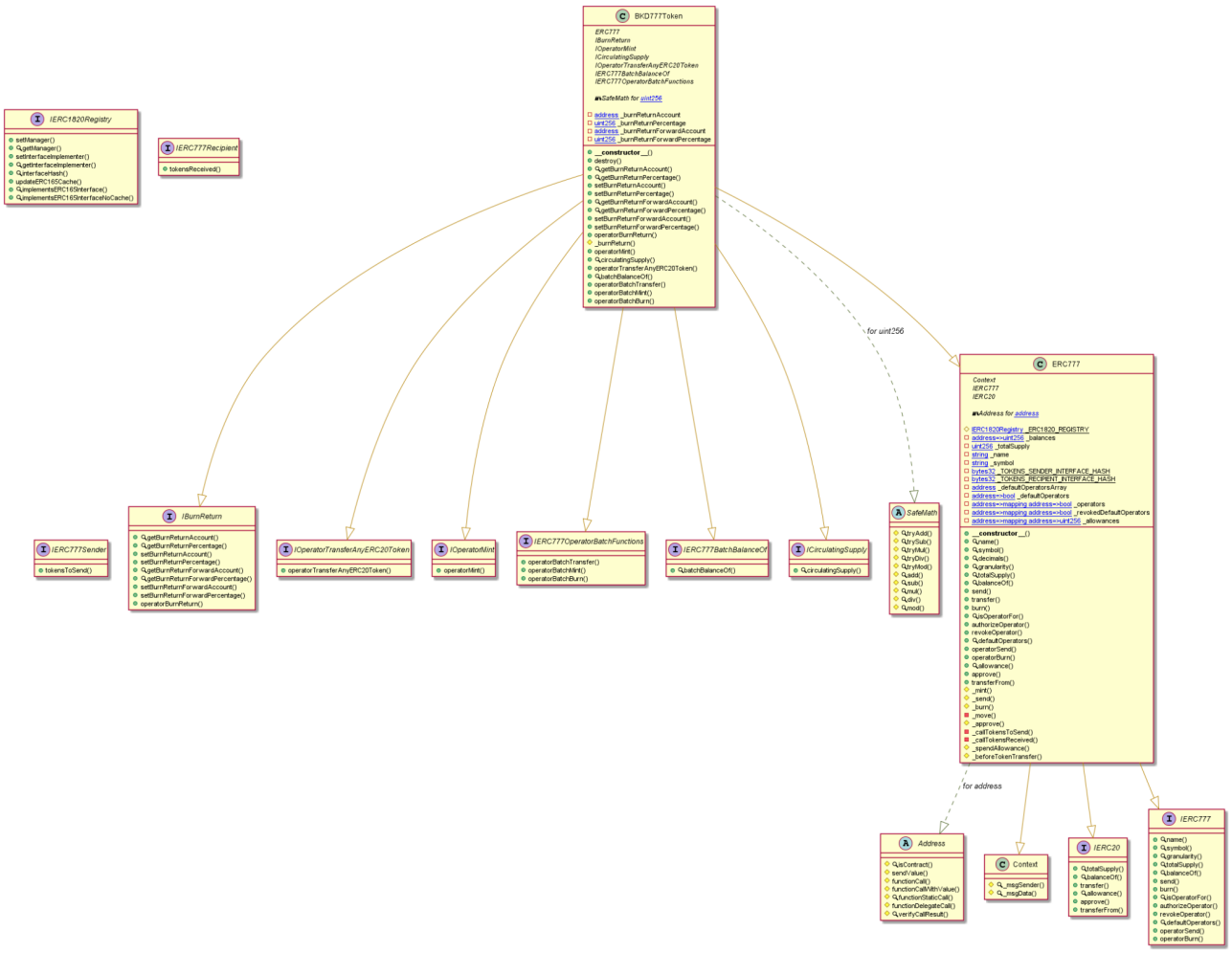
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

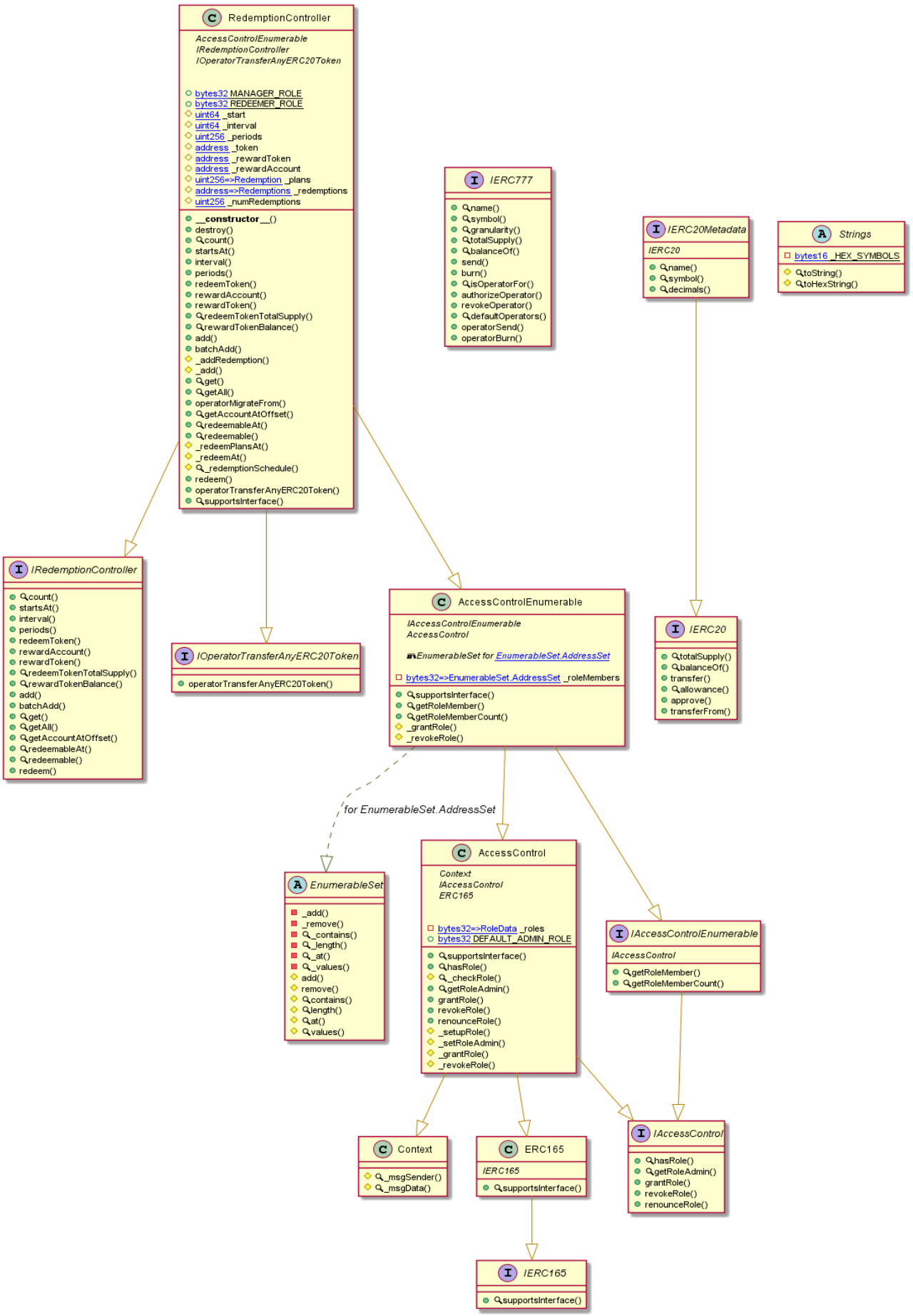
Appendix

Code Flow Diagram - Backters Protocol

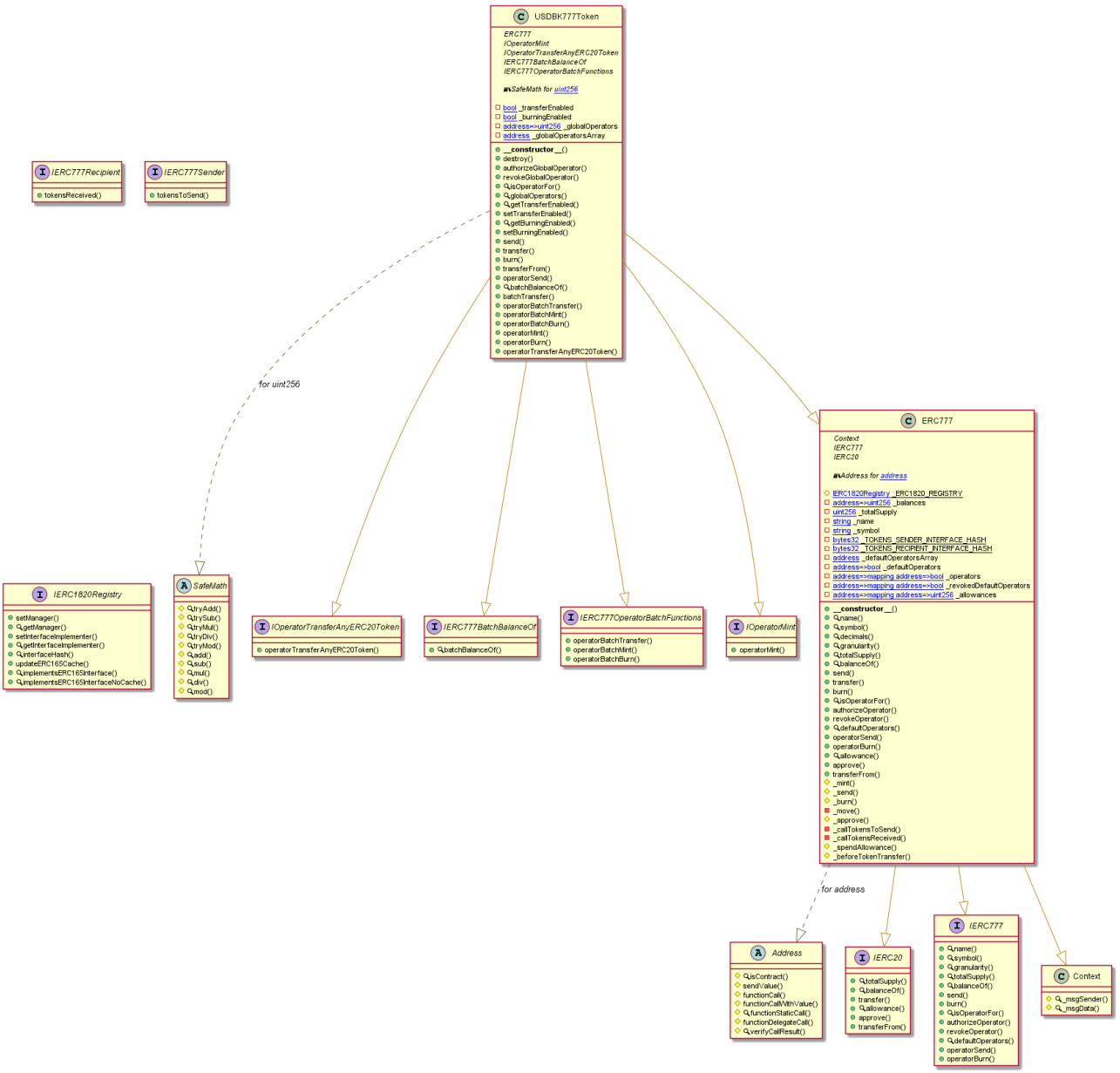
BKD777Token Diagram



RedemptionController Diagram



USDBK777Token Diagram



Slither Results Log

Slither log >> BKD777Token.sol

```
INFO:Detectors:
BKD777Token.constructor(string,string,address[],address,uint256,address,uint256,uint256).name (BKD777Token.sol#1458) shadows:
        - ERC777.name() (BKD777Token.sol#974-976) (function)
        - IERC777.name() (BKD777Token.sol#715) (function)
BKD777Token.constructor(string,string,address[],address,uint256,address,uint256,uint256).symbol (BKD777Token.sol#1459) shadows:
        - ERC777.symbol() (BKD777Token.sol#981-983) (function)
        - IERC777.symbol() (BKD777Token.sol#721) (function)
BKD777Token.constructor(string,string,address[],address,uint256,address,uint256,uint256).defaultOperators (BKD777Token.sol#1460
) shadows:
        - ERC777.defaultOperators() (BKD777Token.sol#1096-1098) (function)
        - IERC777.defaultOperators() (BKD777Token.sol#822) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
BKD777Token.setBurnReturnAccount(address).account (BKD777Token.sol#1499) lacks a zero-check on :
                - _burnReturnAccount = account (BKD777Token.sol#1502)
BKD777Token.setBurnReturnForwardAccount(address).account (BKD777Token.sol#1521) lacks a zero-check on :
                - _burnReturnForwardAccount = account (BKD777Token.sol#1524)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in ERC777._burn(address,uint256,bytes,bytes) (BKD777Token.sol#1283-1307):
        External calls:
        - _callTokensToSend(operator,from,address(0),amount,data,operatorData) (BKD777Token.sol#1293)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (BKD777Token.sol#1366)
        State variables written after the call(s):
        - _balances[from] = fromBalance - amount (BKD777Token.sol#1301)
        - _totalSupply -= amount (BKD777Token.sol#1303)
Reentrancy in ERC777._send(address,address,uint256,bytes,bytes,bool) (BKD777Token.sol#1256-1274):
        External calls:
        - _callTokensToSend(operator,from,to,amount,userData,operatorData) (BKD777Token.sol#1269)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (BKD777Token.sol#1366)
        State variables written after the call(s):
        - _move(operator,from,to,amount,userData,operatorData) (BKD777Token.sol#1271)
                - _balances[from] = fromBalance - amount (BKD777Token.sol#1322)
```

```
        External calls:
        - ERC777(name,symbol,defaultOperators) (BKD777Token.sol#1467)
                - _ERC1820_REGISTRY.setInterfaceImplementer(address(this),keccak256(bytes)(ERC777Token),address(this)) (BKD777T
oken.sol#967)
                - _ERC1820_REGISTRY.setInterfaceImplementer(address(this),keccak256(bytes)(ERC20Token),address(this)) (BKD777To
ken.sol#968)
        State variables written after the call(s):
        - _burnReturnAccount = burnReturnAccount (BKD777Token.sol#1470)
        - _burnReturnForwardAccount = burnReturnForwardAccount (BKD777Token.sol#1475)
        - _burnReturnForwardPercentage = burnReturnForwardPercentage (BKD777Token.sol#1478)
        - _burnReturnPercentage = burnReturnPercentage (BKD777Token.sol#1472)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ERC777._burn(address,uint256,bytes,bytes) (BKD777Token.sol#1283-1307):
        External calls:
        - _callTokensToSend(operator,from,address(0),amount,data,operatorData) (BKD777Token.sol#1293)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (BKD777Token.sol#1366)
        Event emitted after the call(s):
        - Burned(operator,from,amount,data,operatorData) (BKD777Token.sol#1305)
        - Transfer(from,address(0),amount) (BKD777Token.sol#1306)
Reentrancy in BKD777Token._burnReturn(address,uint256,bytes,bytes) (BKD777Token.sol#1548-1588):
        External calls:
        - _send(account,_burnReturnForwardAccount,forwardAmount,data,operatorData,false) (BKD777Token.sol#1574)
                - IERC777Recipient(implementer).tokensReceived(operator,from,to,amount,userData,operatorData) (BKD777Token.sol#
1392)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (BKD777Token.sol#1366)
        - _burn(account,burnAmount,data,operatorData) (BKD777Token.sol#1579)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (BKD777Token.sol#1366)
        Event emitted after the call(s):
```

```
        - _mint(msg.sender,initialSupply,,) (BKD777Token.sol#1481)
                - IERC777Recipient(implementer).tokensReceived(operator,from,to,amount,userData,operatorData) (BKD777Token.sol#
1392)
        - ERC777(name,symbol,defaultOperators) (BKD777Token.sol#1467)
                - _ERC1820_REGISTRY.setInterfaceImplementer(address(this),keccak256(bytes)(ERC777Token),address(this)) (BKD777T
oken.sol#967)
                - _ERC1820_REGISTRY.setInterfaceImplementer(address(this),keccak256(bytes)(ERC20Token),address(this)) (BKD777To
ken.sol#968)
        Event emitted after the call(s):
        - Minted(operator,account,amount,userData,operatorData) (BKD777Token.sol#1243)
                - _mint(msg.sender,initialSupply,,) (BKD777Token.sol#1481)
        - Transfer(address(0),account,amount) (BKD777Token.sol#1244)
                - _mint(msg.sender,initialSupply,,) (BKD777Token.sol#1481)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (BKD777Token.sol#481-501) uses assembly
        - INLINE ASM (BKD777Token.sol#493-496)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (BKD777Token.sol#365-367) is never used and should be removed
Address.functionCall(address,bytes,string) (BKD777Token.sol#375-381) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (BKD777Token.sol#394-400) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (BKD777Token.sol#408-419) is never used and should be removed
Address.functionDelegateCall(address,bytes) (BKD777Token.sol#454-456) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (BKD777Token.sol#464-473) is never used and should be removed
Address.functionStaticCall(address,bytes) (BKD777Token.sol#427-429) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (BKD777Token.sol#437-446) is never used and should be removed
Address.sendValue(address,uint256) (BKD777Token.sol#340-345) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (BKD777Token.sol#481-501) is never used and should be removed
```

Slither log >> RedemptionController.sol

```
INFO:Detectors:
RedemptionController.redeemToken(address).at (RedemptionController.sol#1250) lacks a zero-check on :
                - _token = at (RedemptionController.sol#1252)
RedemptionController.rewardAccount(address).account (RedemptionController.sol#1259) lacks a zero-check on :
                - _rewardAccount = account (RedemptionController.sol#1261)
RedemptionController.rewardToken(address).at (RedemptionController.sol#1268) lacks a zero-check on :
                - _rewardToken = at (RedemptionController.sol#1270)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
Reentrancy in RedemptionController._redeemAt(uint256,uint64) (RedemptionController.sol#1548-1584):
        External calls:
        - IERC20(rewardToken()).transfer(account,amountValue) (RedemptionController.sol#1563)
        - IERC20(rewardToken()).transferFrom(rewardAccount(),account,amountValue) (RedemptionController.sol#1565)
        - IERC777(redeemToken()).operatorBurn(account,amount,redemption.data,) (RedemptionController.sol#1568)
        Event emitted after the call(s):
        - RedemptionDistributed(account,plan,amount,redemption.amount,span,redemption.data) (RedemptionController.sol#1573-1580
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
RedemptionController.constructor(uint64,uint64,uint256,address,address) (RedemptionController.sol#1166-1189) uses timestamp for
omparisons
        Dangerous comparisons:
        - require(bool,string)(startTimestamp == 0 || startTimestamp > block.timestamp,RedemptionController: start time is befo
 current time) (RedemptionController.sol#1173)
RedemptionController.startsAt(uint64) (RedemptionController.sol#1213-1218) uses timestamp for comparisons
        Dangerous comparisons:
```

```
EnumerableSet.values(EnumerableSet.AddressSet) (RedemptionController.sol#326-335) is never used and should be removed
EnumerableSet.values(EnumerableSet.Bytes32Set) (RedemptionController.sol#260-262) is never used and should be removed
EnumerableSet.values(EnumerableSet.UintSet) (RedemptionController.sol#399-408) is never used and should be removed
Strings.toHexString(uint256) (RedemptionController.sol#810-821) is never used and should be removed
Strings.toString(uint256) (RedemptionController.sol#785-805) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (RedemptionController.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.
/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable RedemptionController._start (RedemptionController.sol#1141) is not in mixedCase
Variable RedemptionController._interval (RedemptionController.sol#1142) is not in mixedCase
Variable RedemptionController._periods (RedemptionController.sol#1143) is not in mixedCase
Variable RedemptionController._token (RedemptionController.sol#1144) is not in mixedCase
Variable RedemptionController._rewardToken (RedemptionController.sol#1145) is not in mixedCase
Variable RedemptionController._rewardAccount (RedemptionController.sol#1146) is not in mixedCase
Variable RedemptionController._plans (RedemptionController.sol#1162) is not in mixedCase
Variable RedemptionController._redemptions (RedemptionController.sol#1163) is not in mixedCase
Variable RedemptionController._numRedemptions (RedemptionController.sol#1164) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Redundant expression "this (RedemptionController.sol#845)" inContext (RedemptionController.sol#839-848)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (RedemptionController.sol#962-964)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (RedemptionController.sol#975-977)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (RedemptionController.sol#993-997)
getRoleMember(bytes32,uint256) should be declared external:
        - AccessControlEnumerable.getRoleMember(bytes32,uint256) (RedemptionController.sol#1104-1106)
getRoleMemberCount(bytes32) should be declared external:
        - AccessControlEnumerable.getRoleMemberCount(bytes32) (RedemptionController.sol#1112-1114)
destroy() should be declared external:
        - RedemptionController.destroy() (RedemptionController.sol#1191-1194)
count() should be declared external:
        - RedemptionController.count() (RedemptionController.sol#1199-1201)
startsAt(uint64) should be declared external:
        - RedemptionController.startsAt(uint64) (RedemptionController.sol#1213-1218)
interval(uint64) should be declared external:
        - RedemptionController.interval(uint64) (RedemptionController.sol#1227-1231)
```

Slither log >> USDBK777Token.sol

```
INFO:Detectors:
USDBK777Token.constructor(string,string,address[],uint256).name (USDBK777Token.sol#1440) shadows:
        - ERC777.name() (USDBK777Token.sol#945-947) (function)
        - IERC777.name() (USDBK777Token.sol#535) (function)
USDBK777Token.constructor(string,string,address[],uint256).symbol (USDBK777Token.sol#1441) shadows:
        - ERC777.symbol() (USDBK777Token.sol#952-954) (function)
        - IERC777.symbol() (USDBK777Token.sol#541) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in ERC777._burn(address,uint256,bytes,bytes) (USDBK777Token.sol#1253-1277):
        External calls:
        - _callTokensToSend(operator,from,address(0),amount,data,operatorData) (USDBK777Token.sol#1263)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#133
6)
        State variables written after the call(s):
        - _balances[from] = fromBalance - amount (USDBK777Token.sol#1271)
        - _totalSupply -= amount (USDBK777Token.sol#1273)
Reentrancy in ERC777._send(address,address,uint256,bytes,bytes,bool) (USDBK777Token.sol#1226-1244):
        External calls:
        - _callTokensToSend(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#1239)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#133
6)
        State variables written after the call(s):
        - _move(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#1241)
                - _balances[from] = fromBalance - amount (USDBK777Token.sol#1292)
                - _balances[to] += amount (USDBK777Token.sol#1294)
Reentrancy in USDBK777Token.constructor(string,string,address[],uint256) (USDBK777Token.sol#1439-1458):
        External calls:
        - ERC777(name,symbol,operators) (USDBK777Token.sol#1445)
                - _ERC1820_REGISTRY.setInterfaceImplementer(address(this),keccak256(bytes)(ERC777Token),address(this)) (USDBK77
7Token.sol#938)
                - _ERC1820_REGISTRY.setInterfaceImplementer(address(this),keccak256(bytes)(ERC20Token),address(this)) (USDBK777
Token.sol#939)
        State variables written after the call(s):
        - _burningEnabled = false (USDBK777Token.sol#1448)
        - _globalOperators[operators[i]] = i (USDBK777Token.sol#1452)
        - _globalOperatorsArray = operators (USDBK777Token.sol#1450)
```

```
        - _transferEnabled = false (USDBK777Token.sol#1447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ERC777._burn(address,uint256,bytes,bytes) (USDBK777Token.sol#1253-1277):
        External calls:
        - _callTokensToSend(operator,from,address(0),amount,data,operatorData) (USDBK777Token.sol#1263)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#133
6)
        Event emitted after the call(s):
        - Burned(operator,from,amount,data,operatorData) (USDBK777Token.sol#1275)
        - Transfer(from,address(0),amount) (USDBK777Token.sol#1276)
Reentrancy in ERC777._mint(address,uint256,bytes,bytes,bool) (USDBK777Token.sol#1194-1215):
        External calls:
        - _callTokensReceived(operator,address(0),account,amount,userData,operatorData,requireReceptionAck) (USDBK777Token.sol#
1211)
                - IERC777Recipient(implementer).tokensReceived(operator,from,to,amount,userData,operatorData) (USDBK777Token.so
l#1362)
        Event emitted after the call(s):
        - Minted(operator,account,amount,userData,operatorData) (USDBK777Token.sol#1213)
        - Transfer(address(0),account,amount) (USDBK777Token.sol#1214)
Reentrancy in ERC777._send(address,address,uint256,bytes,bytes,bool) (USDBK777Token.sol#1226-1244):
        External calls:
        - _callTokensToSend(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#1239)
                - IERC777Sender(implementer).tokensToSend(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#133
6)
        Event emitted after the call(s):
        - Sent(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#1296)
                - _move(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#1241)
        - Transfer(from,to,amount) (USDBK777Token.sol#1297)
                - _move(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#1241)
Reentrancy in USDBK777Token.constructor(string,string,address[],uint256) (USDBK777Token.sol#1439-1458):
        External calls:
        - _mint(msg.sender,initialSupply,,) (USDBK777Token.sol#1456)
                - IERC777Recipient(implementer).tokensReceived(operator,from,to,amount,userData,operatorData) (USDBK777Token.so
l#1362)
        - ERC777(name,symbol,operators) (USDBK777Token.sol#1445)
                - _ERC1820_REGISTRY.setInterfaceImplementer(address(this),keccak256(bytes)(ERC777Token),address(this)) (USDBK77
```

```
        - Transfer(from,to,amount) (USDBK777Token.sol#1297)
                - _move(operator,from,to,amount,userData,operatorData) (USDBK777Token.sol#1241)
Reentrancy in USDBK777Token.constructor(string,string,address[],uint256) (USDBK777Token.sol#1439-1458):
        External calls:
        - _mint(msg.sender,initialSupply,,) (USDBK777Token.sol#1456)
                - IERC777Recipient(implementer).tokensReceived(operator,from,to,amount,userData,operatorData) (USDBK777Token.so
l#1362)
        - ERC777(name,symbol,operators) (USDBK777Token.sol#1445)
                - _ERC1820_REGISTRY.setInterfaceImplementer(address(this),keccak256(bytes)(ERC777Token),address(this)) (USDBK77
7Token.sol#938)
                - _ERC1820_REGISTRY.setInterfaceImplementer(address(this),keccak256(bytes)(ERC20Token),address(this)) (USDBK777
Token.sol#939)
        Event emitted after the call(s):
        - Minted(operator,account,amount,userData,operatorData) (USDBK777Token.sol#1213)
                - _mint(msg.sender,initialSupply,,) (USDBK777Token.sol#1456)
        - Transfer(address(0),account,amount) (USDBK777Token.sol#1214)
                - _mint(msg.sender,initialSupply,,) (USDBK777Token.sol#1456)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (USDBK777Token.sol#197-217) uses assembly
        - INLINE ASM (USDBK777Token.sol#209-212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

Solidity Static Analysis

BKD777Token.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ERC777.(string,string,address[]): Could potentially lead to re-entrancy vulnerability.
more
Pos: 953:4:

Low level calls:

Use of "delegatecall": should be avoided whenever possible. External code, that is called can change the state of the calling contract and send ether from the caller's balance. If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 471:50:

Selfdestruct:

Use of selfdestruct: Can block calling contracts unexpectedly. Be especially careful if this contract is planned to be used by other contracts (i.e. library contracts, interactions). Selfdestruction of the callee contract can leave callers in an inoperable state.
more
Pos: 1487:8:

Gas & Economy

Gas costs:

Gas requirement of function BKD777Token.batchBalanceOf is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1627:4:

Gas costs:

Gas requirement of function BKD777Token.operatorBatchTransfer is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1643:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1693:8:

Miscellaneous

Constant/View/Pure functions:

ERC777._beforeTokenTransfer(address,address,address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 1434:4:

Similar variable names:

BKD777Token.operatorBatchMint(address[],uint256[],bytes,bytes) : Variables have very similar names "_operators" and "operator".
Pos: 1669:8:

Similar variable names:

BKD777Token.operatorBatchMint(address[],uint256[],bytes,bytes) : Variables have very similar names "_operators" and "operator".
Pos: 1670:16:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1651:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 1087:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 258:19:

RedemptionController.sol

Security

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1614:85:

Selfdestruct:

Use of selfdestruct: Can block calling contracts unexpectedly. Be especially careful if this contract is planned to be used by other contracts (i.e. library contracts, interactions). Selfdestruction of the callee contract can leave callers in an inoperable state.
more
Pos: 1193:8:

Gas & Economy

Gas costs:

Gas requirement of function RedemptionController.add is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1287:4:

Gas costs:

Gas requirement of function RedemptionController.batchAdd is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1299:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1311:8:

Miscellaneous

Similar variable names:

RedemptionController._addRedemption(uint64,uint256,address,uint256,bytes) : Variables have very similar names "_periods" and "period". Note: Modifiers are currently not considered by this static analysis.
Pos: 1324:41:

Similar variable names:

RedemptionController._addRedemption(uint64,uint256,address,uint256,bytes) : Variables have very similar names "_plans" and "plan". Note: Modifiers are currently not considered by this static analysis.
Pos: 1323:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1242:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1626:8:

Delete from dynamic array:

Using "delete" on an array leaves a gap. The length of the array remains the same. If you want to remove the empty position you need to shift items manually and update the "length" property.
more
Pos: 150:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1601:44:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1601:45:

USDBK777Token.sol

Security

Selfdestruct:

Use of selfdestruct: Can block calling contracts unexpectedly. Be especially careful if this contract is planned to be used by other contracts (i.e. library contracts, interactions). Selfdestruction of the callee contract can leave callers in an inoperable state.
more
Pos: 106:9:

Gas & Economy

Gas costs:

Gas requirement of function USDBK777Token.destroy is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 104:5:

Gas costs:

Gas requirement of function USDBK777Token.revokeGlobalOperator is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 140:5:

Gas costs:

Gas requirement of function USDBK777Token.send is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 224:5:

Gas costs:

Gas requirement of function ERC777.burn is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 251:5:

Gas costs:

Gas requirement of function USDBK777Token.batchBalanceOf is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 300:5:

Gas costs:

Gas requirement of function ERC777.operatorBurn is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 467:5:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 367:9:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 427:9:

Miscellaneous

Constant/View/Pure functions:

IOperatorTransferAnyERC20Token.operatorTransferAnyERC20Token(address,address,uint256) : Potentially should be constant/view/pure but is not.
more
Pos: 14:4:

Similar variable names:

USDBK777Token.operatorBatchMint(address[],uint256[],bytes,bytes) : Variables have very similar names "recipient" and "recipients".
Pos: 397:33:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 336:9:

Solhint Linter

BKD777Token.sol

```
BKD777Token.sol:83:18: Error: Parse error: missing ';' at '{'
BKD777Token.sol:96:18: Error: Parse error: missing ';' at '{'
BKD777Token.sol:256:18: Error: Parse error: missing ';' at '{'
BKD777Token.sol:282:18: Error: Parse error: missing ';' at '{'
BKD777Token.sol:1300:18: Error: Parse error: missing ';' at '{'
BKD777Token.sol:1321:18: Error: Parse error: missing ';' at '{'
BKD777Token.sol:1414:22: Error: Parse error: missing ';' at '{'
```

RedemptionController.sol

```
RedemptionController.sol:2:1: Error: Compiler version >=0.8.2 <0.9.0
does not satisfy the r semver requirement
RedemptionController.sol:330:9: Error: Avoid using inline assembly.
It is acceptable only in rare cases
RedemptionController.sol:403:9: Error: Avoid using inline assembly.
It is acceptable only in rare cases
RedemptionController.sol:1166:5: Error: Explicitly mark visibility in
function (Set ignoreConstructors to true if using solidity >=0.7.0)
RedemptionController.sol:1173:57: Error: Avoid to make time-based
decisions in your business logic
decisions in your business logic
RedemptionController.sol:1614:86: Error: Avoid to make time-based
decisions in your business logic
```

USDBK777Token.sol

```
USDBK777Token.sol:2:1: Error: Compiler version >=0.8.2 <0.9.0 does
not satisfy the r semver requirement
USDBK777Token.sol:80:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
USDBK777Token.sol:230:9: Error: Check result of "send" call
```

Overall Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io