Cyberscope

Audit Report

Web23

August 2022

Github	https://github.com/rahul-web23/HbarSmartContract
Commit	20efdd10cd7146915fbe0a7f49192660201b2d26
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	DomainWeb23
Compiler Version	v0.8.11+commit.d7f03943
Github	https://github.com/rahul-web23/HbarSmartContrac
Commit	20efdd10cd7146915fbe0a7f49192660201b2d26
Unit Tests	https://github.com/cyberscope-io/audits/tree/main/web23/tests
Testing Deploy	https://bscscan.com/token/0x6d2E683793c7c3dBf7b832Da52165D3ea0E37746

Audit Updates

Initial Audit	12th August 2022
Corrected phase 1	18th August 2022
Corrected phase 2	24th August 2022

Source Files

Filename	**SHA256**
DomainWeb23.sol	f5b1d9620c9efff7a10ee105c13df2b770fc2e9f970bdb4cb19b6cc160aba1e5
HederaResponseCodes.sol	23d77e84bd8c92ed5f5f52491cc83abae4d690cdcba547130dd5d24f56c6035a
HederaTokenService.sol	3a5047606a5e170530b55eddae4cca72ce3d8f59e8fe8b63c0b30275529b79d6

IHederaTokenService.sol	081b85a32145744dd00d13943562c729387bb6141d9f36c758f73d25b1eaba41

Audit Scope

The audit focuses on the DomainWeb23 contract. The token processing operations like mint, associate are delegated to an external contract that is out of the audit scope. The payment methods in the DomainWeb23 are not calling back the sender, but the delegation calls to HederaTokenService address are passing the sender's address. We assume that the contract owner is a trusted address and does not handle the receive payment method. Hence, the contract is not vulnerable for a reentrance attack by the DomainWeb23 methods. On the other hand, it may produce potential vulnerabilities if the HederaTokenService is calling back the original sender.

Unit Tests

As an integral part of the auditing process, 15 scenarios were scripted to test the contract's functionality. Additionally, a scenario has been implemented where multiple users try to buy one domain.

Implementation

https://github.com/cyberscope-io/audits/tree/main/web23/tests

Business Scenarios

- Should receive a payment and mint successfully (1,6)
- Should setDomainAsset successfully (1,2)
- Should return empty value in an unregistered domain (1)
- Should check if domain exist
- Should check if sender is the owner
- Should blacklist a domain (3)
- Should not allow an unregistered domain
- Should allow a registered domain (4,7)
- Should update the site address (5,7)
- Should update the site address only from owner (5)
- Should not allow changing an unregistered site address (5)
- Should book a domain when payment received (6)
- Should get all registered domains (8)
- Should check that domain exists (9)
- Should receive multiple payments (7,9,10)

Multiple Users Scenario

- Register multiple wallets the same domain

Contract Analysis

🔴 Critical 🟠 Medium ⚪ Minor 🔵 Pass

Severity	Code	Description	Status
🟠	BLC	Business Logic Concern	Acknowledged
⚪	DPC	Domain Purchase Cost	
⚪	ZAA	Zero Address Association	
⚪	L04	Conformance to Solidity Naming Conventions	Acknowledged
⚪	L02	State Variables could be Declared Constant	Unresolved
⚪	L05	Unused State Variable	Unresolved

BLC - Business Logic Concern

Criticality	medium
Location	contract.sol#L85
Status	Acknowledged

Description

The contract is using a variable that is alway set with the zero value. This variable is passed to the ‘mintToken()’ method. So the contract always executes the ‘mintToken()’ method with zero amount. The specification of the mintToken states the following:

```
@param amount Applicable to tokens of type FUNGIBLE_COMMON. The amount to mint
to the Treasury Account.
Amount must be a positive non-zero number represented in the lowest denomination
of the token. The new supply must be lower than 2^63.
```

The actual argument of the ‘mintToken()’ method comes into conflict with the method specification.

```
uint64 _amount=0;
string memory domName=hashToDomainInfo[_hash].domainName;
uint256 ii=indexOf(domName,".");
address domainOwner=hashToDomainInfo[_hash].domainOwnerAddress;
string memory parentBtld=substring(domName,ii+1);
    (int response, uint64 newTotalSupply, int64[] memory serialNumbers) =
HederaTokenService.mintToken(btldToTokenAddress[parentBtld], _amount,
_metadata);
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

Team Update (18/08)

This is a default behavior of HTS, NFTMinting Engine, where 0 is passed as amount while NFT is minted.

DPC - Domain Purchase Cost

Criticality	minor
Location	contract.sol#L15
Status	

Description

MIN_DOMAIN_PRICE is fixed to the value 1. As a result the minimum cost of one domain will be `1/10^18` native tokens, so the cost of a purchase is almost zero.

```
uint256 MIN_DOMAIN_PRICE=1;
```

Recommendation

The contract could implement a fees setter method and take in account the native token’s decimals.

ZAA - Zero Address Association

Criticality	minor
Location	contract.sol#L152
Status	

Description

The contract is allowing token association with the zero address. According to the mintToken specification, if a token does not exist, the transaction results in INVALID_TOKEN_ID. This may happen if the caller provide a domain that both the top level domain and the second level domain are not registered.

```
address
btldToken=btldToTokenAddress[parentBtld]==address(0x0)?btldToTokenAddress[substr
ing(parentBtld,indexOf(parentBtld,".")+1)]:btldToTokenAddress[parentBtld];
...
HederaTokenService.associateToken(msg.sender, btldToken);
```

According to the HederaTokenService specification:

```
///  Associates the provided account with the provided tokens. Must be signed by
the provided
///  Account's key or called from the accounts contract key
///  If the provided account is not found, the transaction will resolve to
INVALID_ACCOUNT_ID.
```

Recommendation

The contract could embed a check for not allowing associations with the zero address.

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contracts/DomainWeb23.sol#L15
Status	Unresolved

Description

Constant state variables should be declared constant to save gas.

```
MIN_DOMAIN_PRICE
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/DomainWeb23.sol#L64,68,72,76,95,130,167,171,180,203,208,220,226,230,238,242,15
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_blacklistedAddress
_whitelistedAddress
_hash
_metadata
_domainNames
_userAddress
_domainName
_siteAddress
_btld
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions.

Team Update (18/08)

We have used _variable for function parameters

L05 - Unused State Variable

Criticality	minor
Location	contracts/DomainWeb23.sol#L9
Status	Unresolved

Description

There are segments that contain unused state variables.

```
DomainWeb23
...
```

Recommendation

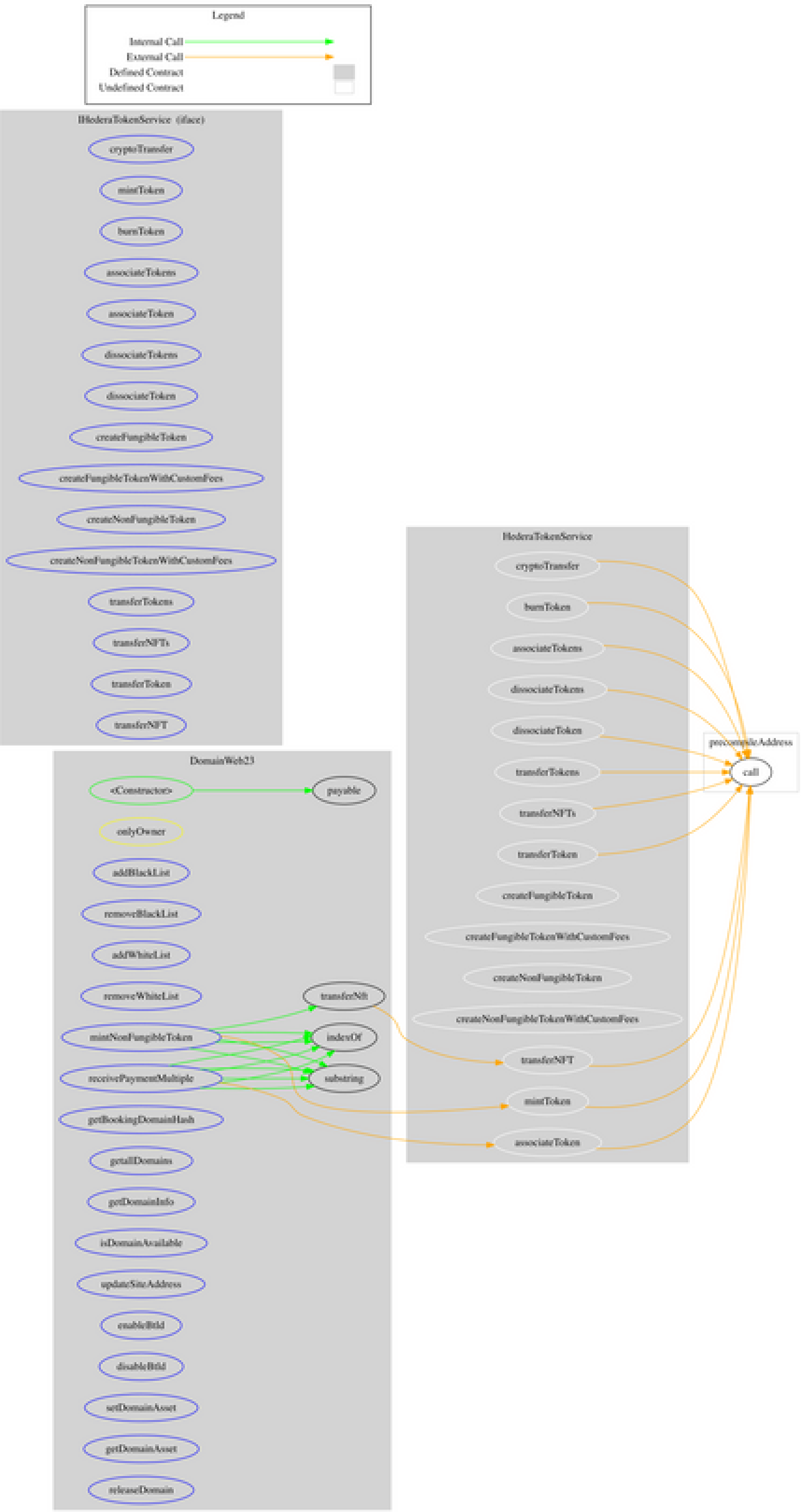
Remove unused state variables.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
DomainWeb23	Implementation	HederaTokenService		
	<Constructor>	Public	✓	-
	substring	Private		
	addBlackList	External	✓	onlyOwner
	removeBlackList	External	✓	onlyOwner
	addWhiteList	External	✓	onlyOwner
	removeWhiteList	External	✓	onlyOwner
	indexOf	Private		
	mintNonFungibleToken	External	✓	onlyOwner
	receivePaymentMultiple	External	Payable	-
	getBookingDomainHash	External		-
	getallDomains	External		-
	getDomainInfo	External		-
	transferNft	Internal	✓	
	isDomainAvailable	External		-
	updateSiteAddress	External	✓	-
	enableBtId	External	✓	onlyOwner
	disableBtId	External	✓	onlyOwner
	setDomainAsset	External	✓	-
	getDomainAsset	External		-
	releaseDomain	External	✓	onlyOwner
HederaResponseCodes	Implementation			
HederaTokenService	Implementation	HederaResponseCodes		
	cryptoTransfer	Internal	✓	
	mintToken	Internal	✓	

	burnToken	Internal	✓	
	associateTokens	Internal	✓	
	associateToken	Internal	✓	
	dissociateTokens	Internal	✓	
	dissociateToken	Internal	✓	
	createFungibleToken	Internal	✓	
	createFungibleTokenWithCustomFees	Internal	✓	
	createNonFungibleToken	Internal	✓	
	createNonFungibleTokenWithCustomFees	Internal	✓	
	transferTokens	Internal	✓	
	transferNFTs	Internal	✓	
	transferToken	Internal	✓	
	transferNFT	Internal	✓	
IHederaTokenService	Interface			
	cryptoTransfer	External	✓	-
	mintToken	External	✓	-
	burnToken	External	✓	-
	associateTokens	External	✓	-
	associateToken	External	✓	-
	dissociateTokens	External	✓	-
	dissociateToken	External	✓	-
	createFungibleToken	External	Payable	-
	createFungibleTokenWithCustomFees	External	Payable	-
	createNonFungibleToken	External	Payable	-
	createNonFungibleTokenWithCustomFees	External	Payable	-
	transferTokens	External	✓	-
	transferNFTs	External	✓	-
	transferToken	External	✓	-
	transferNFT	External	✓	-

Contract Flow



Summary

Web23 implements domain registration functionality based on web3. This audit focuses on the potential vulnerabilities, business logic concerns and suggested improvements. A batch of scenarios and unit tests have been implemented in order to validate the business logic and the flows.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Cyberscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Cyberscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Cyberscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Cyberscope team disclaims any liability for the resulting losses.

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Cyberscope and Coinscope are aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.

The Cyberscope team

https://www.cyberscope.io