ERC1155CONTRACT
SELLCONTRACT

RESEARCH

SERCAN ÇELENK

29 July 2021

CONTENTS

1

HEBYS enables users to discover, create and trade digital assets.

It is an NFT marketplace that provides. The main purpose of the marketplace is to maximize user experience and complexity minimizes it.

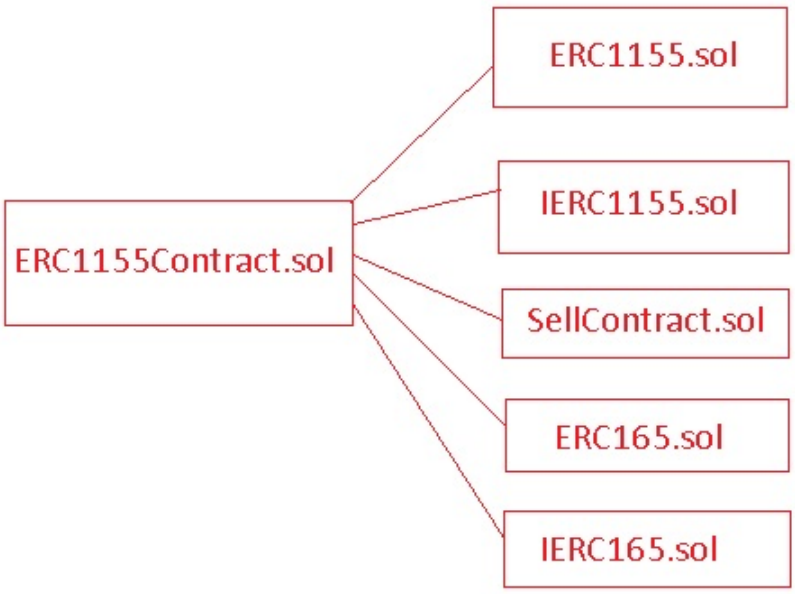
This report;

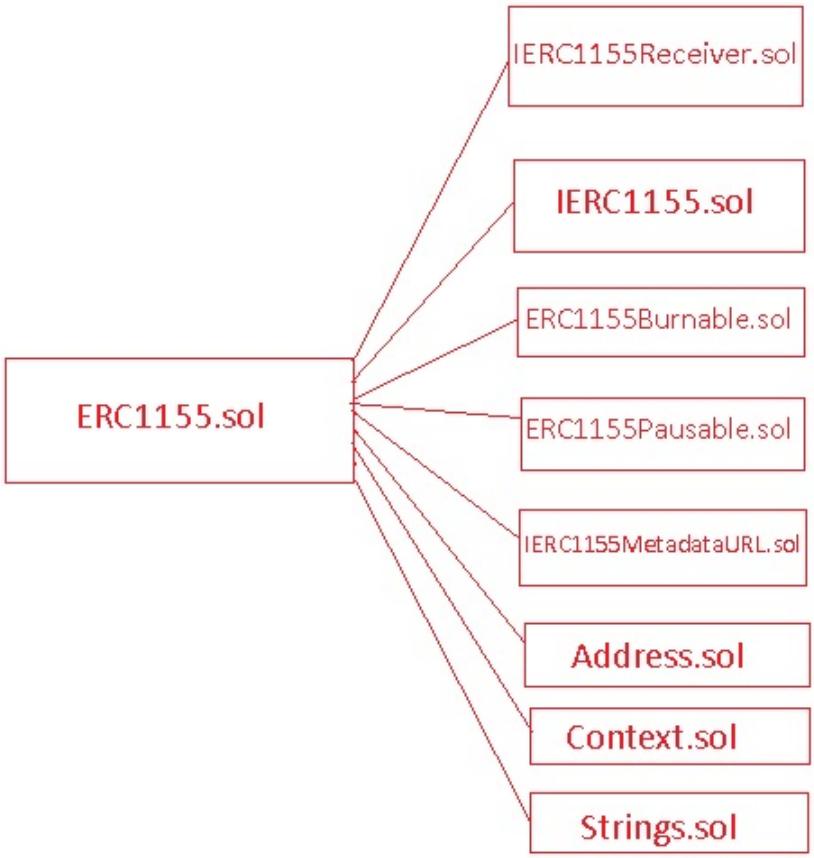
- Continuous development and improvement of the project
- Software Development Lifecycle (SDLC)
- Vulnerabilities and security vulnerabilities

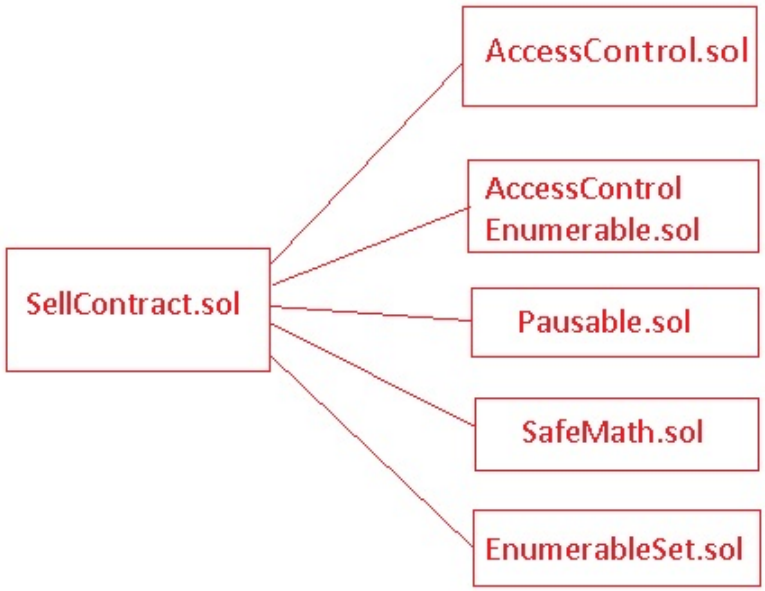
It has been prepared for the purpose of gaining knowledge on many subjects such as

With this compiled report, a framework has been drawn for the future of smart contracts and low-level factors have been determined.

2 File Dependency Graph




ERC1155.sol
IERC1155Receiver.sol
IERC1155.sol
ERC1155Burnable.sol
ERC1155Pausable.sol
IERC1155MetadataURL.sol
Address.sol
Context.sol
Strings.sol


SellContract.sol
AccessControl.sol
AccessControl
Enumerable.sol
Pausable.sol
SafeMath.sol
EnumerableSet.sol

3 Findings

3.1 Integer Division

When integer division occurs, the number is rounded to the nearest integer. If more accuracy is required, use a multiplier or provide the numerator and denominator values is better. These stored values help to calculate the denominator result in off-chain mode

When I reviewed the contract called SellContract.sol, the SafeMath library was called directly from openzeppelin. This is similar to calling a contract within the contract and carries various risks such as exploit leakage. We multiply ourselves in our master contract We have to write SafeMath. Then we can call again from anywhere we want. For instance;

```
library SafeMathKutuphanesi {
    function add(uint a, uint b) internal pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    function sub(uint a, uint b) internal pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    //sercancelenk
    function mul(uint a, uint b) internal pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    function div(uint a, uint b) internal pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}
```

3.2 Forwarding

For the IDO planned in the 4th quarter of 2021; Funds must be kept in a multisig contract at the start of the IDO. In this way, funds cannot be moved until the end of the IDO. A single signature when moved

It does not require the signature of more than one person. In the worst case scenario, let's assume that IDO is done with the smart contract we have. In this IDO, 3 days after the funds are collected in the contract, if the private key of the contract owner is seized by various hacking methods or physically; All funds can be transferred to another wallet.

or by calling the destructive function, it may cause irreversible damage to the contract. Therefore, instead of the standard smart contract, multisig wallet can be used in collective funding such as IDO. contract must be used. In this way, the signatures of the two contract holders to be selected (private) Without the key, no operation can be performed.

3.3 Tradeoff

Interfaces and abstract contracts are instrumental in providing a customizable and reusable approach to smart contracts. Although interfaces are similar to abstract contracts, they lack some functionality. For example, they can't access storage or inherit contracts. When I look at our ERC1155Contract.sol contract, there is only an interface function called IERC165.

3.4 Contract Activity

Monitoring smart contracts is an important way to secure them. In this way, it is possible to monitor all contract transactions, but message calls are not recorded in the blockchain. Therefore, only the input parameters remain visible instead of the actual change made. 'incident' means keeping a record of something that occurs in the contract, one of the methods. Recorded events remain on the blockchain along with other contract data and become available for future auditing.

```
3  contract TEMAvakfi{
4      mapping(address=>uint) balances;
5      function eventMessage() payable public{
6          balances[msg.sender]+=msg.value;
7      }
8  }
9
10 contract tarimBakanligi{
11     function fidanAl() payable public{
12         tarimBakanligi.eventMessage.value(msg.value/20)();
13     }
14 }
15
16 TEMAvakfi.eventMessage();
```

3.5 Killswitch

An overlooked problem on the testnet can cause irreversible results when switching to the Mainnet. Theft of assets, Flash Loan, DoS attacks, Gas Price Impact etc. The first thing to do in such cases is to freeze the contract, namely killswitch. This Although the process can be done with the Pauseable library, it is still possible to add another external contract to the contract. Instead of calling a contract, it would be more convenient to write a function embedded in the main contract.

```
1
2  void passive(){
3      require_auth(_self);
4      setStatusDelegate(0);
5  }
6
7  void active(){
8      require_auth(_self);
9      setStatusDelegate(1);
10 }
11 //sercancelenk
12 void process(){
13     assert(setStatusDelegate().passive == 0, "Contract is passive.!");)
14 }
```

3.6 Numerical Overflow

Values may overflow if boundary conditions are incorrectly controlled while performing arithmetic operations. This results in the loss of users' assets. Using uint64t for arithmetic operations Instead, we should place the SafeMath library that we will write directly into the main contract.

3.7 Low-Level Call

Low-level calls have problems with return values. For example, the user enters a letter instead of a number, or the contract addresses return an incorrect value.

```
//Operator authorization inquiry for sales contract from creator contract
(bool success, bytes memory result) =
    creatorContractAddress.call(
        abi.encodeWithSignature(
            "isApprovedForAll(address,address)",
            _ownerAddress,
            contractAddress
        )
    );
```

The part I've underlined is a low-level call. As a solution; if the invoked contracts are in separate files, they can be imported, or with the called contract, the calling contract can be imported into a separate abstract contract. If importing, the bytecode will increase, which may decrease performance.