Cyberscope

Audit Report

Web23

August 2022

Github	https://github.com/rahul-web23/HbarSmartContract
Commit	f21eaaed3666e6b8607ec1b4099ff7cf6326e2ee
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	DomainWeb23
Compiler Version	v0.8.11+commit.d7f03943
Optimization	200 runs
Github	https://github.com/rahul-web23/HbarSmartContrac
Commit	f21eaaed3666e6b8607ec1b4099ff7cf6326e2ee
Unit Tests	https://github.com/cyberscope-io/audits/tree/main/web23/tests
Testing Deploy	https://testnet.bscscan.com/token/0xB7898999b87b28DA5be9899dEc7C1bB1df2FC93c

Audit Updates

Initial Audit	12th August 2022
Corrected	

Source Files

Filename	**SHA256**
DomainWeb23.sol	a3e755ff958e37087e7013755a8fb3ff13fc34ce091cf2af6fc2e74241935ee4
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
🟠	BLC	Business Logic Concern
⚪	CO	Code Optimization
⚪	CR	Code Repetition
🟠	RVC	Return Value Conflict
⚪	RIC	Range Index Check
⚪	DSC	Domain Sanity Check
⚪	ASC	Address Sanity Check
⚪	RSV	Redundant State Variable
⚪	PAE	Precondition Abort Explanation
🔴	BFA	Booking Functionality Abuse
⚪	USV	Unaccessible State Variable
⚪	MMN	Misleading Method Name
⚪	L01	Public Function could be Declared External
⚪	L02	State Variables could be Declared Constant
⚪	L04	Conformance to Solidity Naming Conventions
⚪	L05	Unused State Variable
⚪	L09	Dead Code Elimination
⚪	L11	Unnecessary Boolean equality

●	L14	Uninitialized Variables in Local Scope

BLC - Business Logic Concern

Criticality	medium
Location	contract.sol#L96

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.

CO - Code Optimization

Criticality	minor
Location	contract.sol#L212

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

```
function getBookingDomainHash(bytes32 _hash) public view returns(bool){
    if(bytes(hashToDomainInfo[_hash].domainName).length>0){
        return true;
    }
    else{
        return false;
    }
}
```

Recommendation

The method could be deducted to a more compact version

```
function getBookingDomainHash(bytes32 _hash) public view returns(bool){
    return bytes(hashToDomainInfo[_hash].domainName).length>0
}
```

CR - Code Repetition

Criticality	minor
Location	contract.sol#L132,169

Description

There are code segments that are repetitive in the contract. Those segments increase the code size of the contract unnecessarily.

The ‘receivePaymentMultiple()’ is the multiplied version of ‘receivePayment()’.

Recommendation

The internal implementation of ‘receivePaymentMultiple()’ could reuse the ‘receivePayment()’ instead of repeating the code segments.

RVC - Return Value Conflict

Criticality	medium
Location	contract.sol#L63

Description

The implementation of indexOf returns the zero value in two cases.

1. If the delim does not exist in the string.
2. If the delim exists in the first index of the string.

As a result, it produces wrong assumptions to the caller.

```
function indexOf(string memory str, string memory delim)
    private
    pure
    returns (uint256)
{
    bytes memory strBytes = bytes(str);
    for (uint256 i = 0; i < strBytes.length; i++) {
        if (strBytes[i] == bytes(delim)[0]) {
            return i;
        }
    }
    return 0;
}
```

Recommendation

The contract should return a different value in every case. For instance, it could use the uint256 maximum values to represent the non-existence.

RIC - Range Index Check

Criticality	minor
Location	contract.sol#L46

Description

The method will produce an underflow subtraction if the provided ‘startIndex’ is greater than the length of the string.

```
function substring(string memory str, uint256 startIndex)
    private
    pure
    returns (string memory)
{
    bytes memory strBytes = bytes(str);
    bytes memory result = new bytes(strBytes.length - startIndex);
    for (uint256 i = startIndex; i < strBytes.length; i++) {
        result[i - startIndex] = strBytes[i];
    }
    return string(result);
}
```

Recommendation

The contract should check that the ‘startIndex’ bounds are between the string’s length.

DSC - Domain Sanity Check

Criticality	minor
Location	contract.sol#L98,137

Description

The contract should check if the dot delimiter exists before proceeding. Otherwise it may produce unexpected values in the state variables.

```
uint256 ii = indexOf(domName, ".");
address domainOwner = hashToDomainInfo[_hash].domainOwnerAddress;
string memory parentBtld = substring(domName, ii + 1);
```

Recommendation

The contract should properly check the variables according to the required specifications.

ASC - Address Sanity Check

Criticality	minor
Location	contract.sol#L307

Description

The btldToTokenAddress variable points a top level domain to an address. This address is used to mint tokens. The methods that are using the ‘btldToTokenAddress’ require not to map on a zero address.

```
function enableBtld(string memory _btld, address _tokenAddress)
    external
    onlyOwner
{
    isBtldEnabled[_btld] = true;
    btldToTokenAddress[_btld] = _tokenAddress;
}
```

Recommendation

The ‘btldToTokenAddress’ should not be allowed to point on the zero address.

RSV - Redundant State Variable

Criticality	minor
Location	contract.sol#L126

Description

The contract is using a private variable ‘addressToDomains’ to store values, but it does not contain any method to access it back. As a result, the ‘addressToDomains’ is redundant since it is not accessible.

```
addressToDomains[domainInfo.domainOwnerAddress].push(domName);
```

Recommendation

The contract could either remove the ‘`addressToDomains`’ variable or add an accessor method.

PAE - Precondition Abort Explanation

Criticality	minor
Location	contract.sol#L324

Description

The contract validates that the caller of the method should be the domain owner. In case of violation, the transaction is reverted without explanation.

```
require(
    msg.sender == nameToDomainInfo[_domainName].domainOwnerAddress,
    ""
);
```

Recommendation

The contract could describe the failure reason.

BFA - Booking Functionality Abuse

Criticality	critical
Location	contract.sol#L132,169

Description

The contract offers two methods for purchasing a domain. A single and a multiple version. The domains are not limited by price, as a result, any user can unlimitedly buy domains without paying.

Recommendation

The contract could limit the purchasing functionality. It could add an intuitive price for each domain or it could set a maximum amount of domains per address.

USV - Unaccessible State Variable

Criticality	minor
Location	contract.sol#L31

Description

The contract stores the initialization token address to the contract’s state. The property tokenAddress is stored as private and it is not accessed by the contract.

```
constructor(address _tokenAddress) {
    tokenAddress = _tokenAddress;
    owner = payable(msg.sender);
    isBtldEnabled["hbar"] = true;
    btldToTokenAddress["hbar"] = tokenAddress;
}
```

Recommendation

The contract could either remove the ‘tokenAddress’ property or add an public accessor method.

MMN - Misleading Method Name

Criticality	minor
Location	contract.sol#L279

Description

The method name isDomainAvailable(string) intuitively means that it returns true if a domain is available and false otherwise. In the implementation, it yields the opposite values.

```
function isDomainAvailable(string memory _domainName)
    public
    view
    returns (bool)
{
    return isDomainBooked[_domainName];
}
```

Recommendation

The contract could yield the values that are explained intuitively by the method name.

L01 - Public Function could be Declared External

Criticality	minor
Location	contracts/DomainWeb23.sol#L104,132,167,176,185,208,213

Description

Public functions that are never called by the contract should be declared external to save gas.

```
updateSiteAddress
isDomainAvailable
getDomainInfo
getallDomains
getBookingDomainHash
receivePaymentMultiple
receivePayment
```

Recommendation

Use the external attribute for functions never called from the contract.

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contracts/DomainWeb23.sol#L15

Description

Constant state variables should be declared constant to save gas.

```
_tokenIds
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contracts/DomainWeb23.sol#L74,104,132,167,176,185,208,213,225,230,234,242 contracts/HederaTokenService.sol#L11

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.

```
precompileAddress
_domainName
_assethash
_btld
_tokenAddress
_siteAddress
_userAddress
_hash
_domainNames
...
```

Recommendation

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

L05 - Unused State Variable

Criticality	minor
Location	contracts/HederaResponseCodes.sol#L7,8,9,10,11,12,13,14,15,16,17,18,19,20,22,23,24,25,26,27,28,30,31,32,33,34,36,37,38,39,41,42,43,44,45,46,47,49,50,51,53,54,56,57,59,60,61,62,63,64,65,67,68,70,71,72,73,75,76,77,78,79,81,82,83,84,85,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,129,130,131,133,134,135,136,137,138,139,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,183,184,185,186,187,188,189,190,191,192,193,194,195,196 contracts/HederaTokenService.sol#L13,14,15,16,17,18,19 contracts/DomainWeb23.sol#L15

Description

There are segments that contain unused state variables.

```
_tokenIds
PAUSE_KEY_TYPE
FEE_SCHEDULE_KEY_TYPE
SUPPLY_KEY_TYPE
WIPE_KEY_TYPE
FREEZE_KEY_TYPE
KYC_KEY_TYPE
ADMIN_KEY_TYPE
MESSAGE_SIZE_TOO_LARGE
...
```

Recommendation

Remove unused state variables.

L09 - Dead Code Elimination

Criticality	minor
Location	contracts/HederaTokenService.sol#L93,64,146,169,187,201,24,132,125,235,251,221

Description

Functions that are not used in the contract, and make the code's size bigger.

```
transferTokens
transferToken
transferNFTs
dissociateTokens
dissociateToken
cryptoTransfer
createNonFungibleTokenWithCustomFees
createNonFungibleToken
createFungibleTokenWithCustomFees
...
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contracts/DomainWeb23.sol#L104,132

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
success == true
```

Recommendation

Remove the equality to the boolean constant.

L14 - Uninitialized Variables in Local Scope

Criticality	minor
Location	contracts/DomainWeb23.sol#L150,90,116

Description

The are variables that are defined in the local scope and are not initialized.

```
domainInfo
```

Recommendation

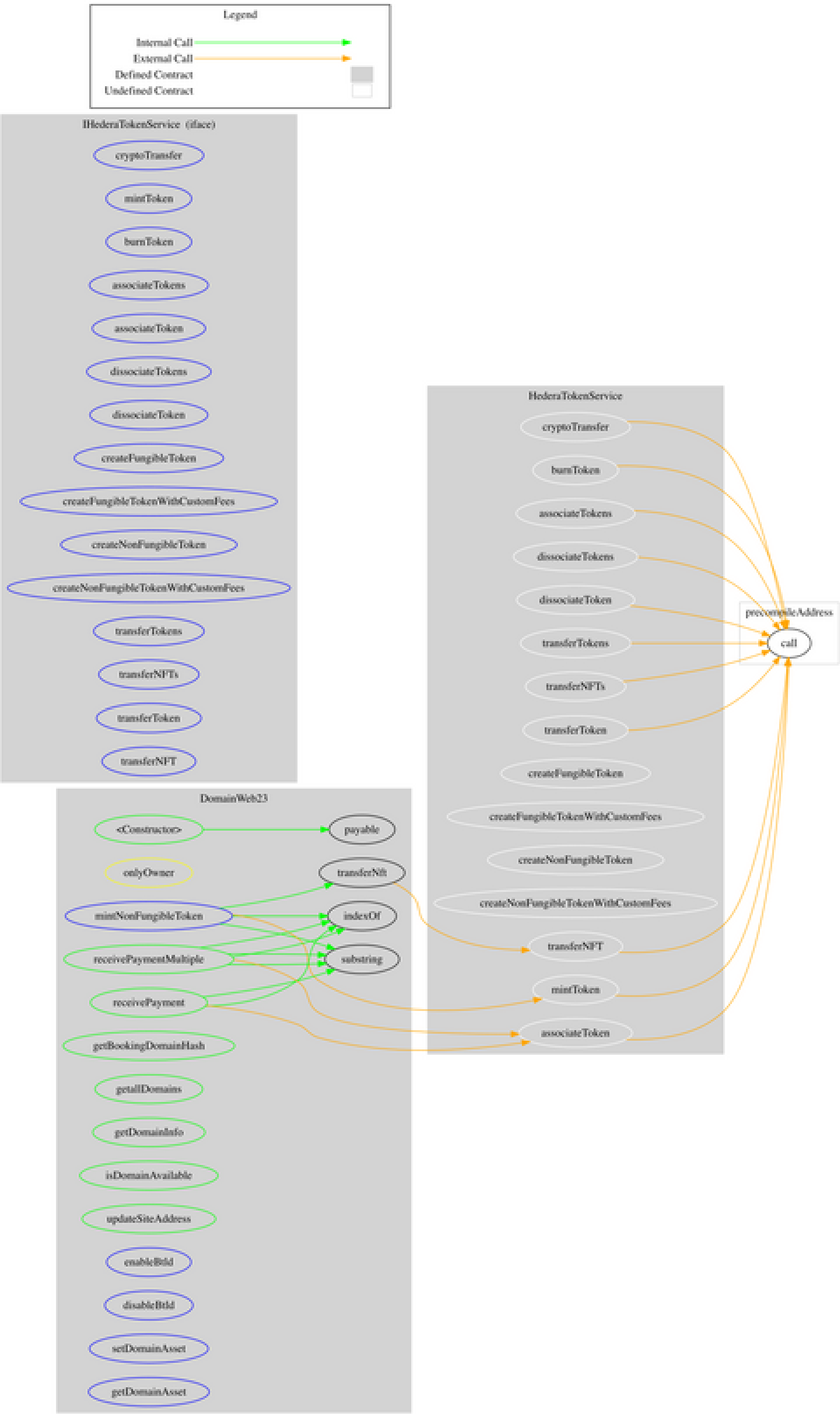
All the local scoped variables should be initialized.

Contract Functions

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