Blockchain

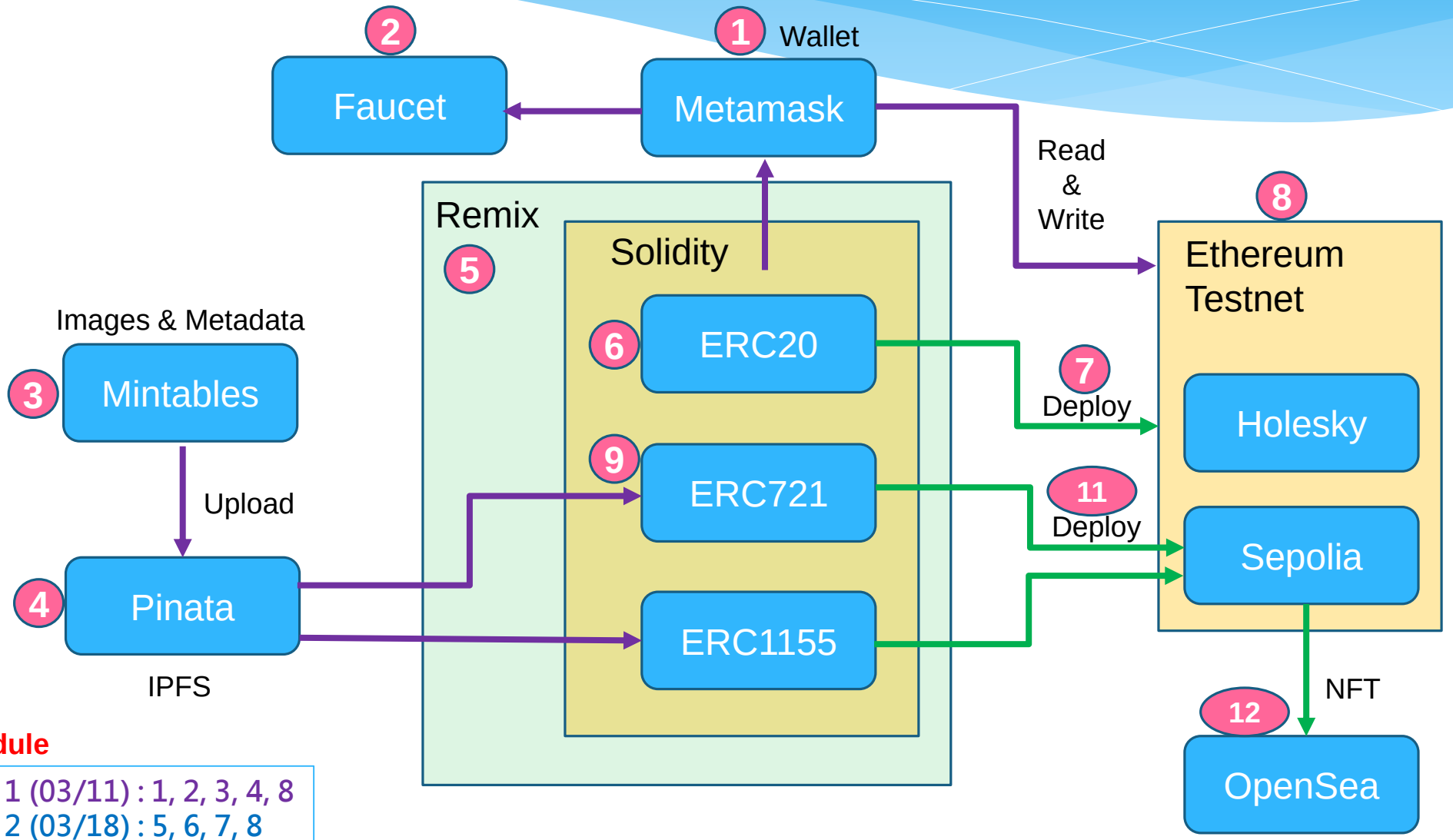
Create and Deploy an ERC20 Token

2024

目錄

- What is an ERC-20 Token
- Install the Metamask extension
- Create an ERC-20 Token
- Deploy an ERC-20 Token
- Import Token
- Transfer Token
- Modify and Extend Token
- Security and Access control
- Ether Token

Architecture Flow



Schedule

week 1 (03/11) : 1, 2, 3, 4, 8
week 2 (03/18) : 5, 6, 7, 8
week 3 (03/25) : 5, 9, 11,12

What is an ERC-20 Token?

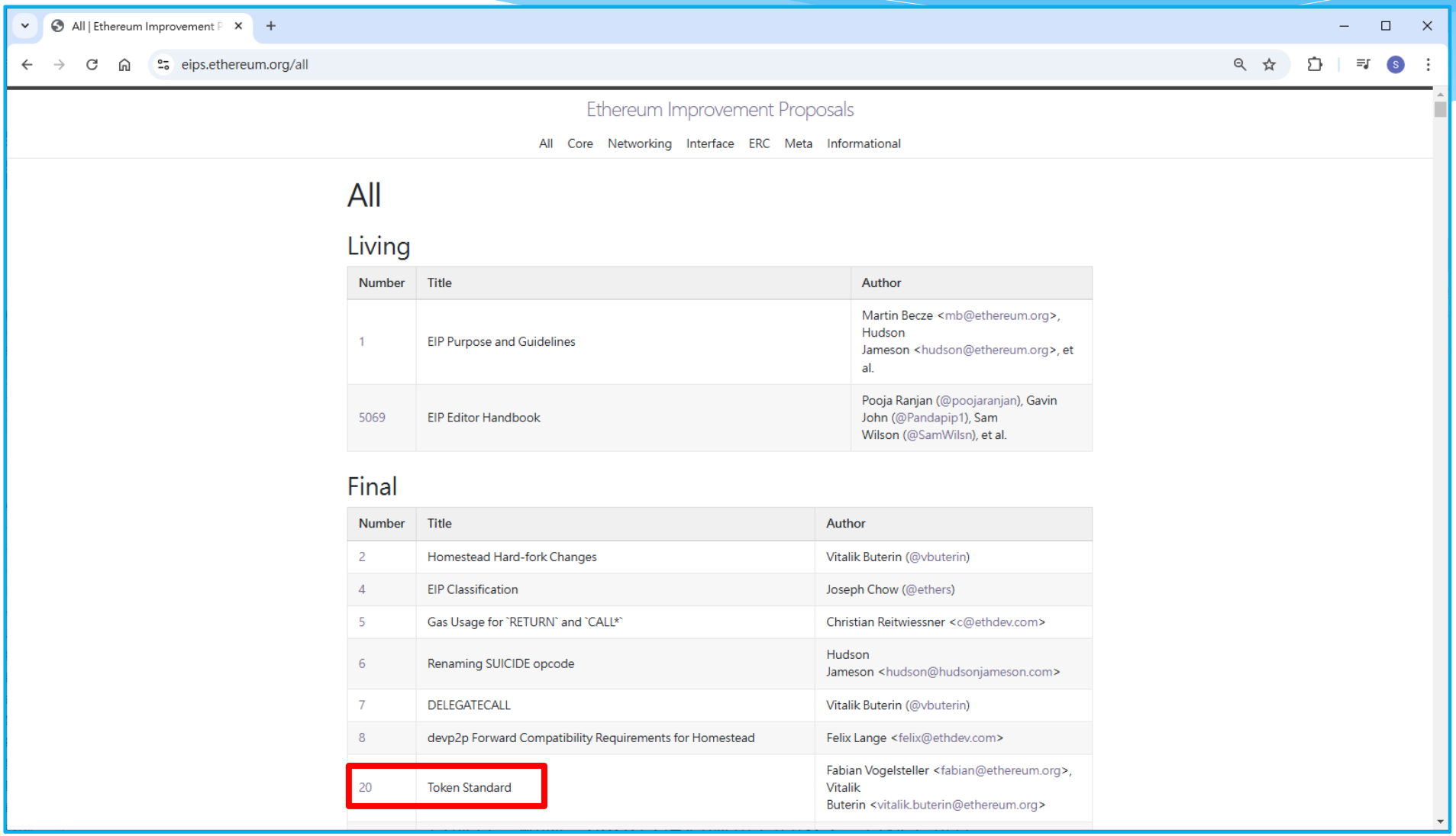
- 在ERC-20出現之前，每個要產生新代幣(Token)的人都必須重新撰寫程式碼，這代表著所有代幣都彼此不同。
  - 開發人員想要使用某一個代幣，他們必須了解該代幣的整個智能合約程式碼，原因就在於缺少建立新代幣的任何特定結構或文件指引。
  - 對於錢包和交易平台來說相當困難，因在任何應用程式上添加不同類型的代幣，需要開發人員仔細檢查每個代幣的程式碼並理解它，才能夠在其平台上處理這些代幣。

- 目前錢包和交易所使用 ERC-20 標準，係將各種標準化代幣整合到其平台上，並促進 ERC-20 代幣與其他代幣之間的互動可輕鬆地進行交換。

What is an ERC-20 Token?

- ERC-20 的ERC代表以太坊(Ethereum)請求意見，而20則是提案時的識別號碼。
- ERC-20 主旨在改善以太坊網路。
- ERC-20 已是最重要的 ERC 之一。它已成為在以太坊區塊鏈網路上編寫智能合約的技術標準，並用於實作建立新代幣(Token) 。
- ERC-20 是包含所有在以太坊的代幣，都必須遵循的一組標準規則。
- ERC-20 將代幣定義為基於區塊鏈的、可以發送/接收且具有價值的資產。
- ERC-20 在許多方面與比特幣和萊特幣相似。然而，最顯著的區別是，ERC-20 不是在自己的區塊鏈網路上運行，而是在以太坊的區塊鏈網路上運行，並使用 Gas 作為交易費用。

Ethereum Improvement Proposals

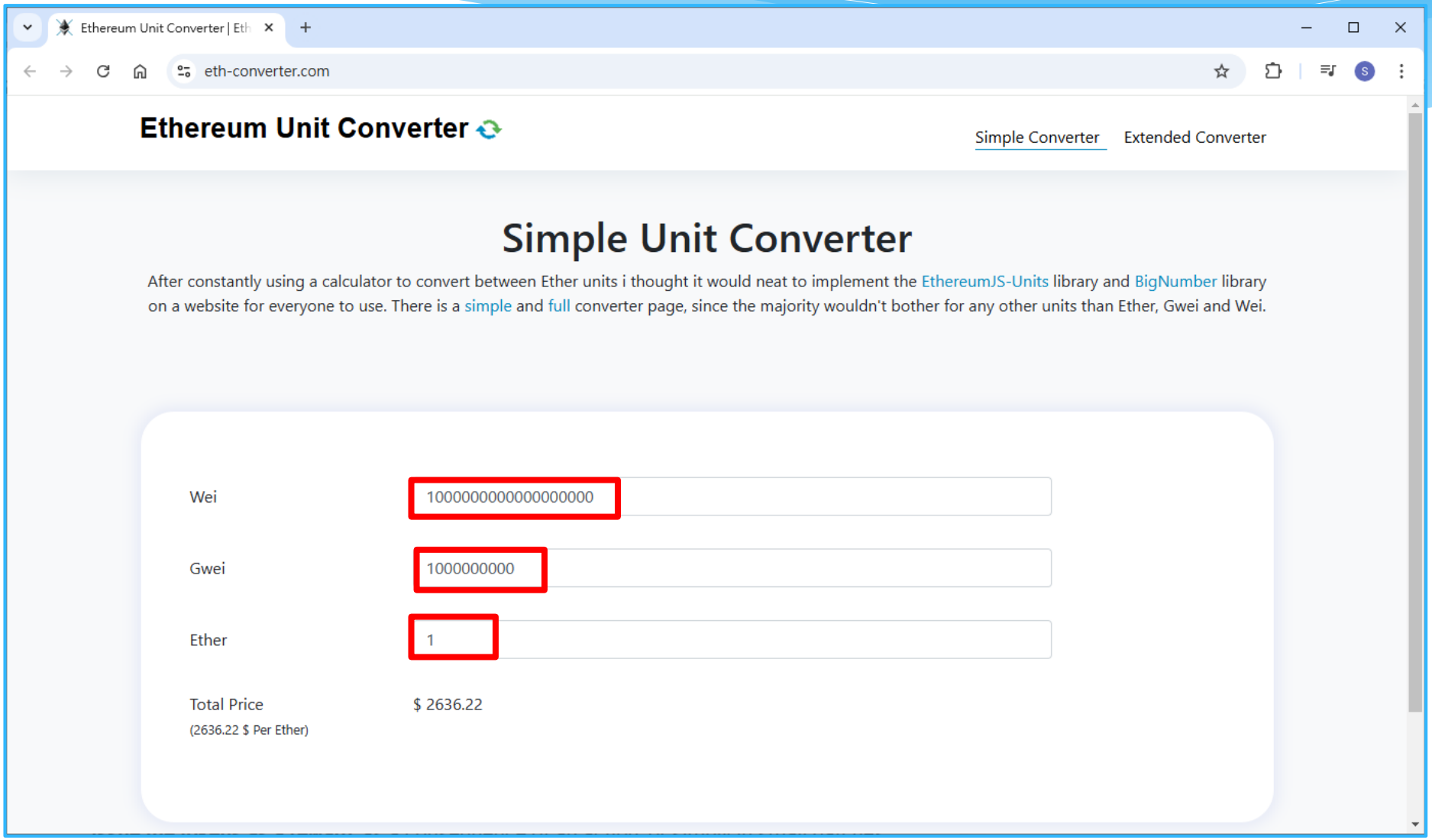


https://eips.ethereum.org/all

Ethereum Gas Fee

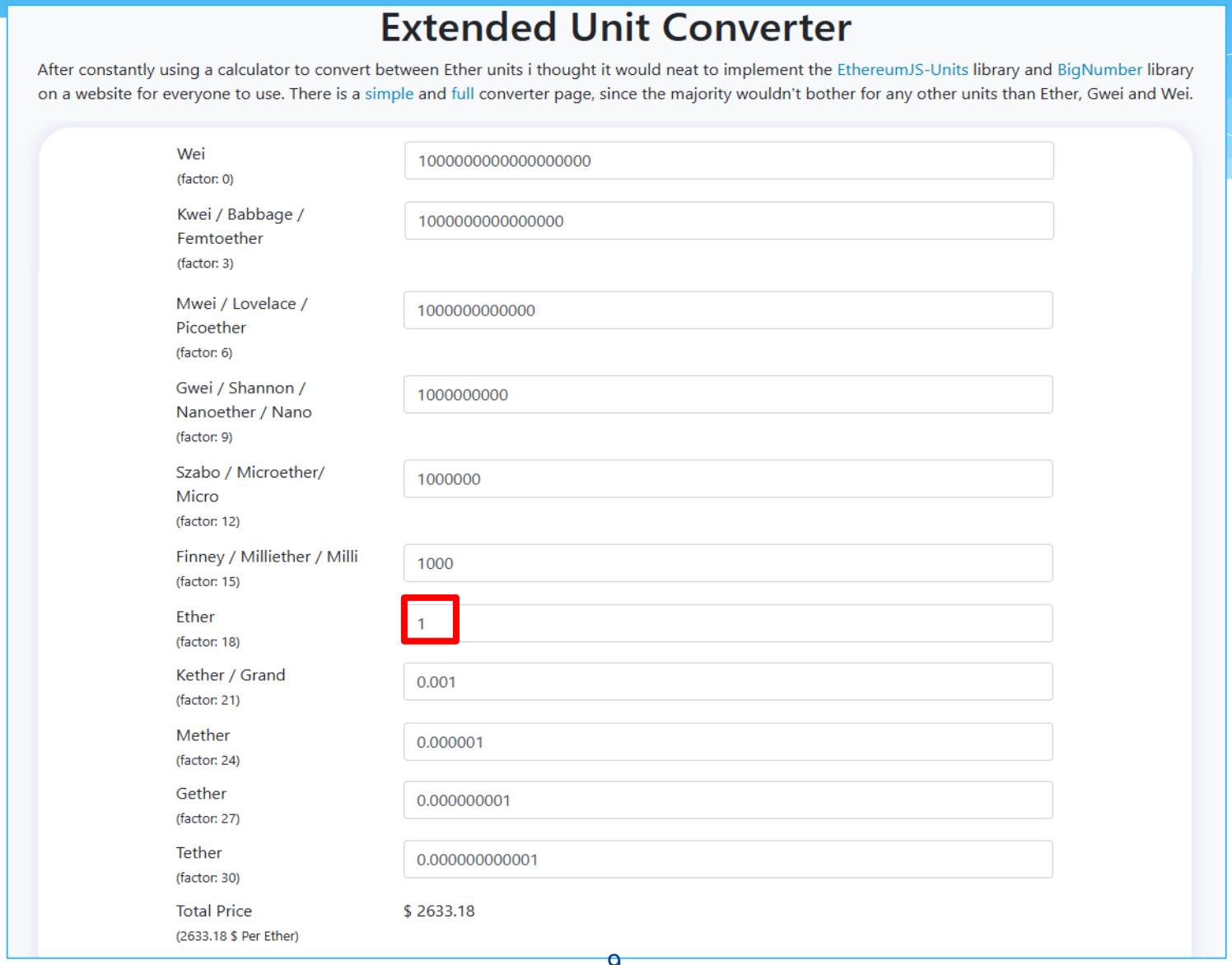
Unit	Alternative Name	Wei Value	Gwei Value	Ether Value
Wei	Wei	1 Wei	10^-9 Gwei	10^-18 ETH
Kwei	Babbage	10^3 Wei	10^-6 Gwei	10^-15 ETH
Mwei	Lovelace	10^6 Wei	10^-3 Gwei	10^-12 ETH
Gwei	Shannon	10^9 Wei	1 Gwei	10^-9 ETH
Twei	Szabo	10^12 Wei	10^3 Gwei	10^-6 ETH
Pwei	Finney	10^15 Wei	10^6 Gwei	10^-3 ETH
Ether	—	10^18 Wei	10^9 Gwei	1 ETH

Ethereum Unit Converter



https://eth-converter.com/

Ethereum Unit Converter



https://eth-converter.com/extended-converter

Key Points about ERC-20

- **標準化功能**：這表示它們具有通用的規則和功能清單。這包括如何轉移代幣、如何批准交易、用戶如何存取有關代幣的數據以及代幣的總供應量。
- **智能合約和 DeFi**（Decentralized Finance）：使用智能合約可以實現複雜金融作業的自動化和執行。這對 DeFi 平台至關重要，因為這些代幣可以代表各種金融工具，例如貸款或流動性池中的股權。
- ***互通性***：由於代幣遵循相同的標準，因此它們可以輕鬆互換，並且可以與以太坊網路上其他符合 ERC-20 的代幣和應用程式無縫協作。這種標準化簡化了建立新代幣的過程，並使它們立即與現有錢包、交易所和其他服務相容。
- **使用案例**：ERC-20 代幣可以代表廣泛的資產或公用事業。例如，貸款抵押品、流動性挖礦中的生息資產以及在去中心化自治組織（DAO）中授予投票權的治理代幣。
- **可轉讓性和交易**：這些代幣可以作為付款從一個帳戶轉移到另一個帳戶，類似於比特幣等加密貨幣，並且可以在各種加密貨幣交易所進行交易。

ERC-20 - 6 Mandatory Functions

- **totalSupply**: A method that defines the total supply of your tokens; when this limit is reached, the smart contract will refuse to create new tokens.
- **balanceOf:** A method that returns the number of tokens a wallet address has.
- **transfer:** A method that takes a certain amount of tokens from the total supply and gives it to a user.
- **transferFrom:** Another type of transfer method that is used to transfer tokens between users.
- **approve:** This method verifies whether a smart contract is allowed to allocate a certain amount of tokens to a user, considering the total supply.
- **allowance:** This method is exactly the same as the approved method except that it checks if one user has enough balance to send a certain amount of tokens to another.

ERC-20 - 2 Events

- **Transfer:** Triggered when tokens are transferred between accounts.
- **Approval:** Triggered when a token owner approves another account to spend tokens on their behalf.

ERC-20 - 3 Optional Functions

- **name**: A method that returns the name of the token.
- **symbol**: A method that returns the symbol of the token.
- **decimals**: A method that returns the number of decimals the token uses. It is used to define the smallest unit of the token. For example, if an ERC-20 token has a decimals value of 6, this means that the token can be divided up to six decimal places.

ERC-20 - 3 cryptocurrency supply

- **Fixed Supply**: The total supply of the token is issued on deployment and sent to the deploying wallet address..

- **Uncapped Lazy Supply**: Tokens aren't minted and issued in a unique batch on deployment, **but in small quantities and sent to the specified wallet(s)**, consequently to one or more actions.

- **Capped Lazy Supply**: Like in the **Uncapped Lazy Supply** mechanism, the **tokens are issued in small quantities**, with the only difference that here the max-supply is decided beforehand and hardcoded in the smart contract, or passed on deployment.

ERC-20 – Fixed supply

- Let’s say we want a token with a fixed supply of 1000 SaluTokens initially allocated to the wallet that deploys the contract, to do so we simply called the _mint() private function inside the Token Constructor:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract SaluToken is ERC20 {
    constructor() ERC20("SaluToken", "ST") {
        _mint(msg.sender, 1000 * (10 ** 18));
    }
}
```

Knowing that the constructor gets called only once, on deployment, this smart contract not only transferred 1000 tokens (the amount has 18 decimal places) to our wallet but made it also impossible to mint any new token, not having an exposed _mint() function in its code.

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol

ERC-20 – Uncapped Lazy supply

- To achieve this we can simply move the mint function from our constructor to a new public function that can be called under different circumstances:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract SaluToken is ERC20 {
    constructor() ERC20("SaluToken", "ST") {}

    function issueToken() public {
        _mint(msg.sender, 1000 * (10 ** 18));
    }
}
```

```
function issueToken(address receiver, uint256 amount) public {
        _mint(receiver, amount);
}
```

The issue here is that, without any further control, there's no limit to the number of tokens a user can issue. As a workaround, we can cap the total supply of the Token.

ERC-20 – Capped Lazy supply

- To add a **max supply** to our Token, we can use the ERC20Capped OpenZeppelin library.
- ERC20Capped is a sub-class of ERC20Mintable that in turn inherits from the standard ERC20 contract. This allows us to substitute the ERC20.sol library in our code, with ERC20Capped, while maintaining methods such as _mint() and balanceOf().

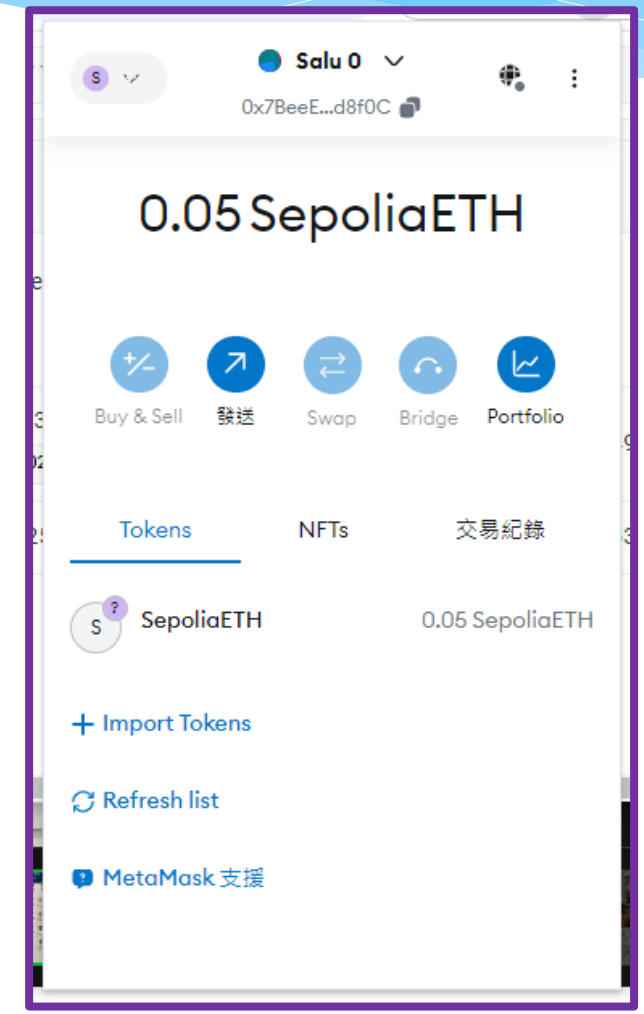
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Capped.sol";

contract SaluToken is ERC20Capped {
    constructor(uint256 cap) ERC20("SaluToken", "ST") ERC20Capped(cap) {}

    function issueToken() public {
        _mint(msg.sender, 1000 * (10 ** 18));
    }
}
```

We'll also need to specify a **cap value** by passing a uint256 argument to our Token constructor and feeding it into the **ERC20Capped constructor**.

https://docs.openzeppelin.com/contracts/4.x/api/token/erc20#ERC20Capped

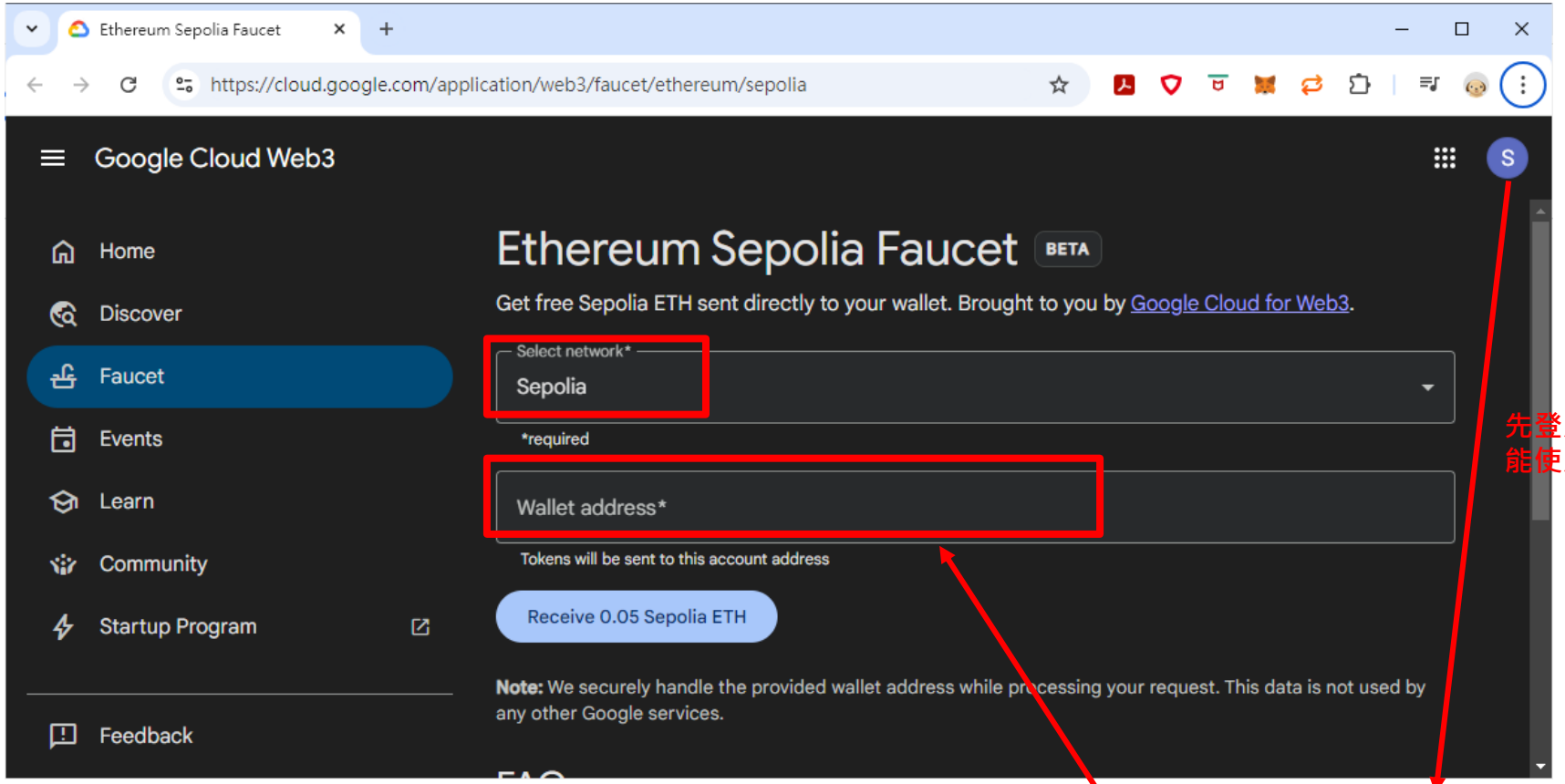
Install the Metamask extension

- 請參考Metamask安裝文件，並安裝。



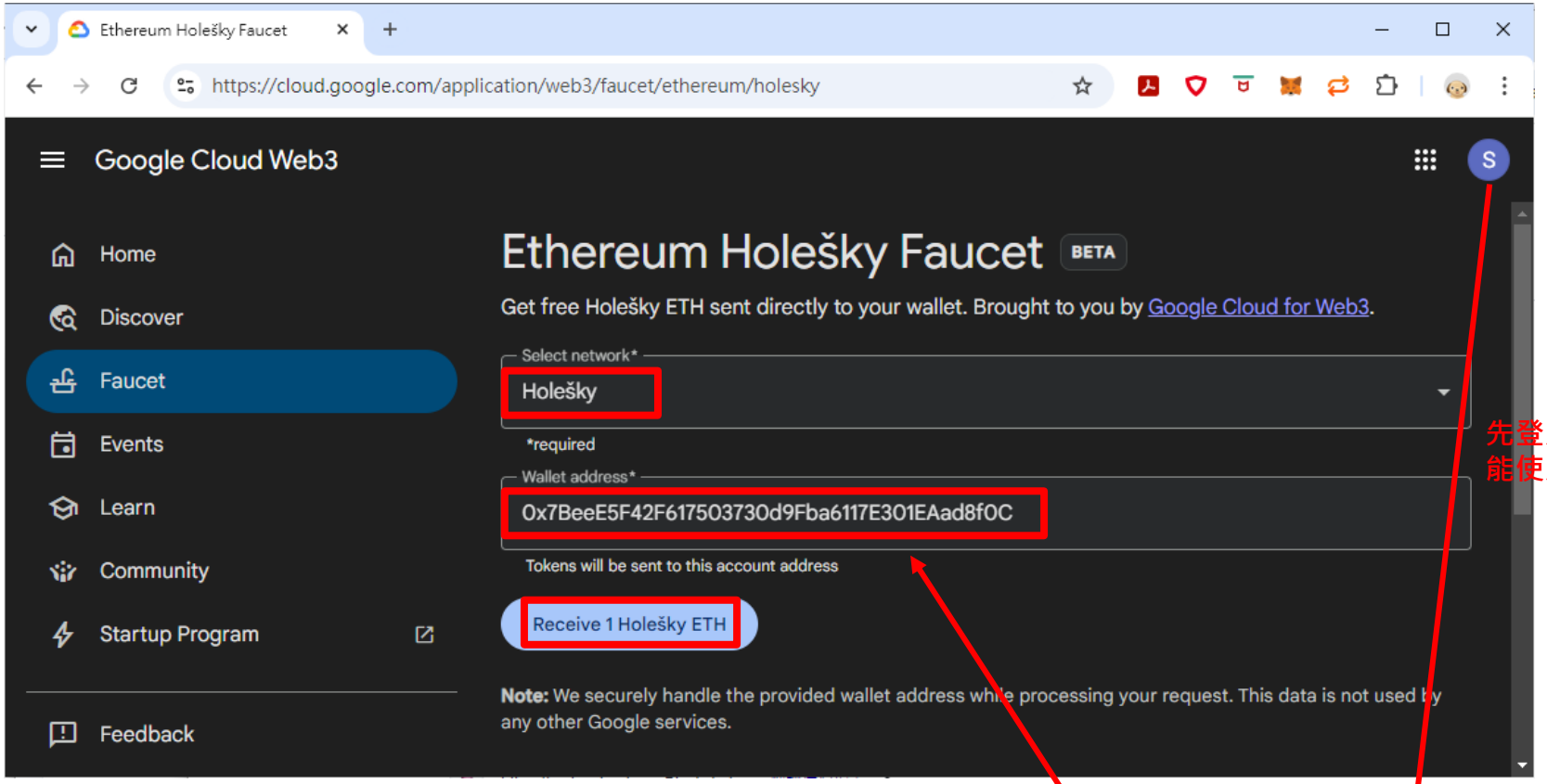
Sepolia - Faucets

- https://cloud.google.com/application/web3/faucet/ethereum/sepolia



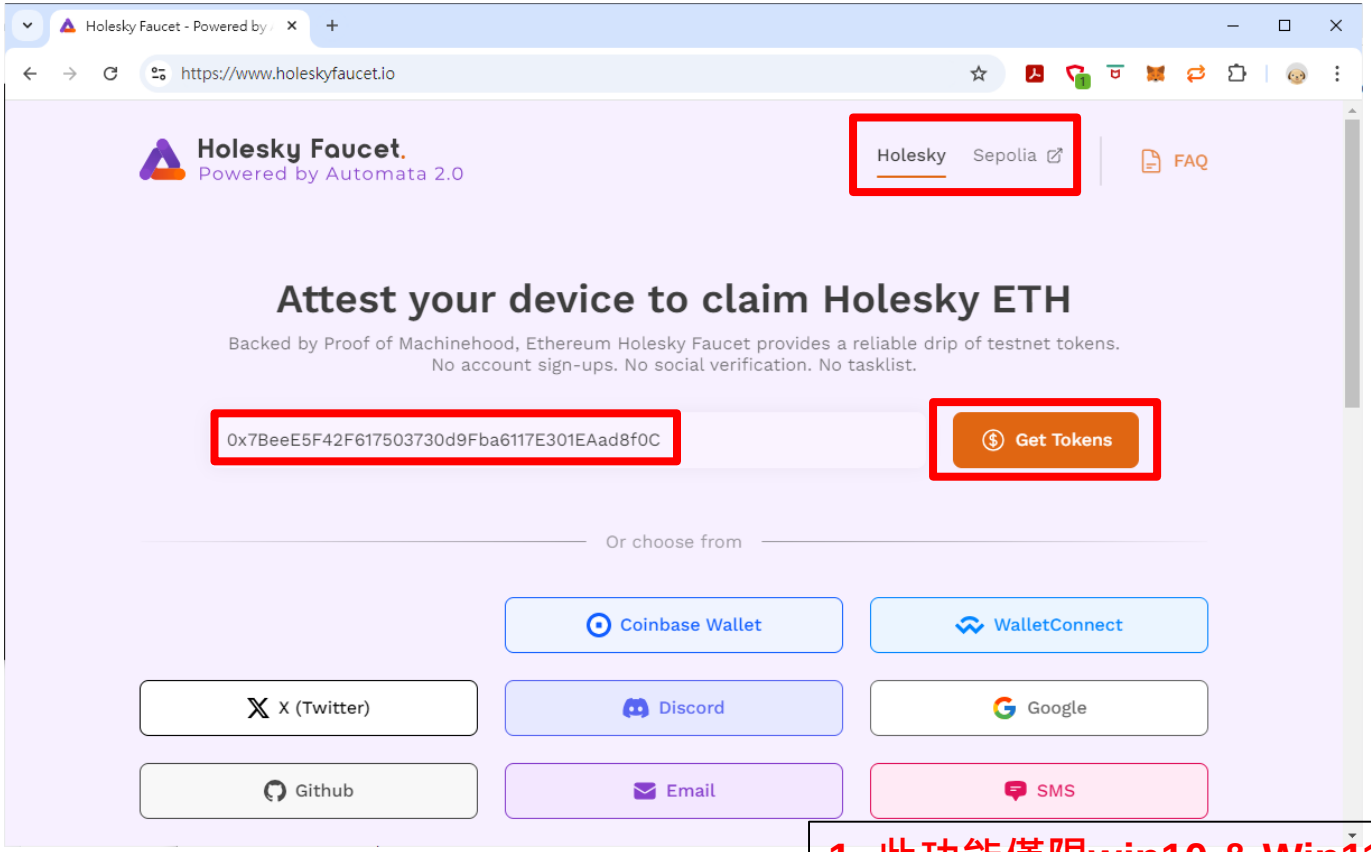
Holesky - Faucets

- https://cloud.google.com/application/web3/faucet/ethereum/holesky



Holesky & Sepolia - Faucets

- https://www.holeskyfaucet.io/



1. 此功能僅限win10 & Win11
2. 每個對外ip一天只能給一個Wallet address使用

Create an ERC-20 Token

Create an ERC-20 Token

- For ease and security, we'll use the OpenZeppelin ERC-20 contract to create our token. With OpenZeppelin, we don't need to write the whole ERC-20 interface. Instead, we can import the library contract and use its functions.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract SaluToken is ERC20 {
    constructor() ERC20("SaluToken", "ST") {
        _mint(msg.sender, 100000000 * (10 ** uint256(decimals())));
    }
}
```

SaluToken.sol

=100,000,000 * (10^{18})

18

See Full ERC-20 code

Means the total supply will be 100,000,000 × 10^{18} tokens. We use 10^n because **Solidity doesn´t (fully) support decimal numbers**, only integers. In wallets, dApps, etc. this number is converted back to the actual balance with decimals.
To specify 2.5 tokens, for example, you would use 2.5 × 10^{18} (= 2,500,000,000,000,000,000) instead. You can reverse it by multiplying with 10^{-18} instead.

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol

Create an ERC-20 Token

- The **SPDX-License-Identifier** comment specifies the license under which the contract is released.
- The **pragma** directive states the compiler version to use.
- The **ERC20** contract from OpenZeppelin is imported and used as a base.
- **SaluToken** is the name of your contract, and it extends the ERC20 contract.
- The **constructor** function initializes your token with a name ("**SaluToken**") and a symbol ("**ST**").
- The **_mint** function in the constructor mints an initial supply of tokens. In this example, **100000000** tokens are minted and assigned to the address that deploys the contract. The number of tokens is adjusted by the decimals value, which defaults to **18** in the OpenZeppelin implementation.

Create an ERC-20 Token

- 假設在ERC-20中，我們宣告一變數_totalSupply為uint256.

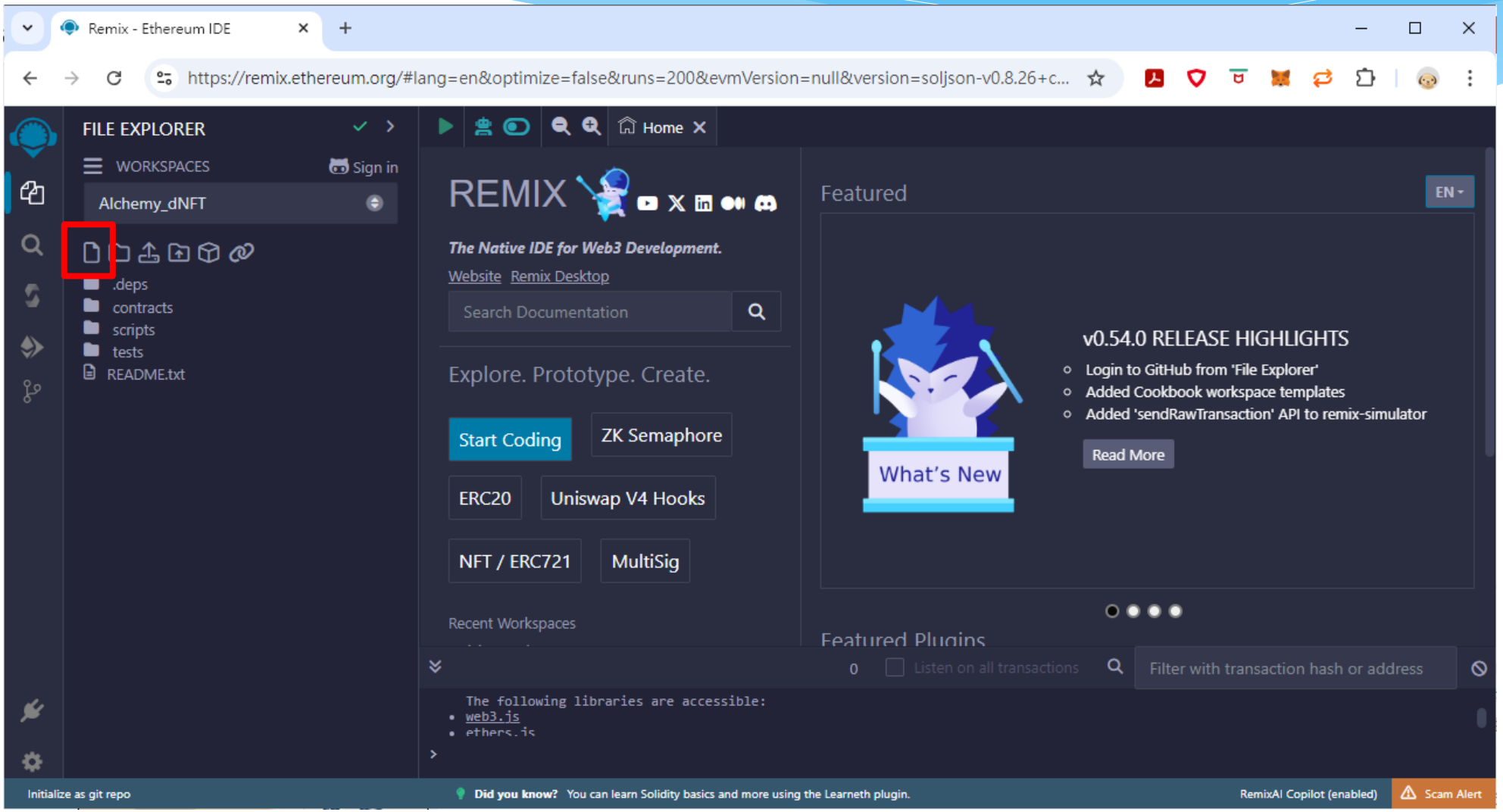
```
uint256 private _totalSupply;
```

- 理論上最大的token數量可以達到 2^{256}-1. 超過這個數字Solidity 0.8會阻止並產生overflow。而且只有在解釋token這個數字時，小數位數才重要。
- 我們曉得 2^{256} 將有 log(2^{256}) + 1 位數，即有 78位數。這意味著如果小數點後為 0，那麼最大的數字是以下數字：

```
                1           2           3           4           5           6           7
123,456,789,012,345,678,901,234,567,890,123,456,789,012,345,678,901,234,567,890,123,456,789,012,345,678

115,792,089,237,316,195,423,570,985,008,687,907,853,269,984,665,640,564,039,457,584,007,913,129,639,936 - 1
```

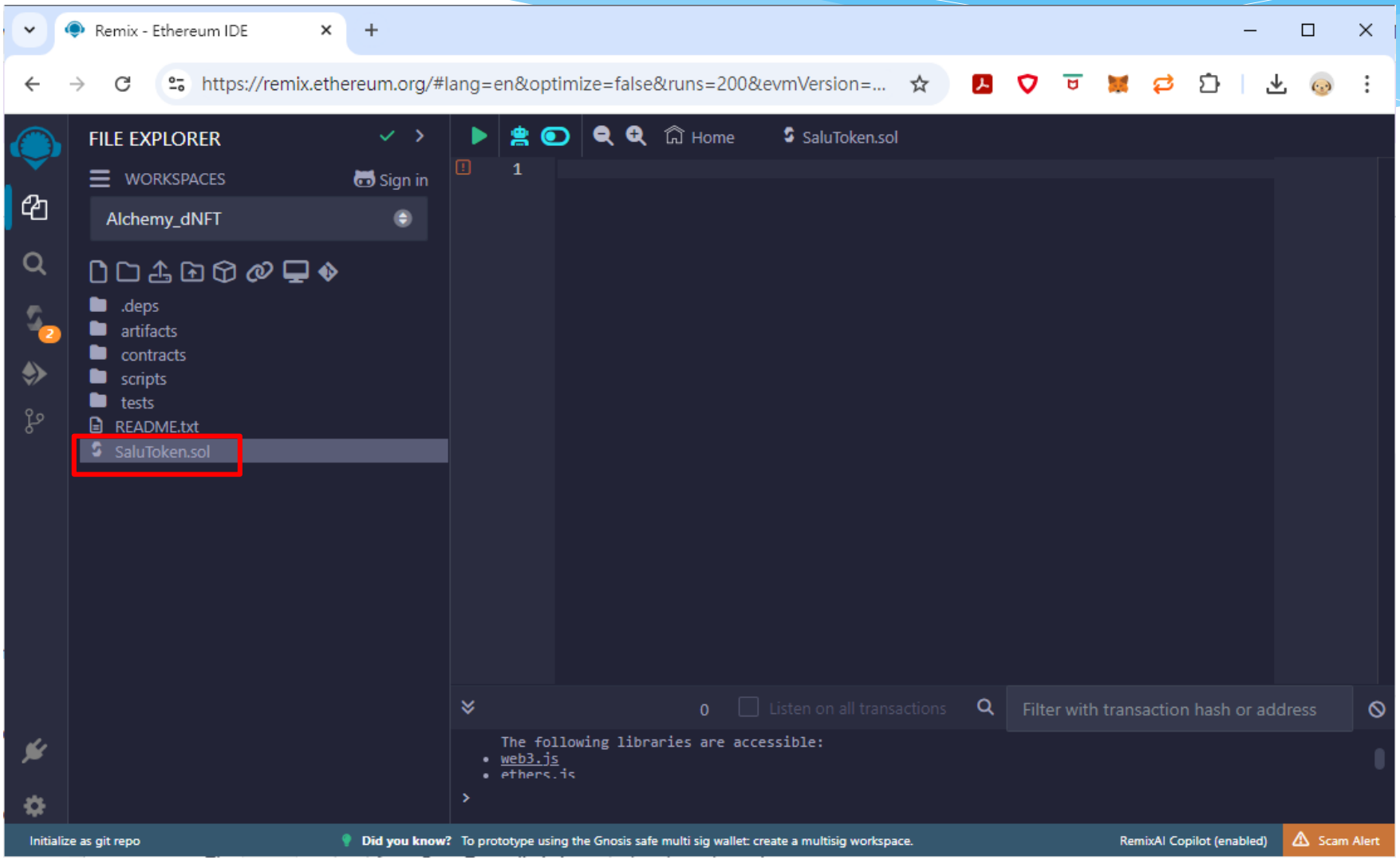
Case 1

Create an ERC-20 Token

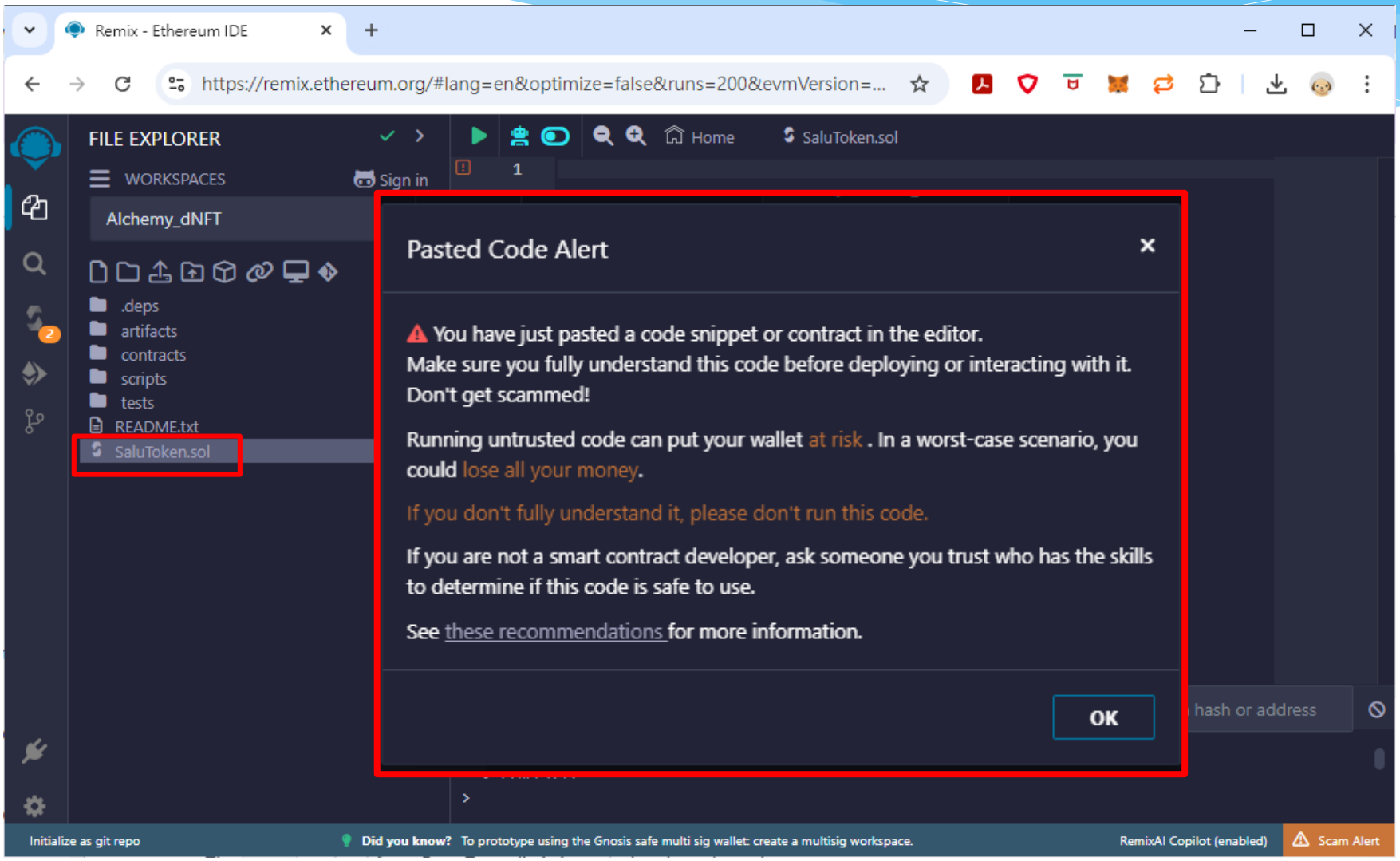


https://remix.ethereum.org

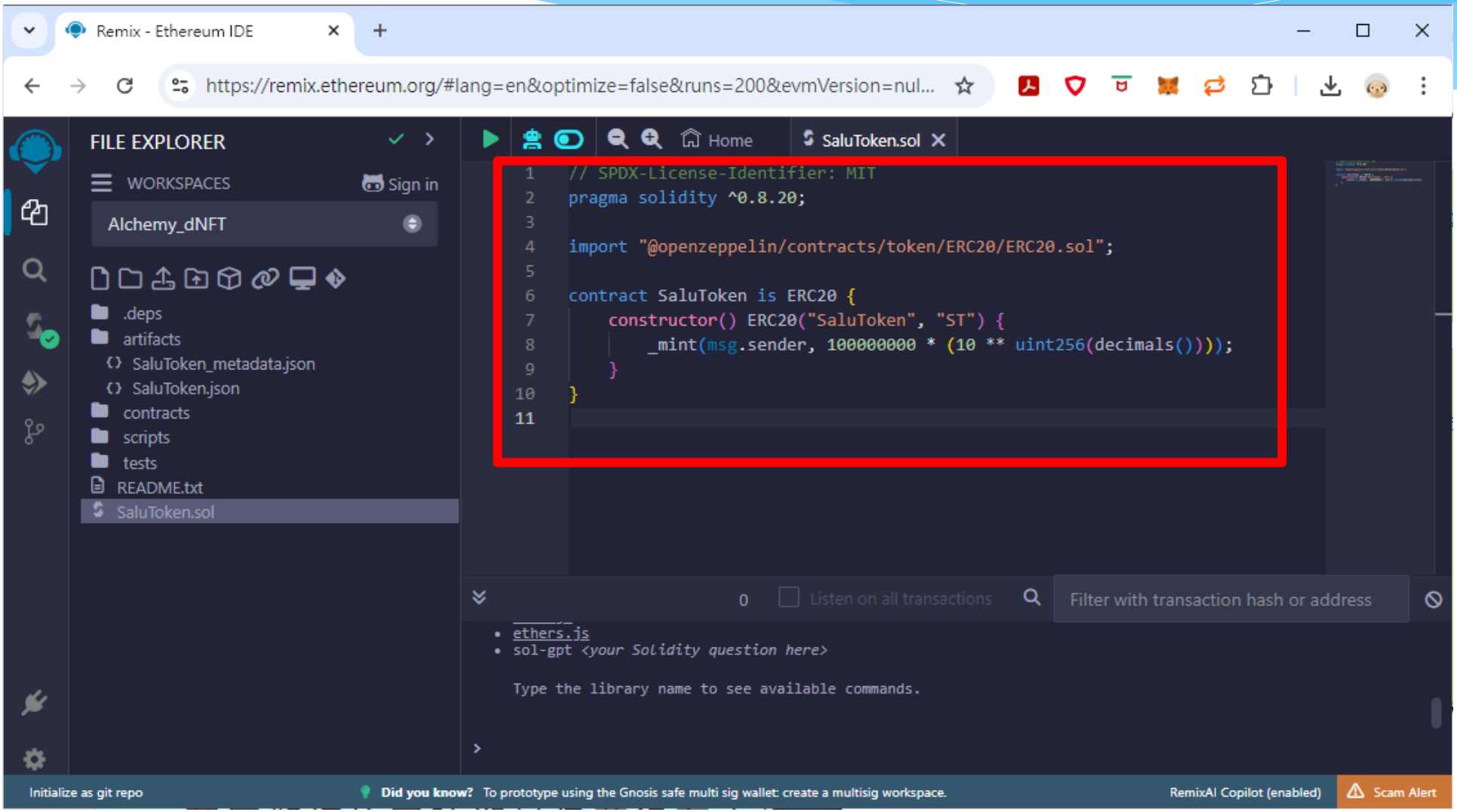
SaluToken.sol



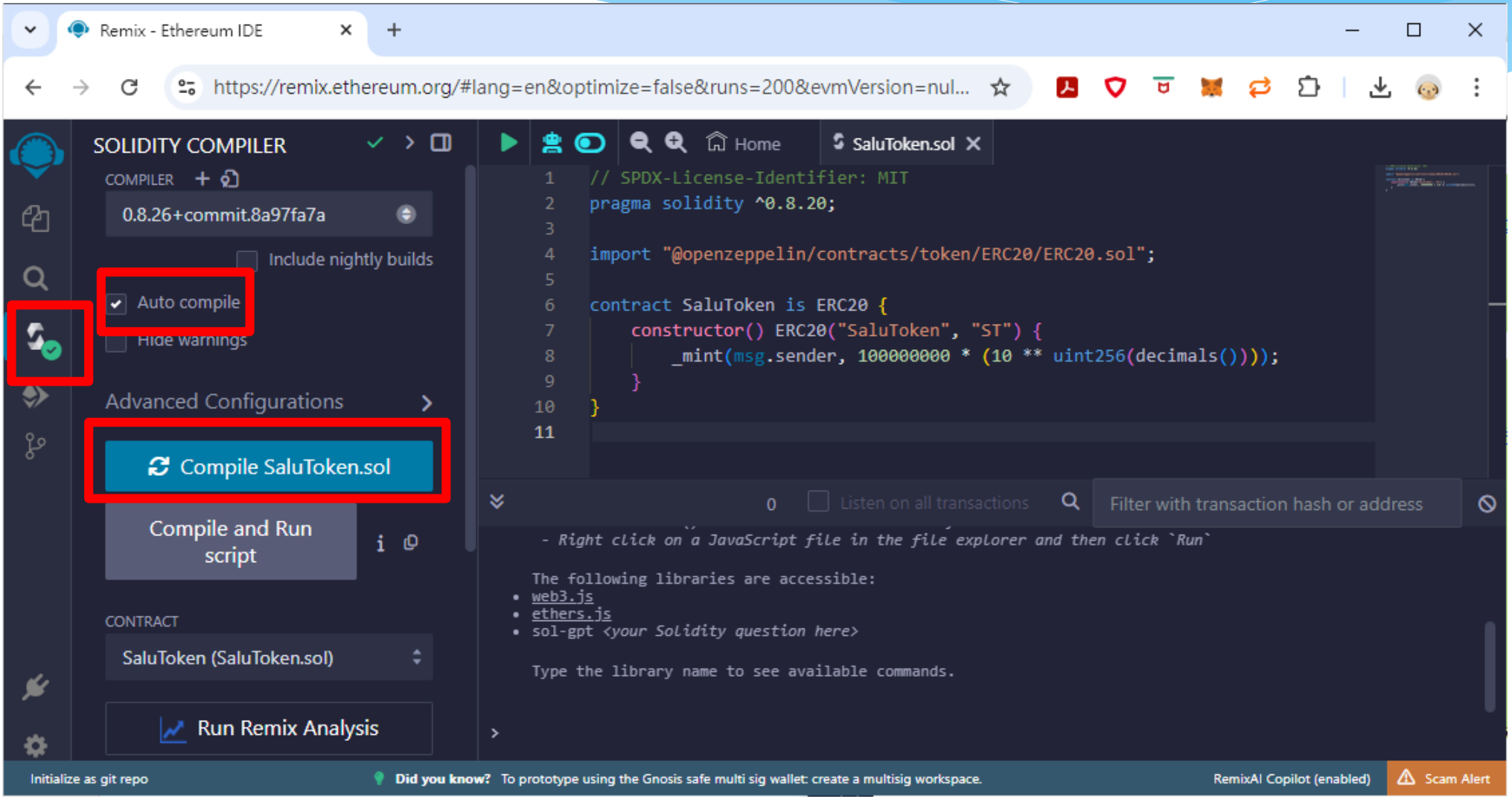
SaluToken.sol



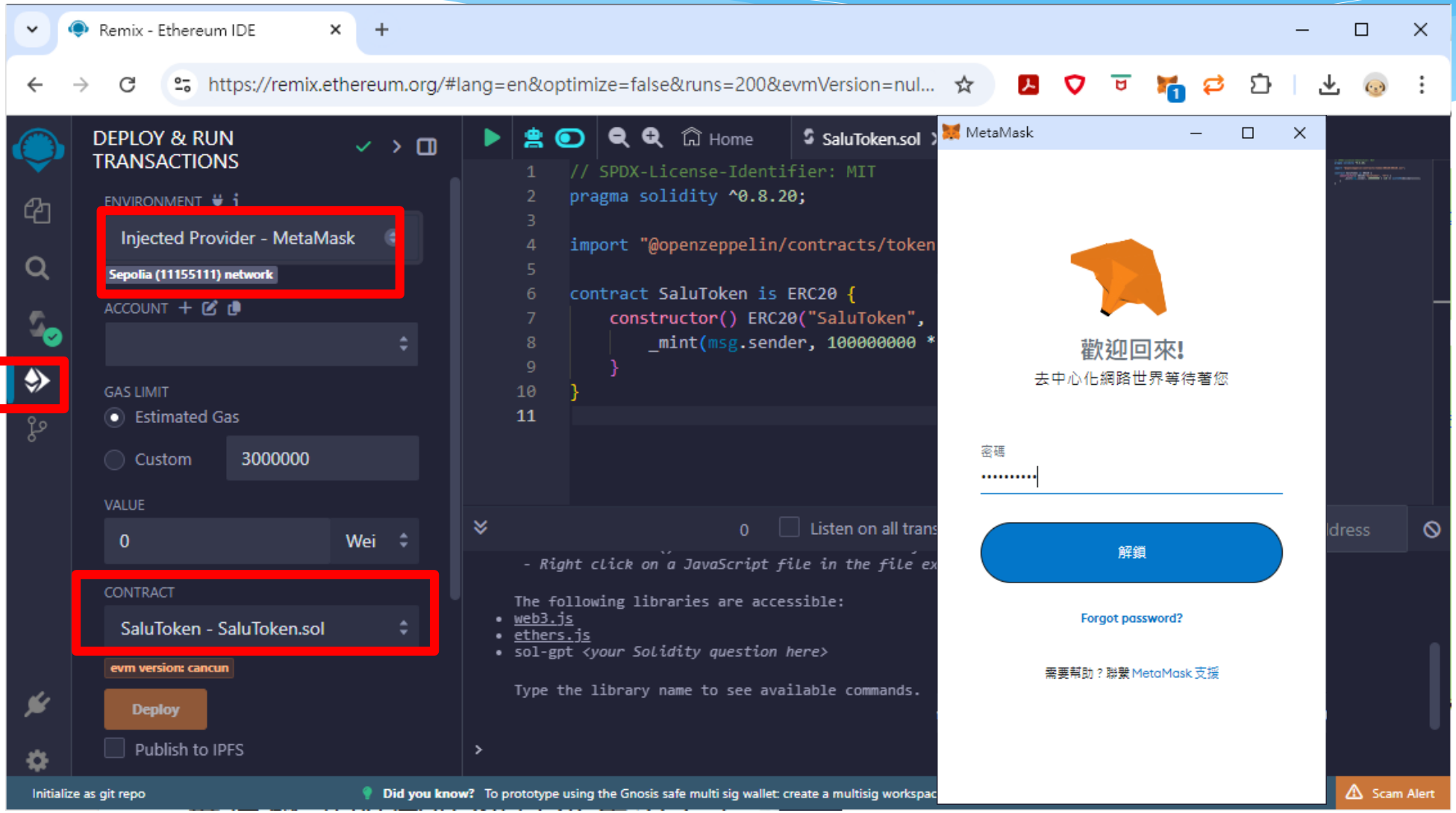
SaluToken.sol



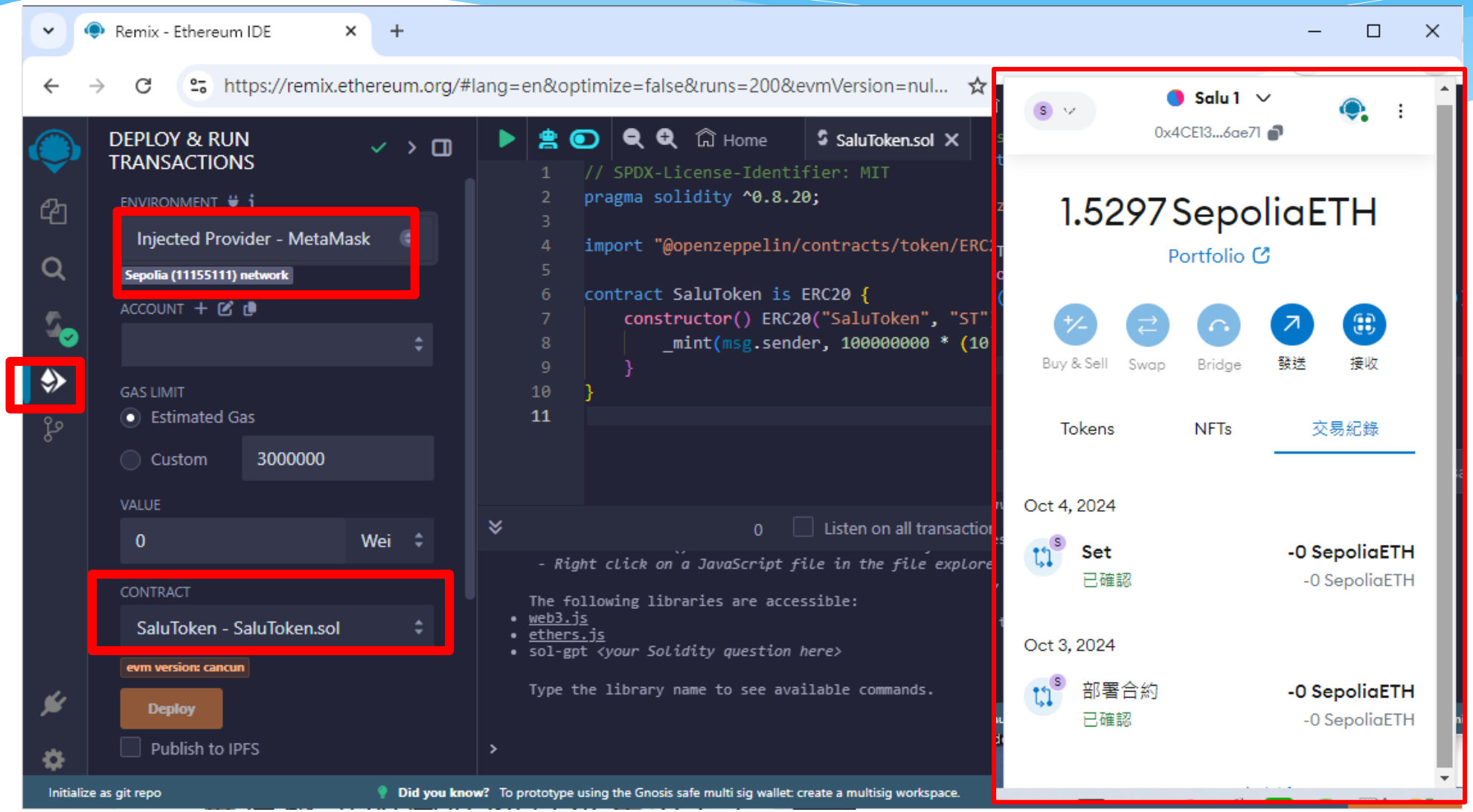
Compile Smart Contract



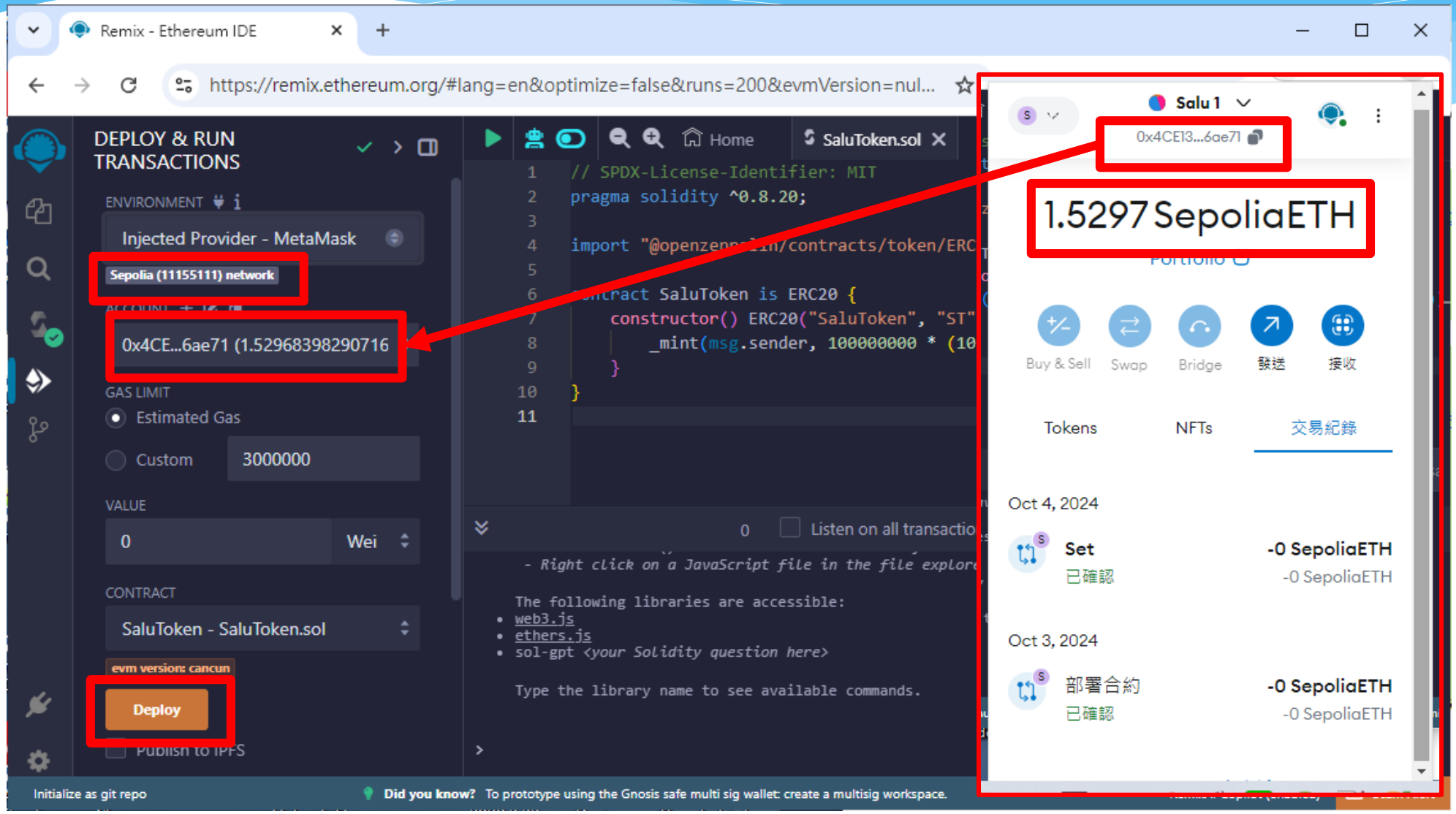
Deploy an ERC-20 Token



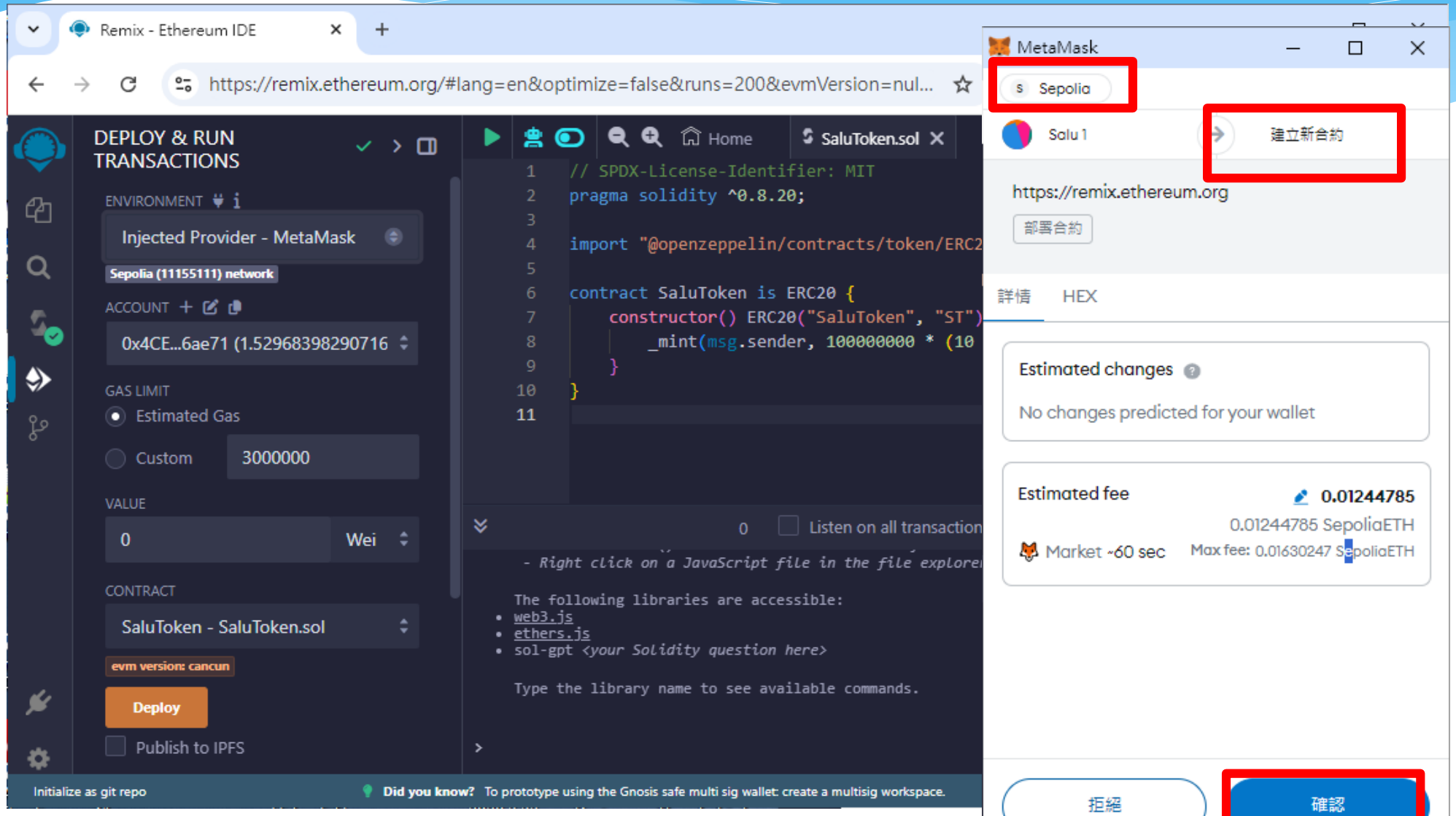
Deploy and Run Transaction



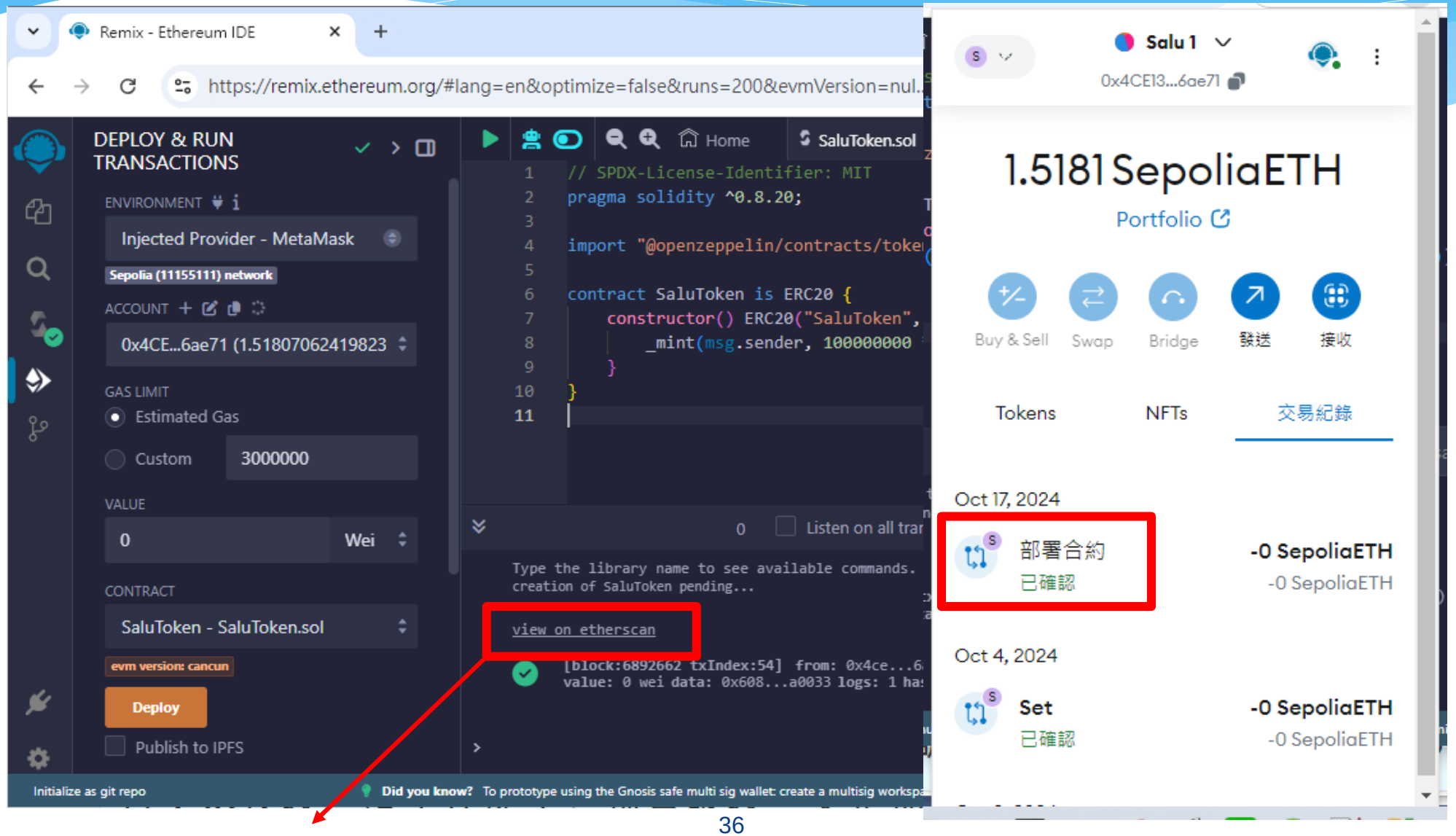
Deploy and Run Transaction



Deploy and Run Transaction

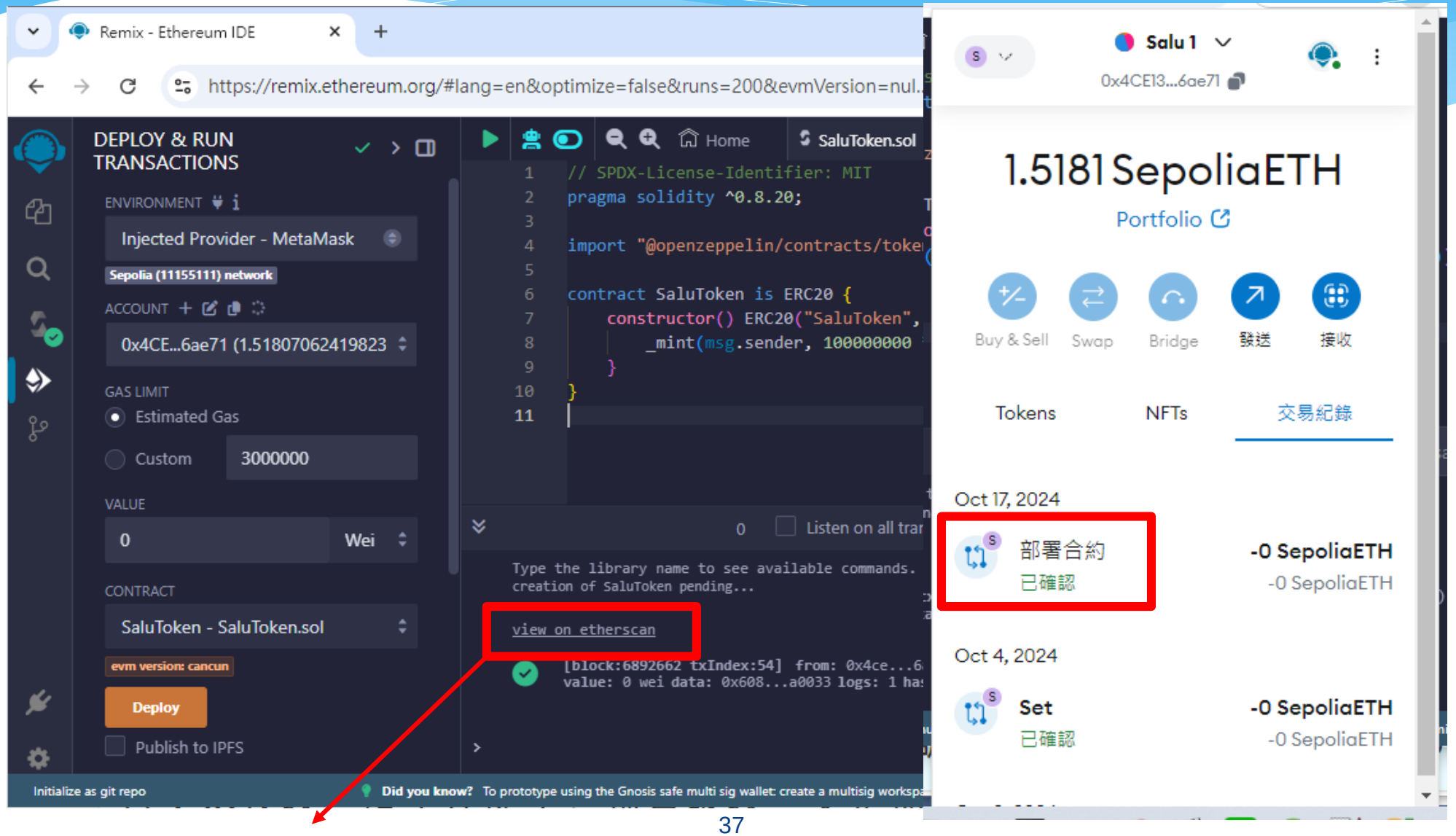


Deploy and Run Transaction



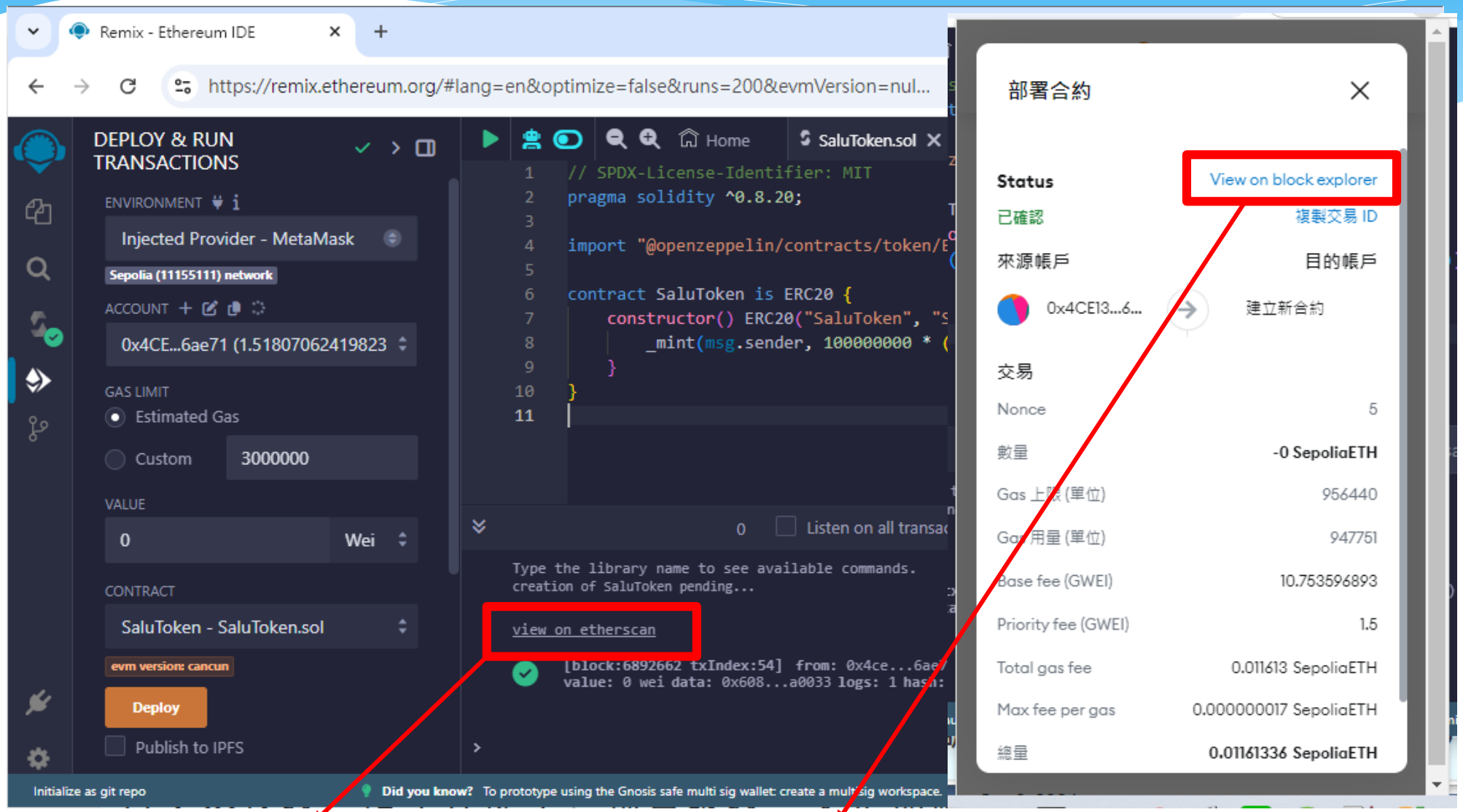
https://sepolia.etherscan.io/tx/0x4b37bb9f122192ce09e19f53e022960f5b4f4c22e65fbe8f6dee114ca09dd9ec

Deploy and Run Transaction



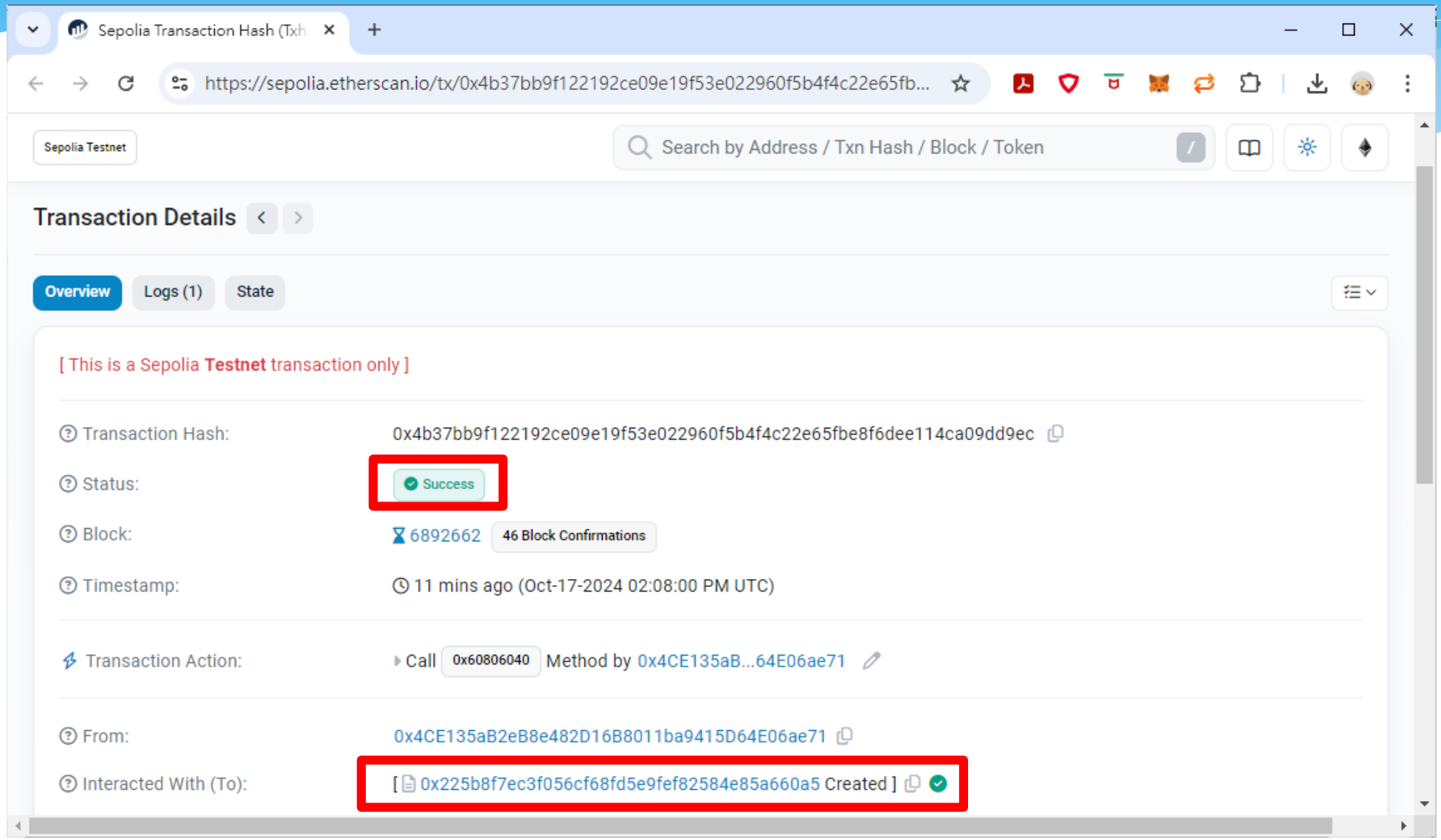
https://sepolia.etherscan.io/tx/0x4b37bb9f122192ce09e19f53e022960f5b4f4c22e65fbe8f6dee114ca09dd9ec

Deploy and Run Transaction



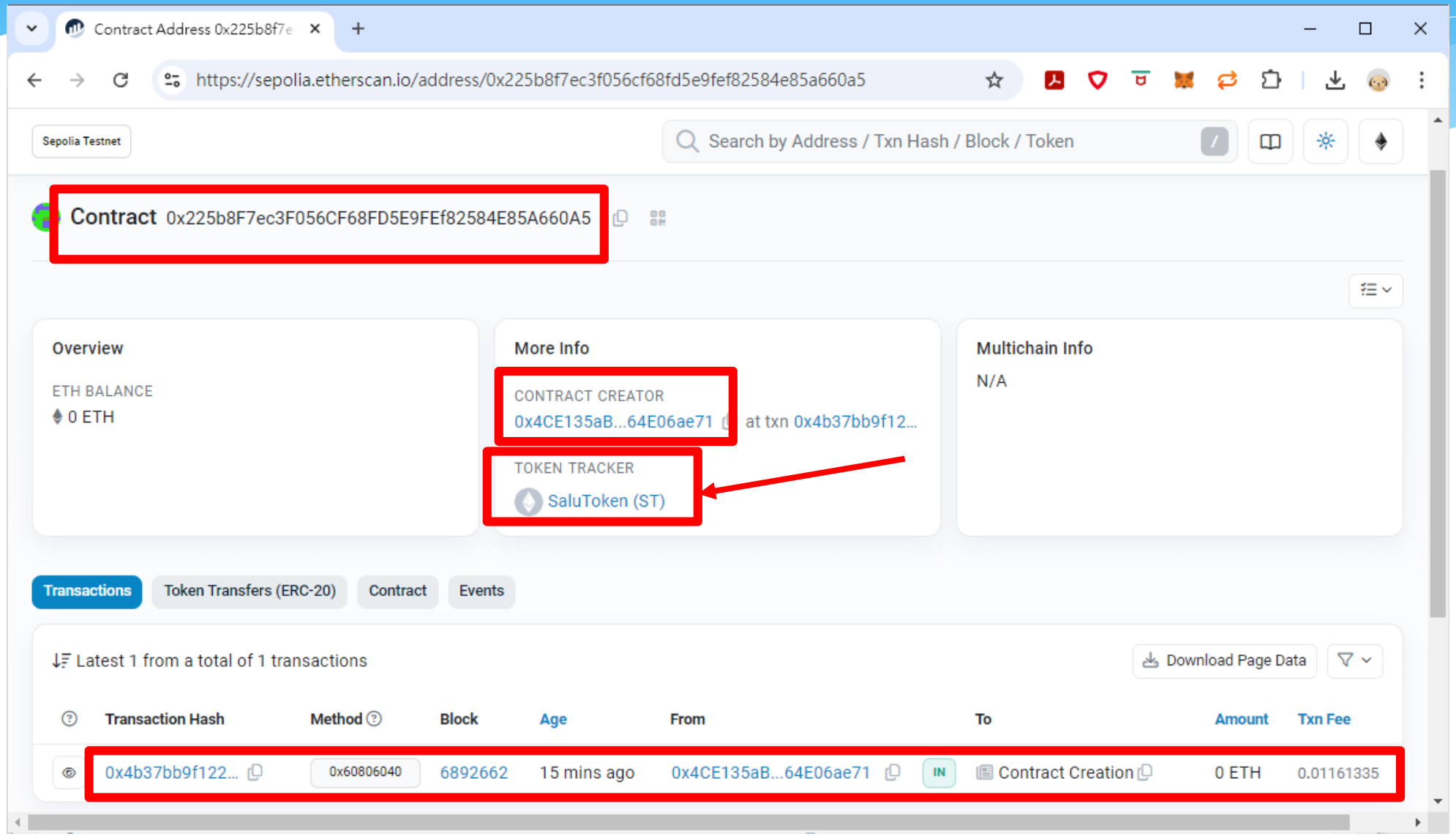
https://sepolia.etherscan.io/tx/0x4b37bb9f122192ce09e19f53e022960f5b4f4c22e65fbe8f6dee114ca09dd9ec

View Transaction on Sepolia



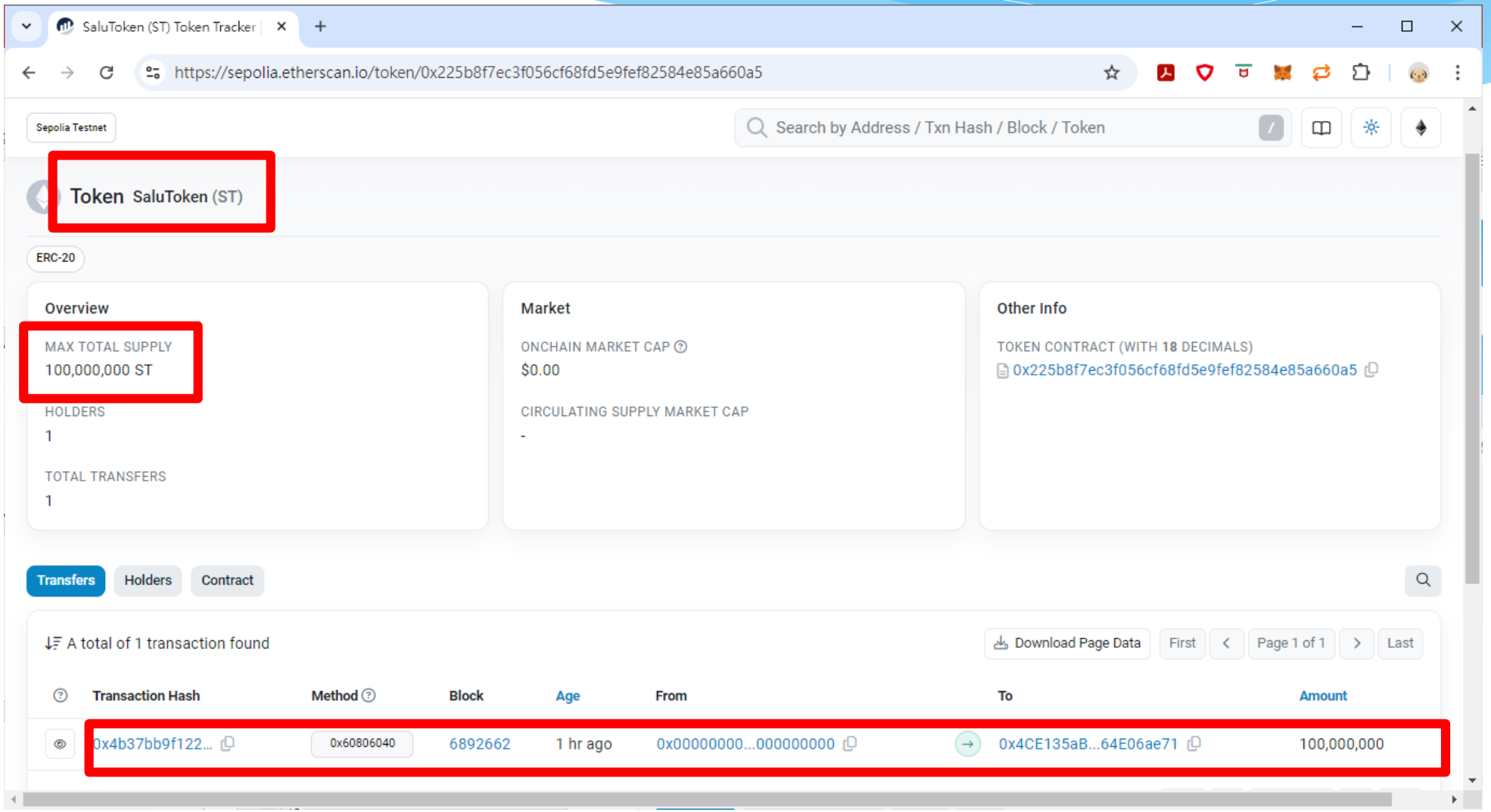
https://sepolia.etherscan.io/tx/0x4b37bb9f122192ce09e19f53e022960f5b4f4c22e65fbe8f6dee114ca09dd9ec

Smart Contract Information



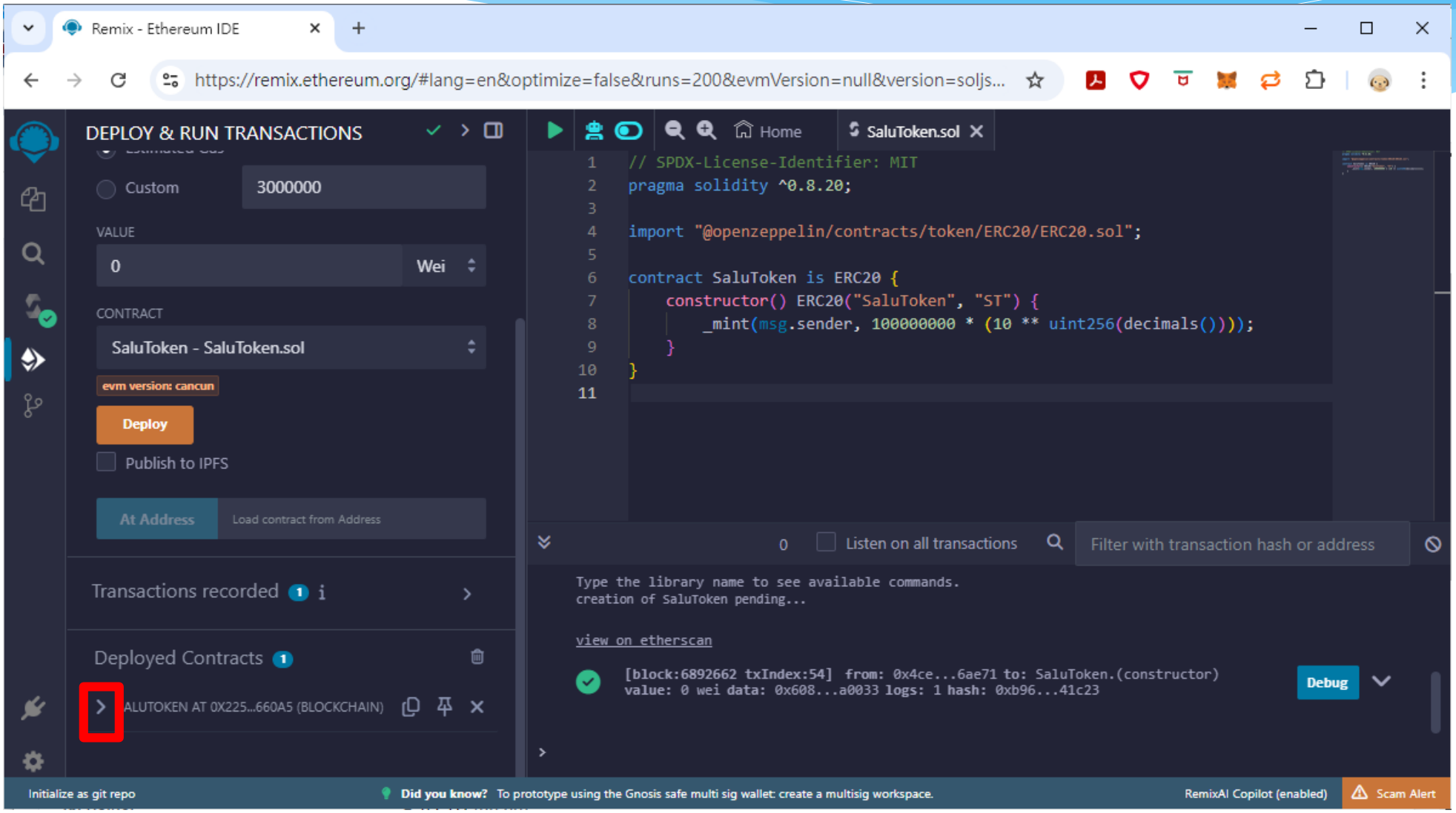
https://sepolia.etherscan.io/address/0x225b8f7ec3f056cf68fd5e9fef82584e85a660a5

Smart Contract Information



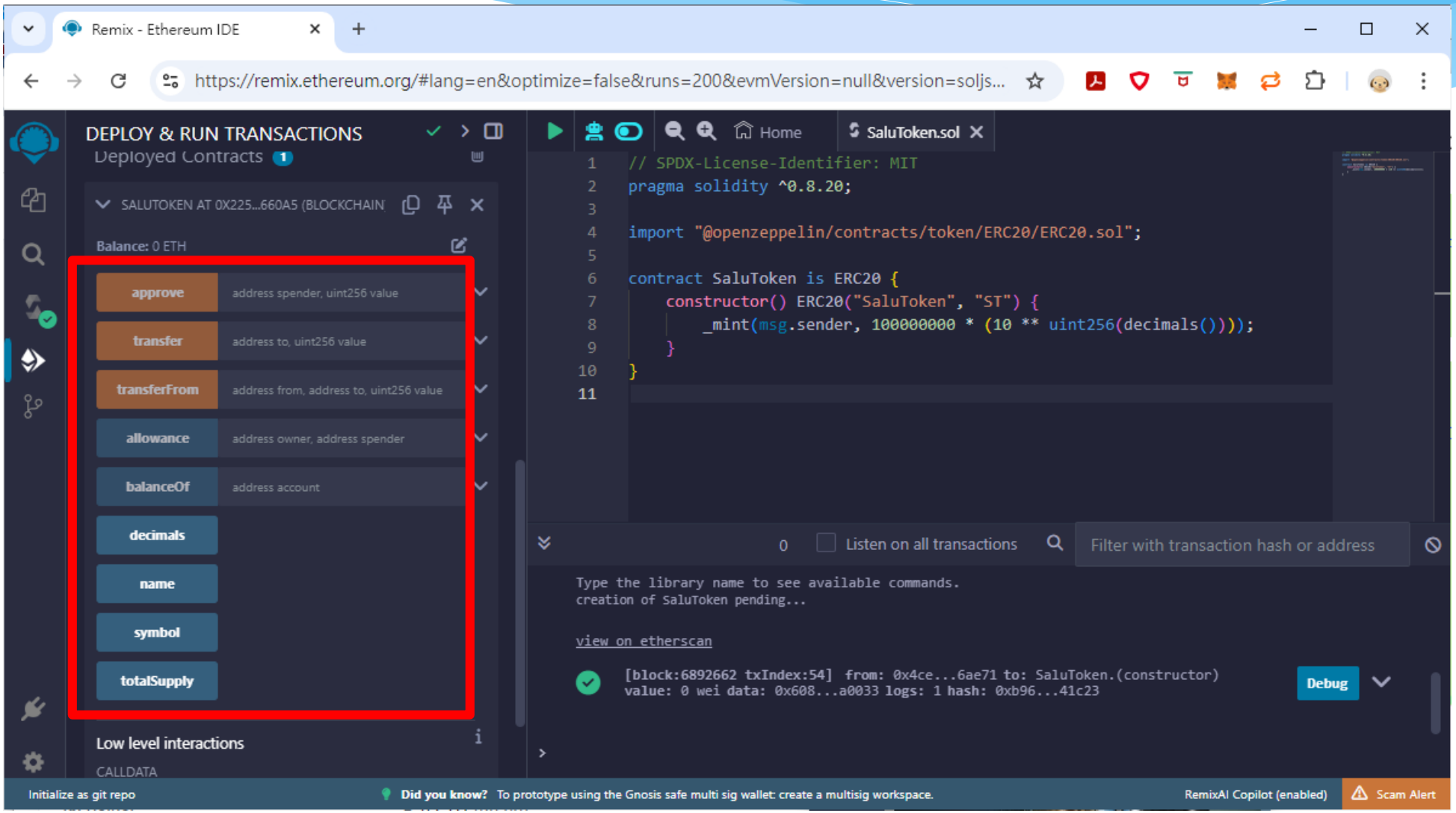
https://sepolia.etherscan.io/address/0x225b8f7ec3f056cf68fd5e9fef82584e85a660a5

Deploy and Run Transaction

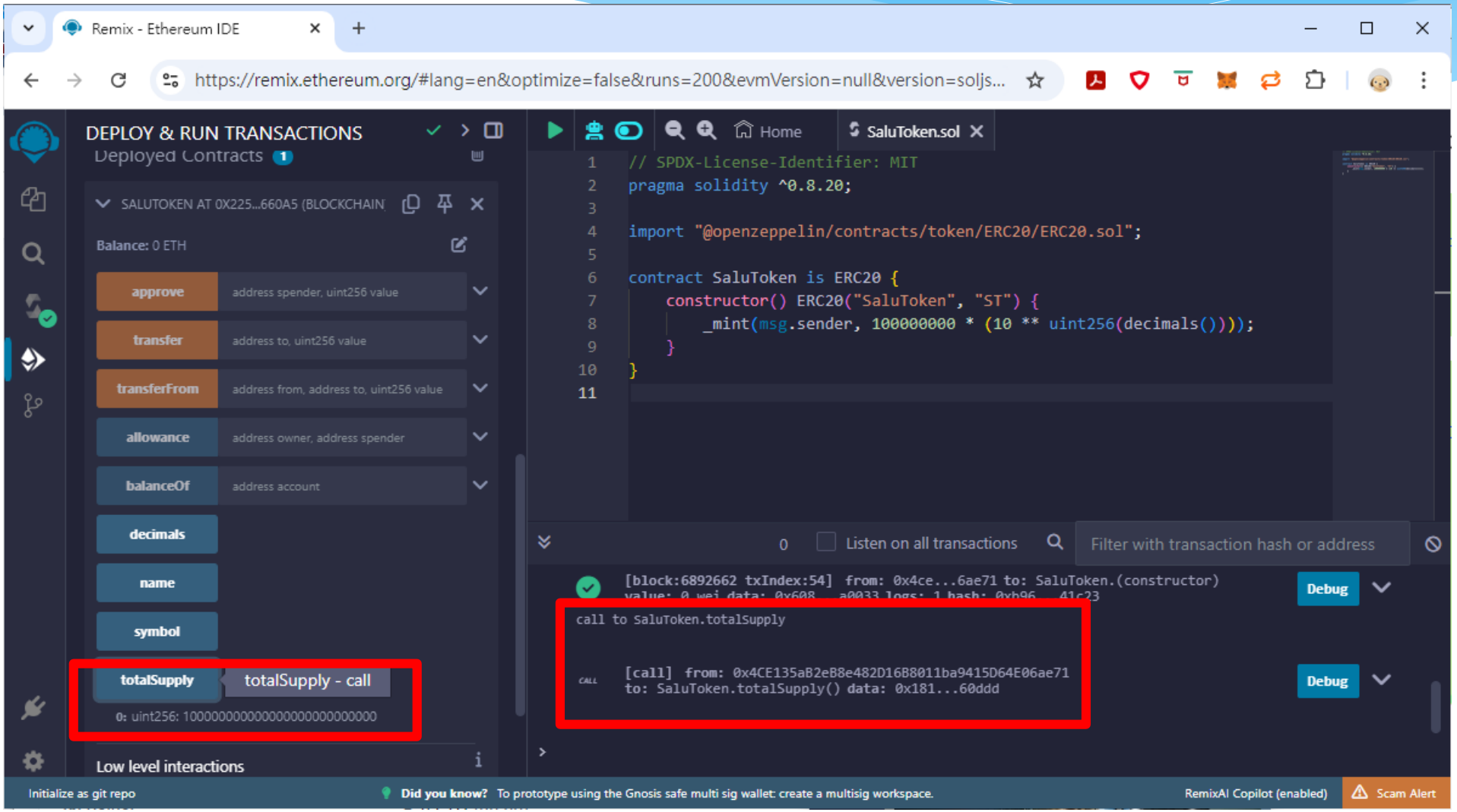


Contract address : 0x225b8F7ec3F056CF68FD5E9FEf82584E85A660A5

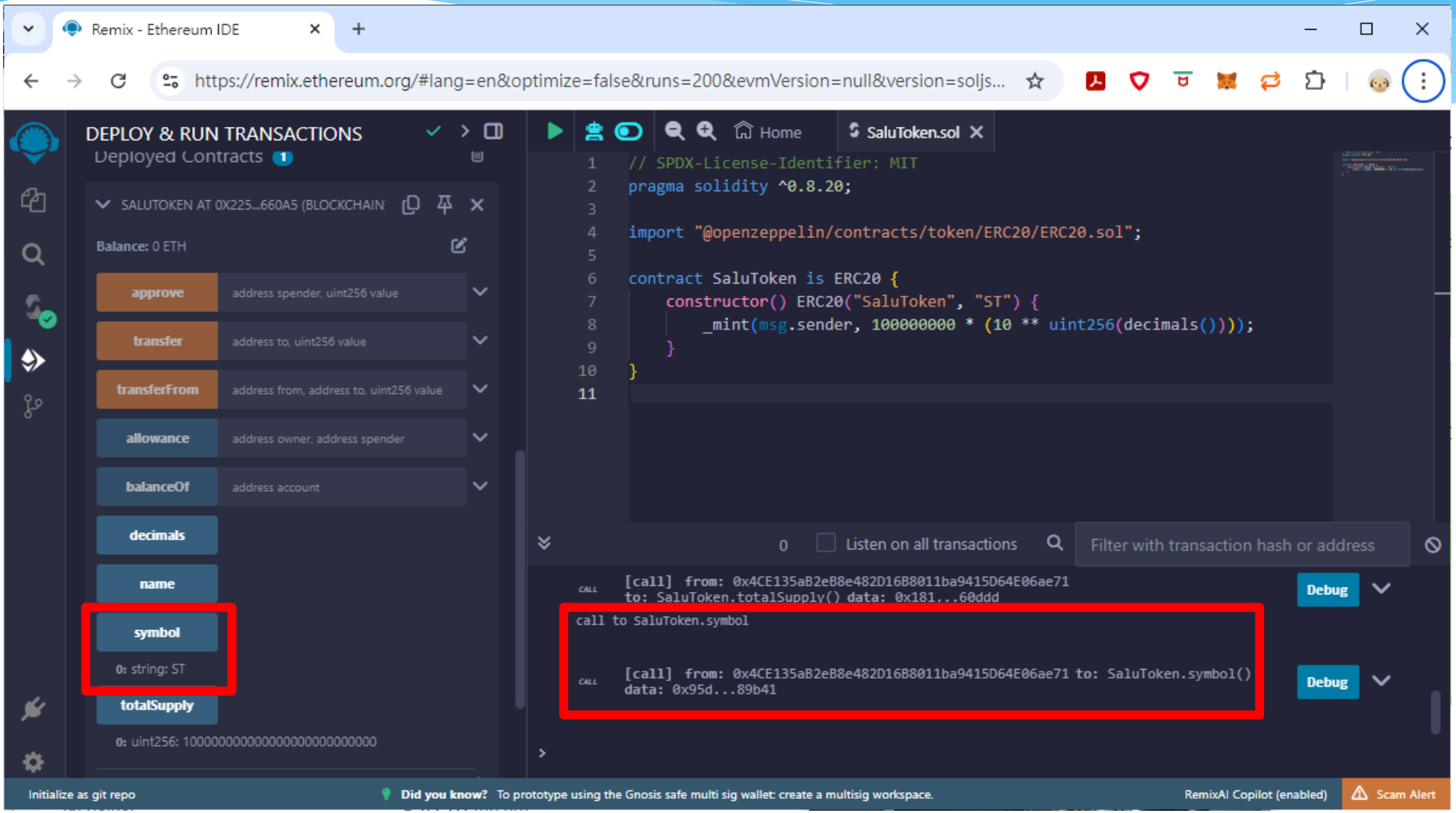
Deploy and Run Transaction



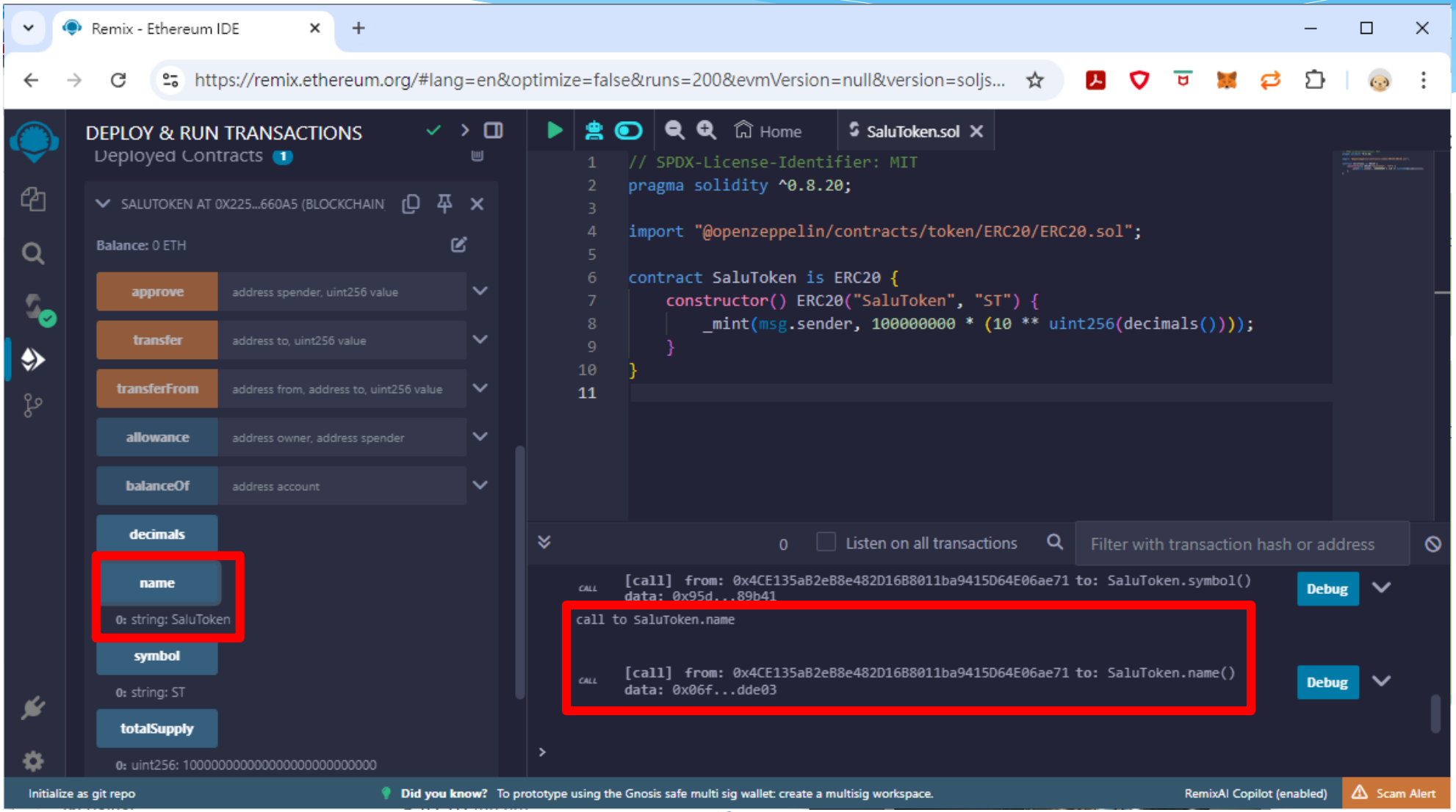
Call totalSupply()



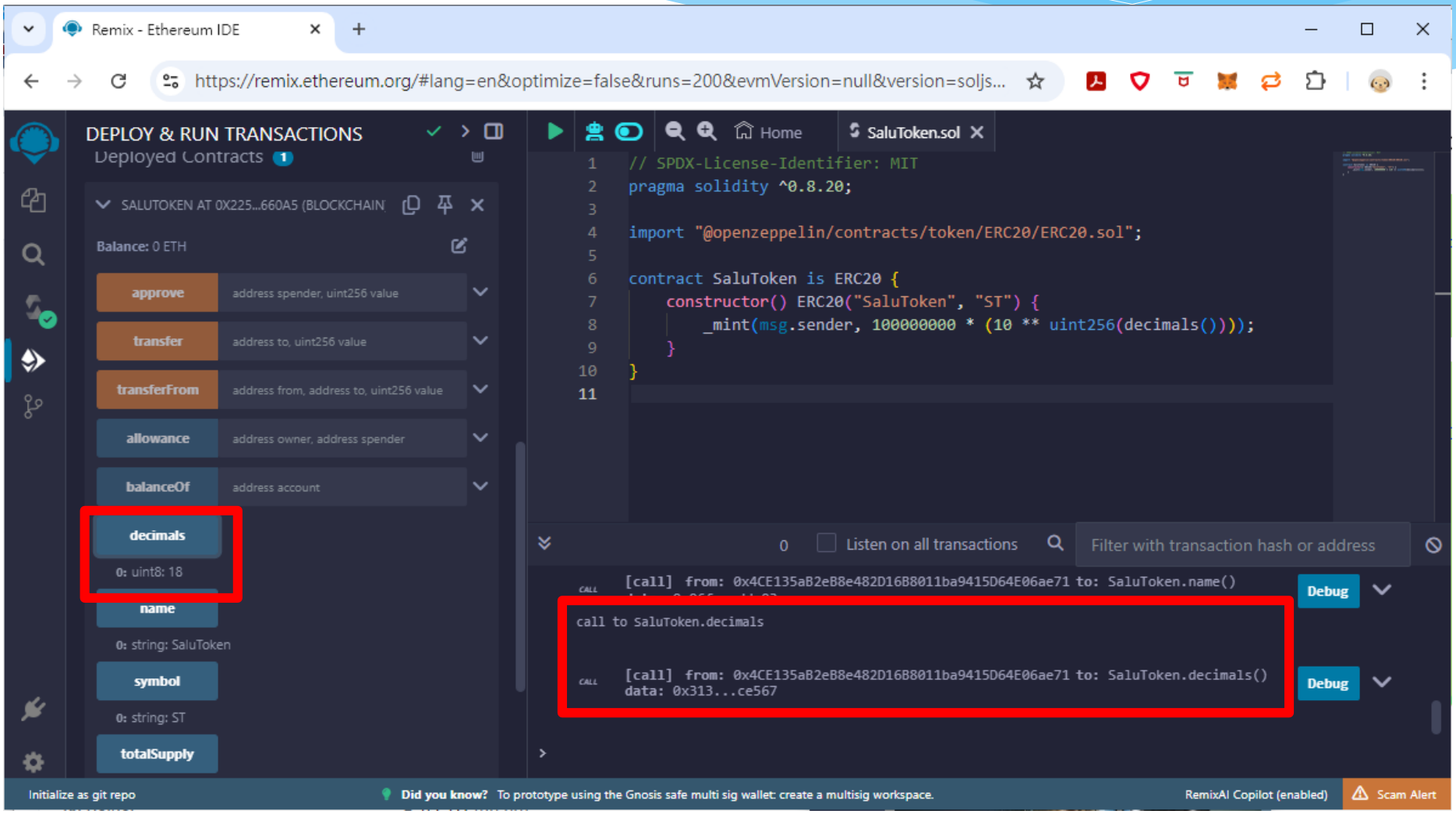
Call symbol()


Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljs...
DEPLOY & RUN TRANSACTIONS
Deployed Contracts 1
SALUTOKEN AT 0X225...660A5 (BLOCKCHAIN
Balance: 0 ETH
approve
address spender, uint256 value
transfer
address to, uint256 value
transferFrom
address from, address to, uint256 value
allowance
address owner, address spender
balanceOf
address account
decimals
name
symbol
0: string: ST
totalSupply
0: uint256: 100000000000000000000000000
Home
SaluToken.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6 contract SaluToken is ERC20 {
7 constructor() ERC20("SaluToken", "ST") {
8 _mint(msg.sender, 100000000 * (10 ** uint256(decimals())));
9 }
10 }
11
0
Listen on all transactions
Filter with transaction hash or address
CALL [call] from: 0x4CE135aB2eB8e482D16B8011ba9415D64E06ae71 to: SaluToken.totalSupply() data: 0x181...60ddd
Debug
call to SaluToken.symbol
CALL [call] from: 0x4CE135aB2eB8e482D16B8011ba9415D64E06ae71 to: SaluToken.symbol() data: 0x95d...89b41
Debug
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

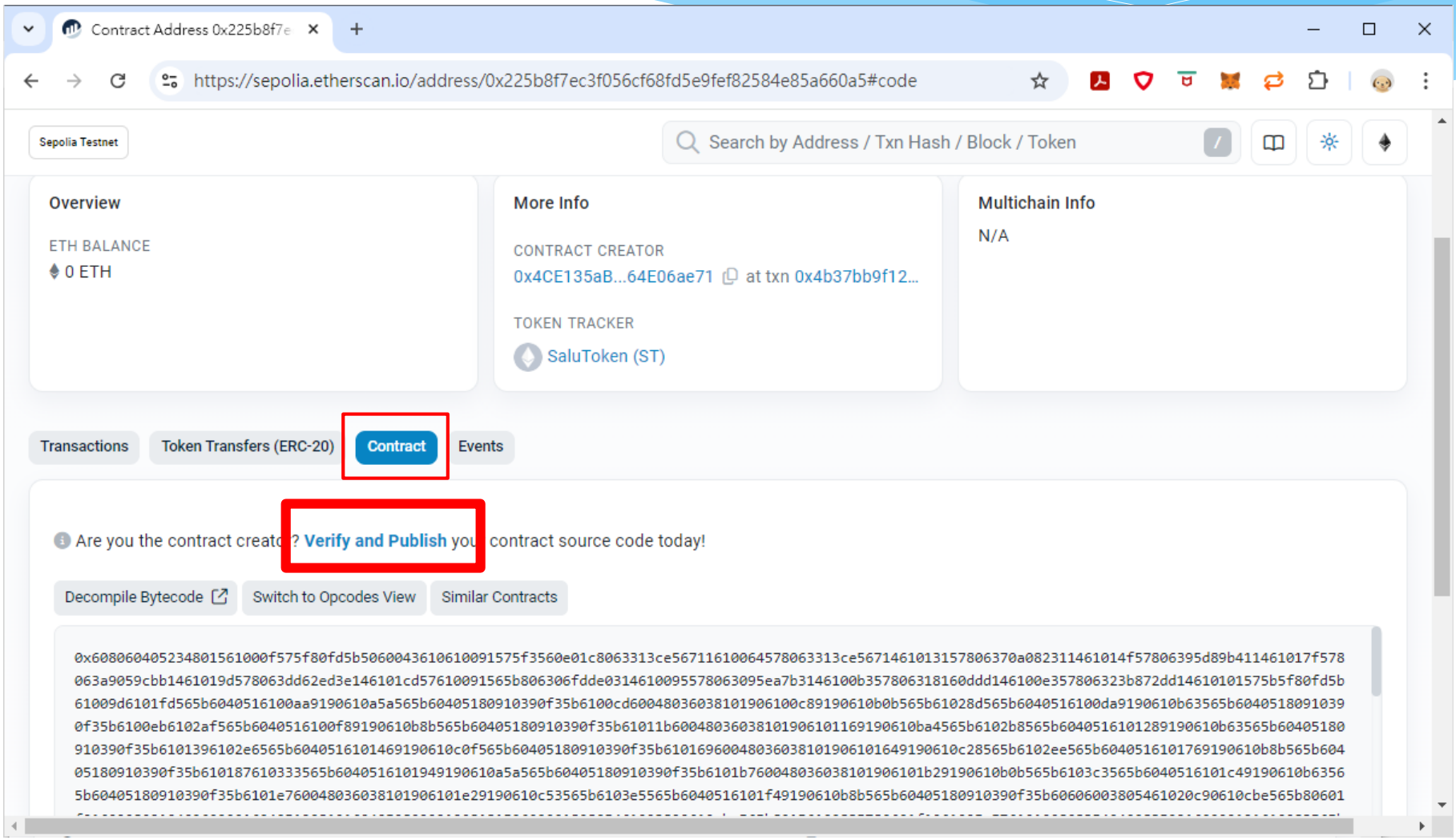
Call name()


Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljs...
DEPLOY & RUN TRANSACTIONS
Deployed Contracts 1
SALUTOKEN AT 0X225...660A5 (BLOCKCHAIN
Balance: 0 ETH
approve
address spender, uint256 value
transfer
address to, uint256 value
transferFrom
address from, address to, uint256 value
allowance
address owner, address spender
balanceOf
address account
decimals
name
0: string: SaluToken
symbol
0: string: ST
totalSupply
0: uint256: 100000000000000000000000000
Home
SaluToken.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6 contract SaluToken is ERC20 {
7 constructor() ERC20("SaluToken", "ST") {
8 _mint(msg.sender, 100000000 * (10 ** uint256(decimals())));
9 }
10 }
11
0
Listen on all transactions
Filter with transaction hash or address
[call] from: 0x4CE135aB2eB8e482D16B8011ba9415D64E06ae71 to: SaluToken.symbol()
data: 0x95d...89b41
Debug
call to SaluToken.name
[call] from: 0x4CE135aB2eB8e482D16B8011ba9415D64E06ae71 to: SaluToken.name()
data: 0x06f...dde03
Debug
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

Call decimals()

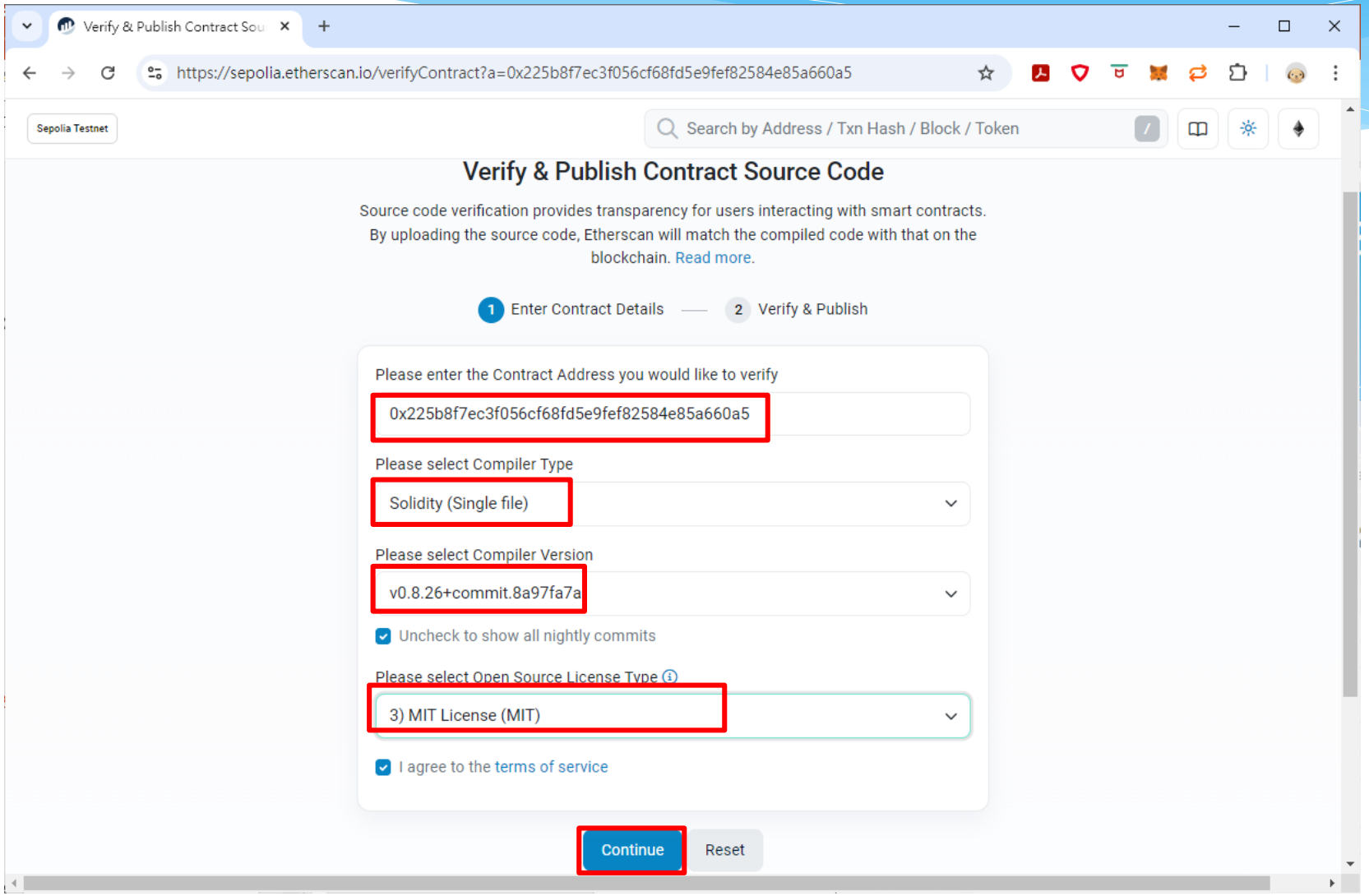


View & Publish



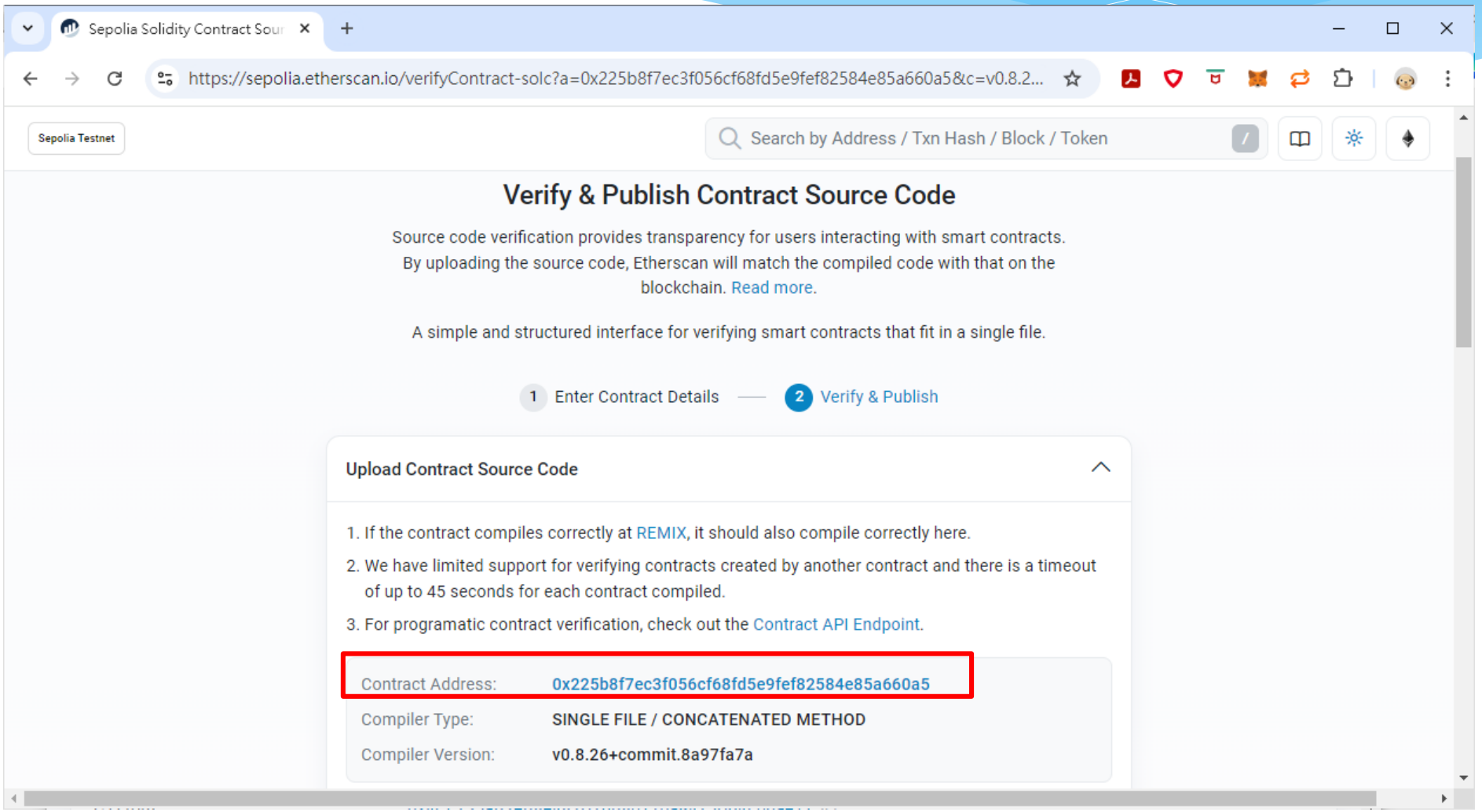
https://sepolia.etherscan.io/address/0x225b8f7ec3f056cf68fd5e9fef82584e85a660a5#code

View & Publish Contract Source Code

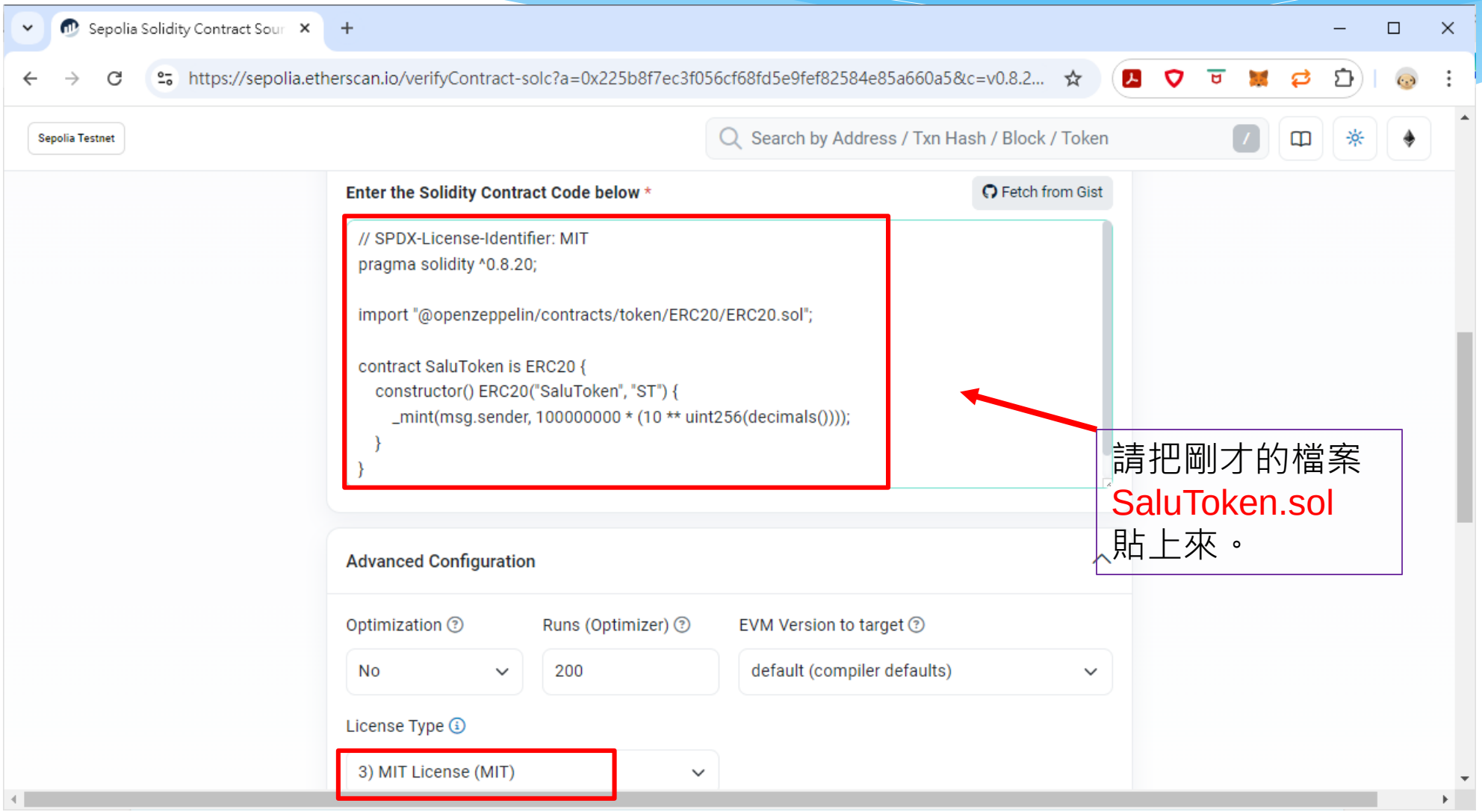


https://sepolia.etherscan.io/verifyContract?a=0x225b8f7ec3f056cf68fd5e9fef82584e85a660a5

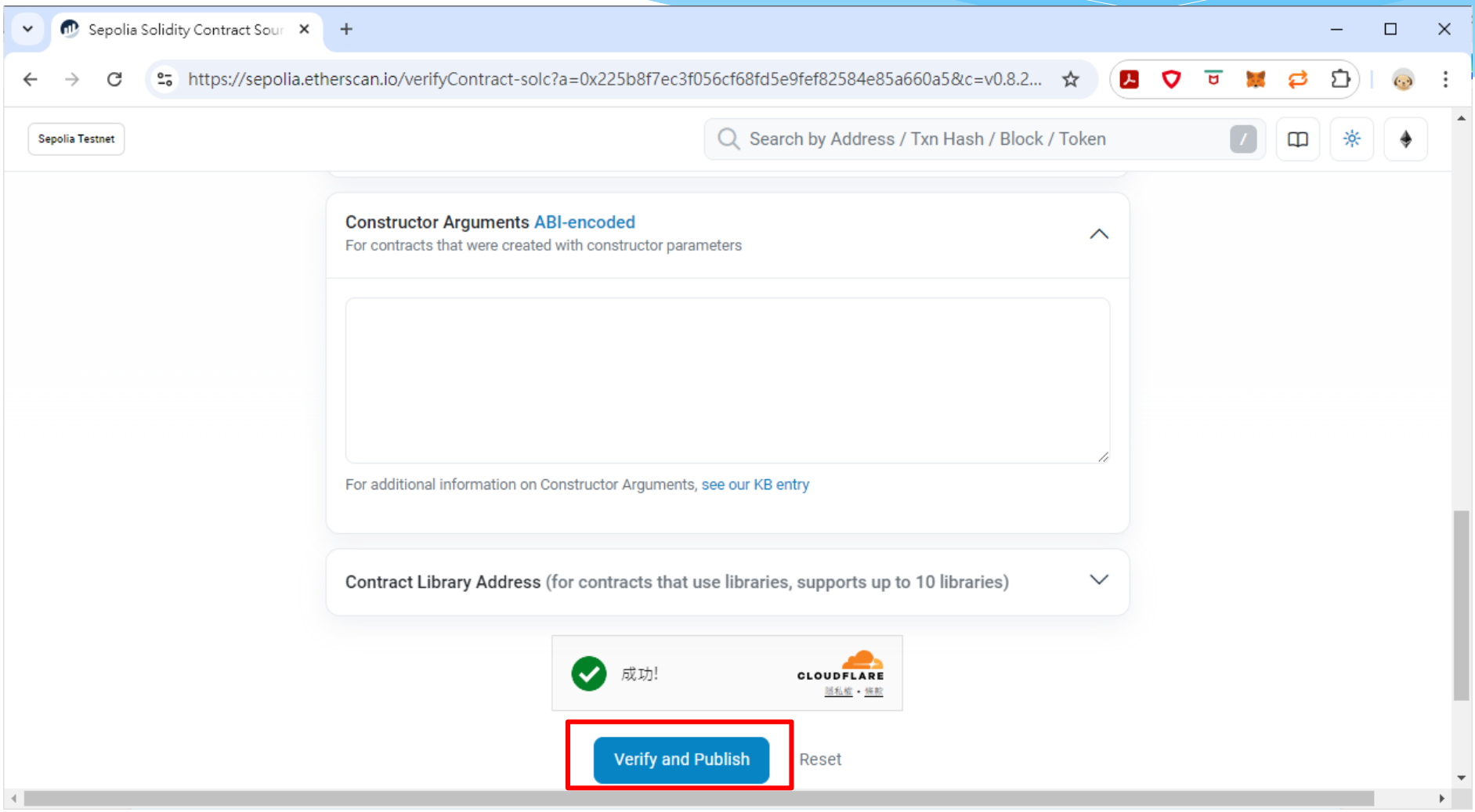
View & Publish Contract Source Code



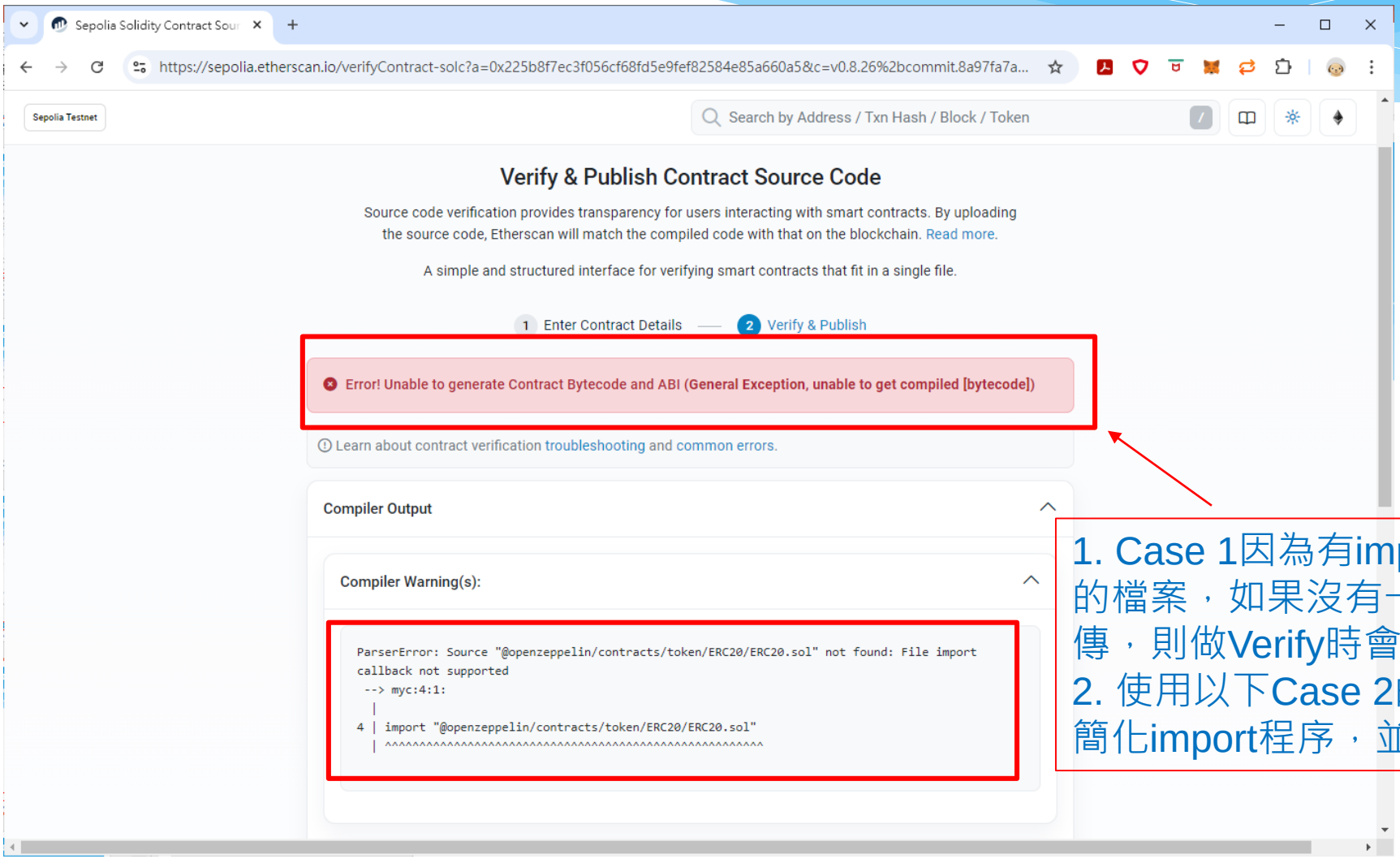
View & Publish Contract Source Code



View & Publish Contract Source Code



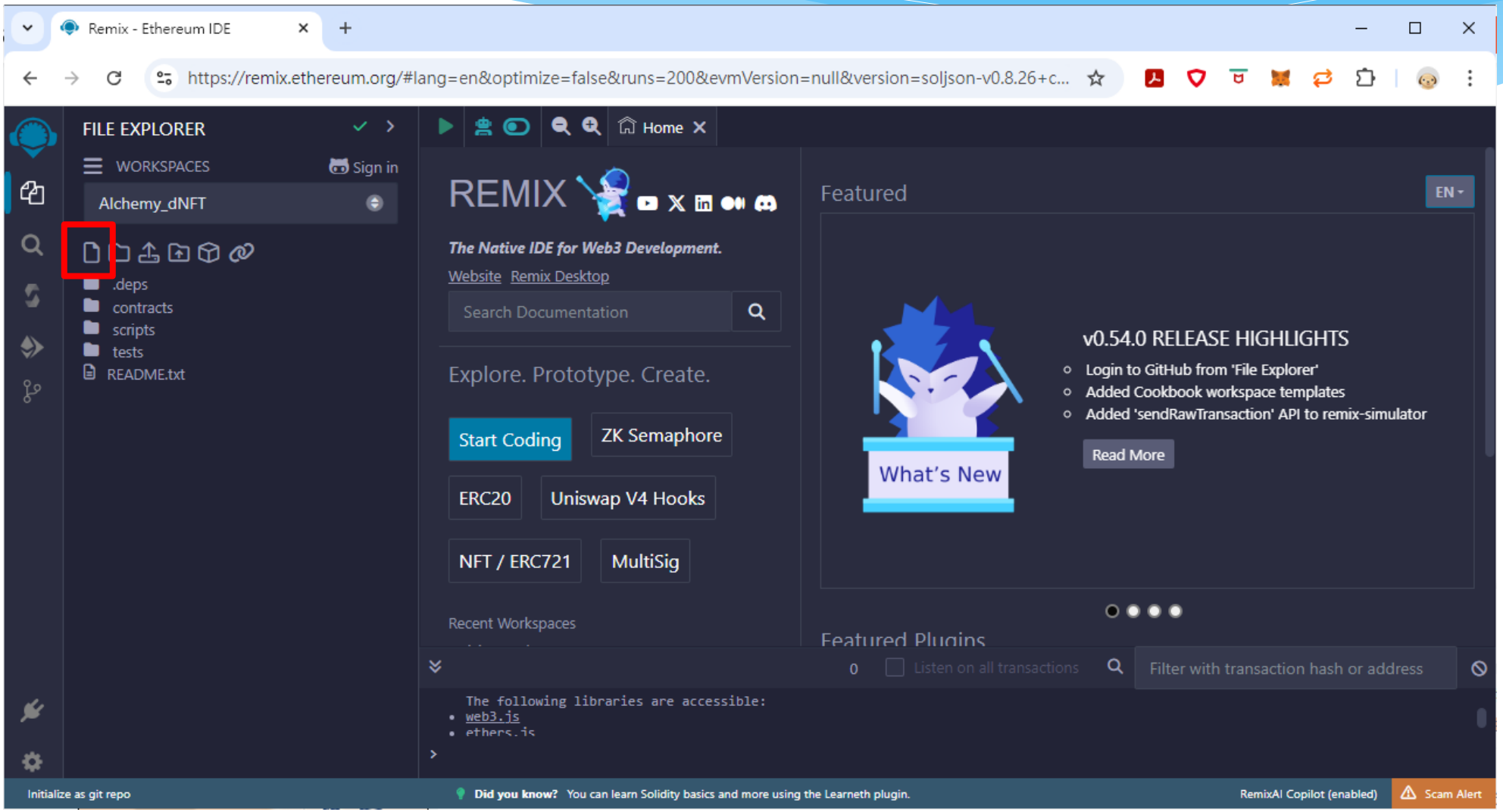
View & Publish Contract Source Code



1. Case 1因為有import OpenZeppelin的檔案，如果沒有一併把相關檔案上傳，則做Verify時會有問題。
2. 使用以下Case 2的Flatten方式可以簡化import程序，並使Verify正常無誤。

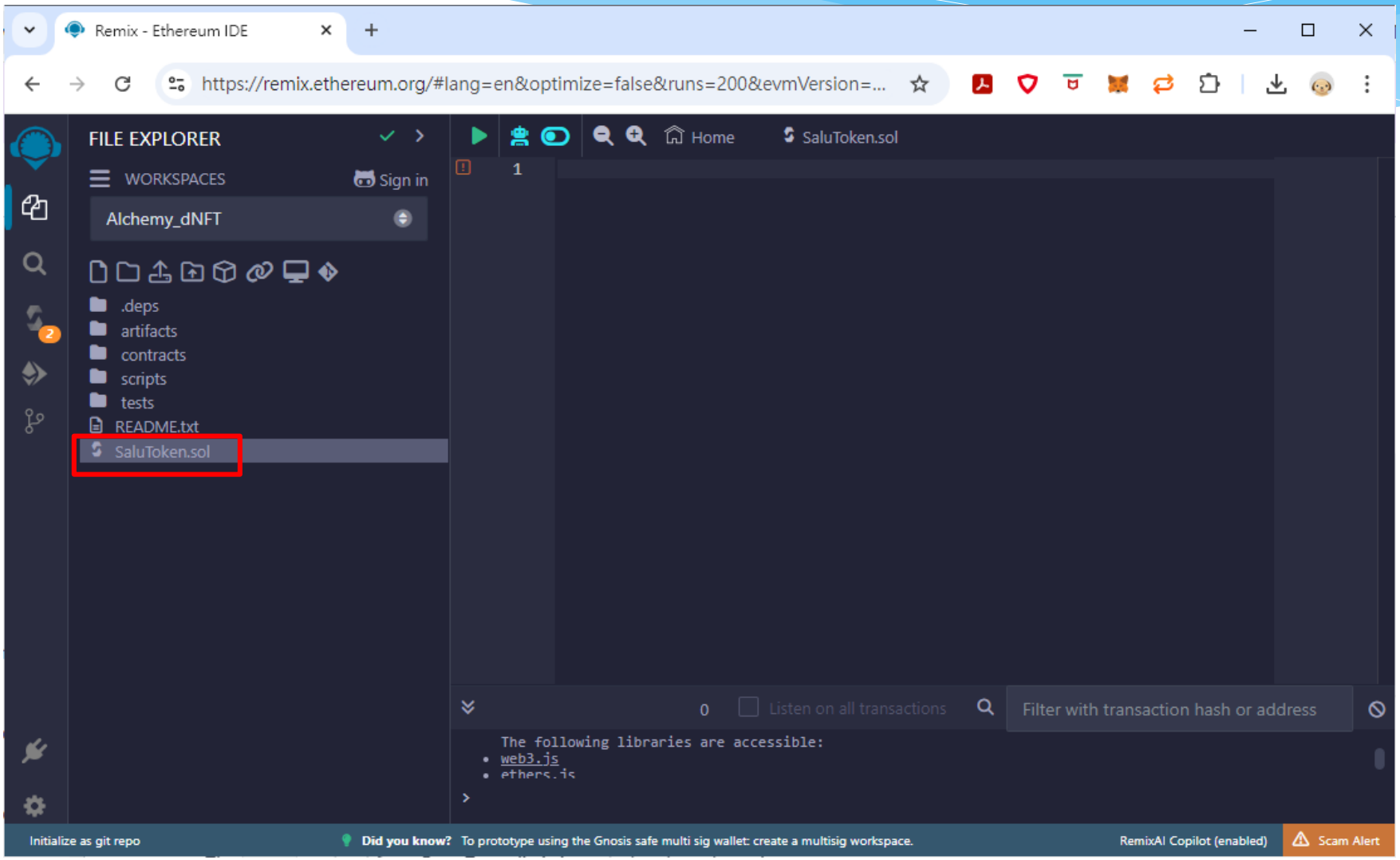
Case 2

Create an ERC-20 Token

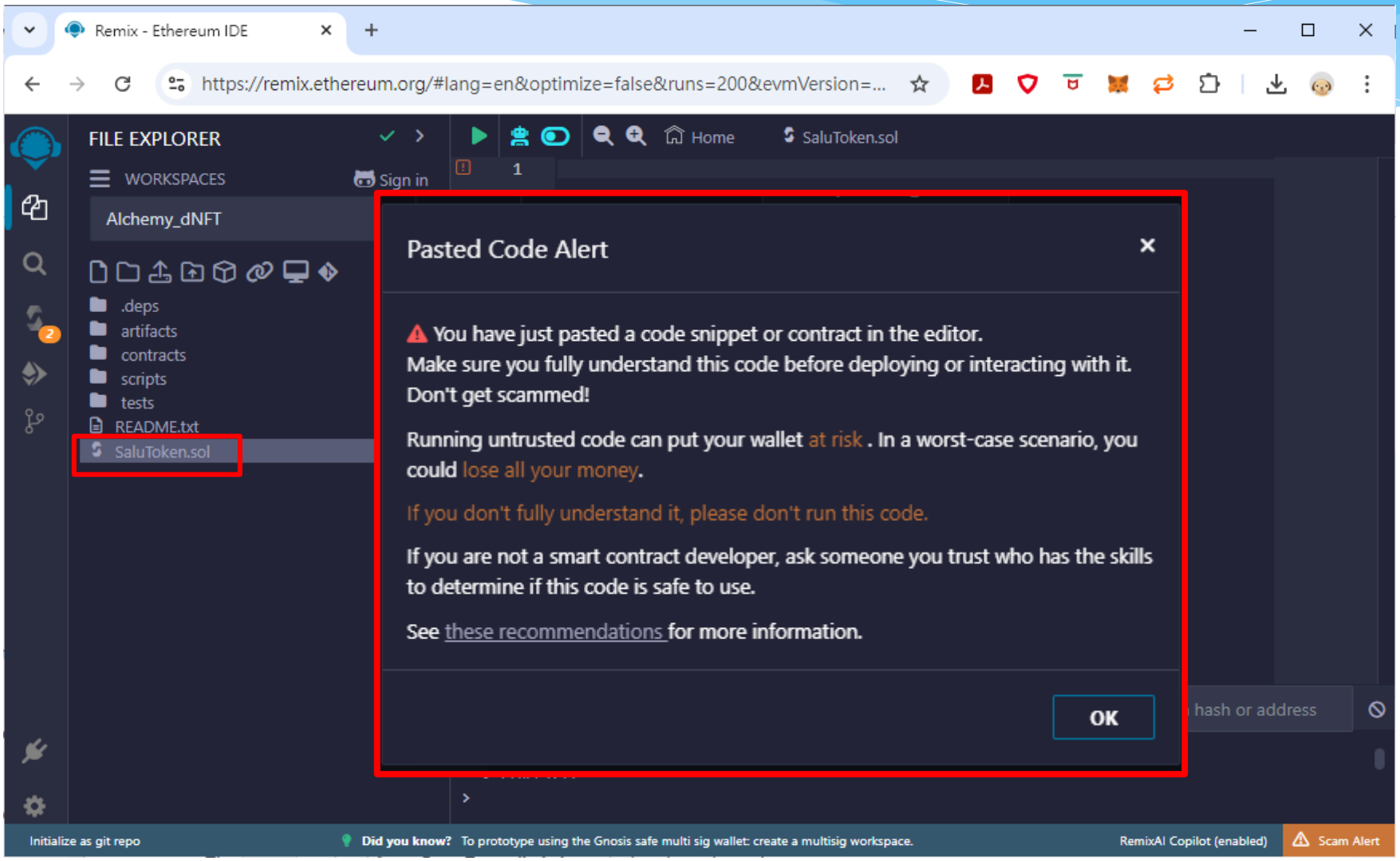


https://remix.ethereum.org

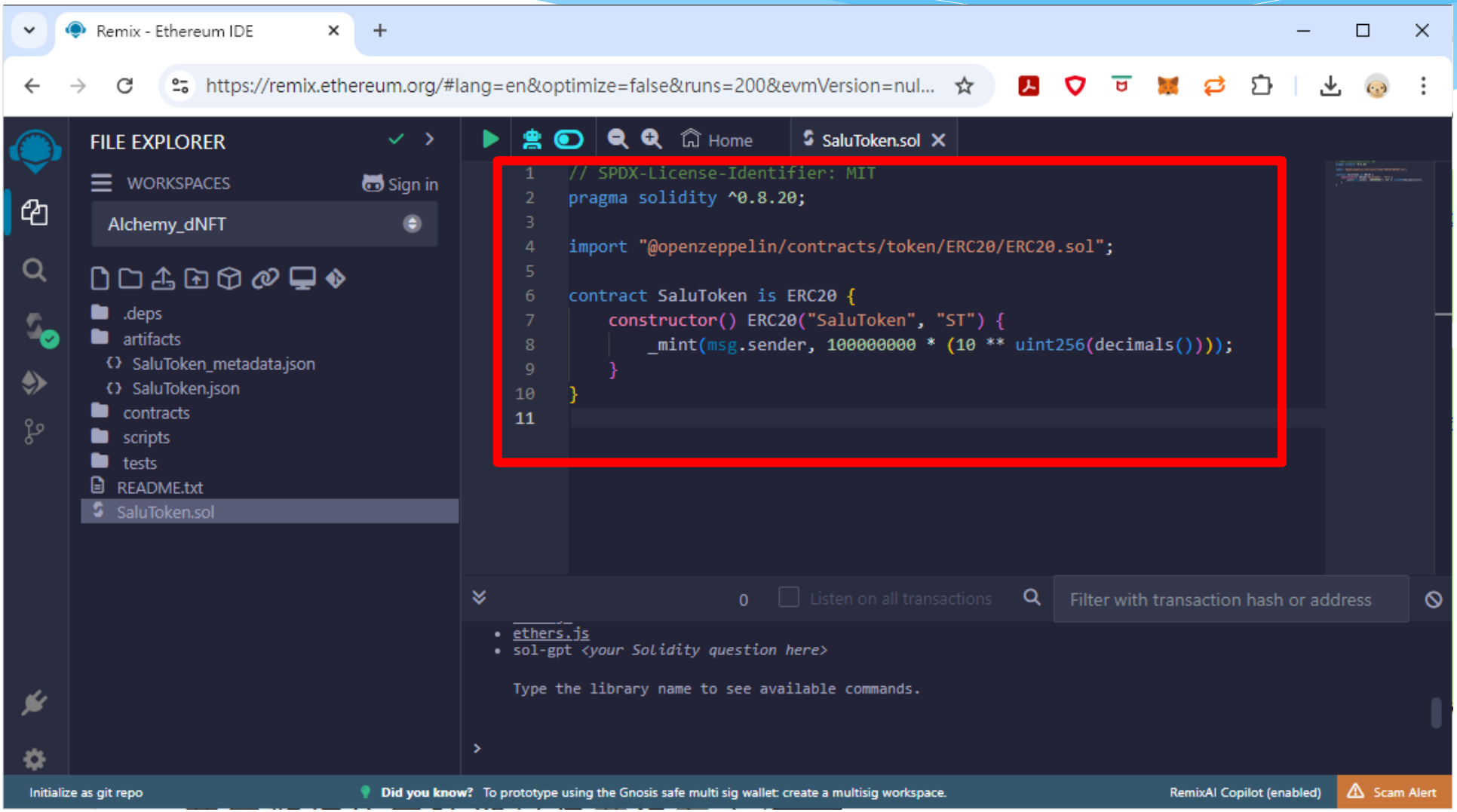
SaluToken.sol



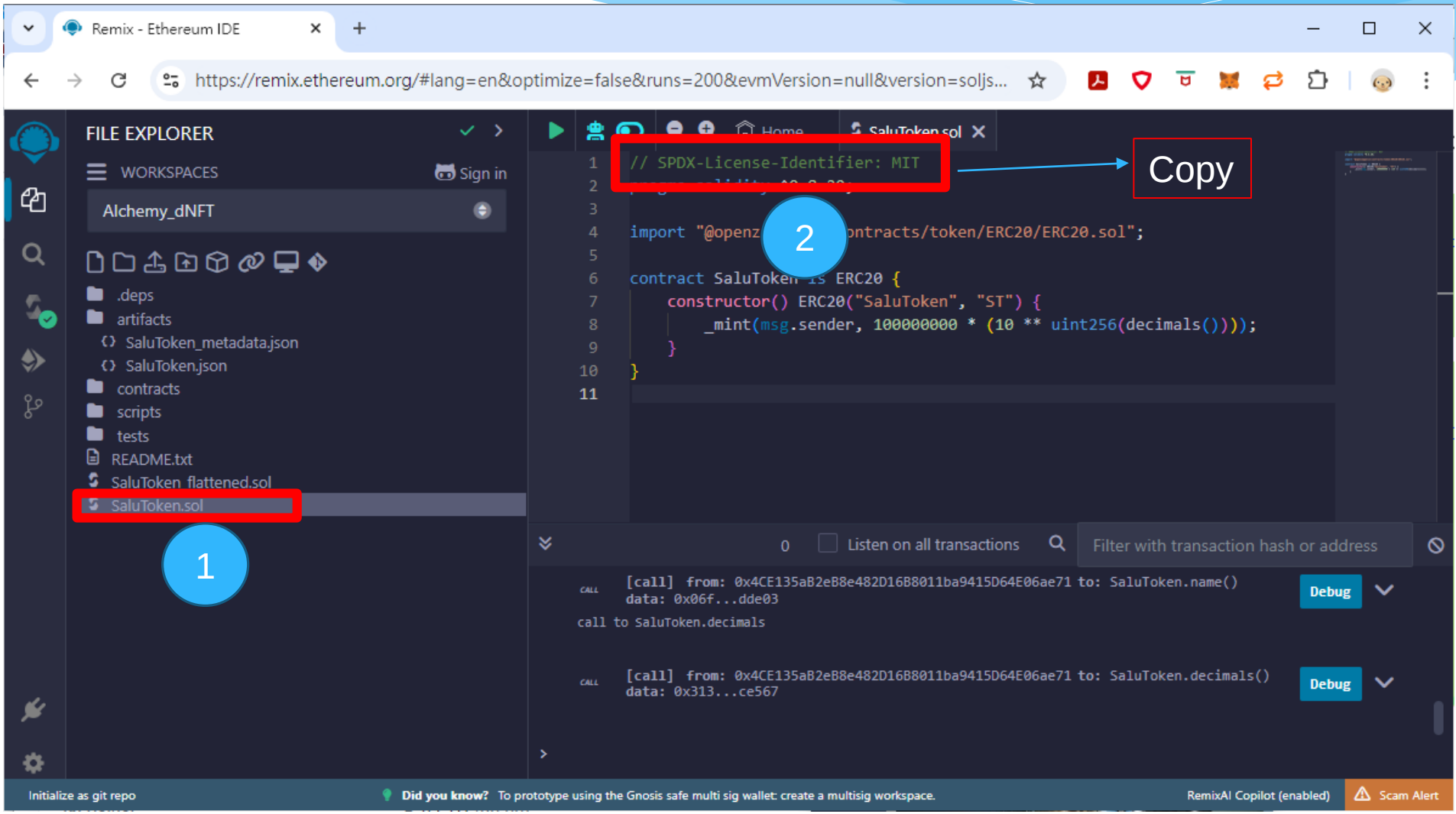
SaluToken.sol



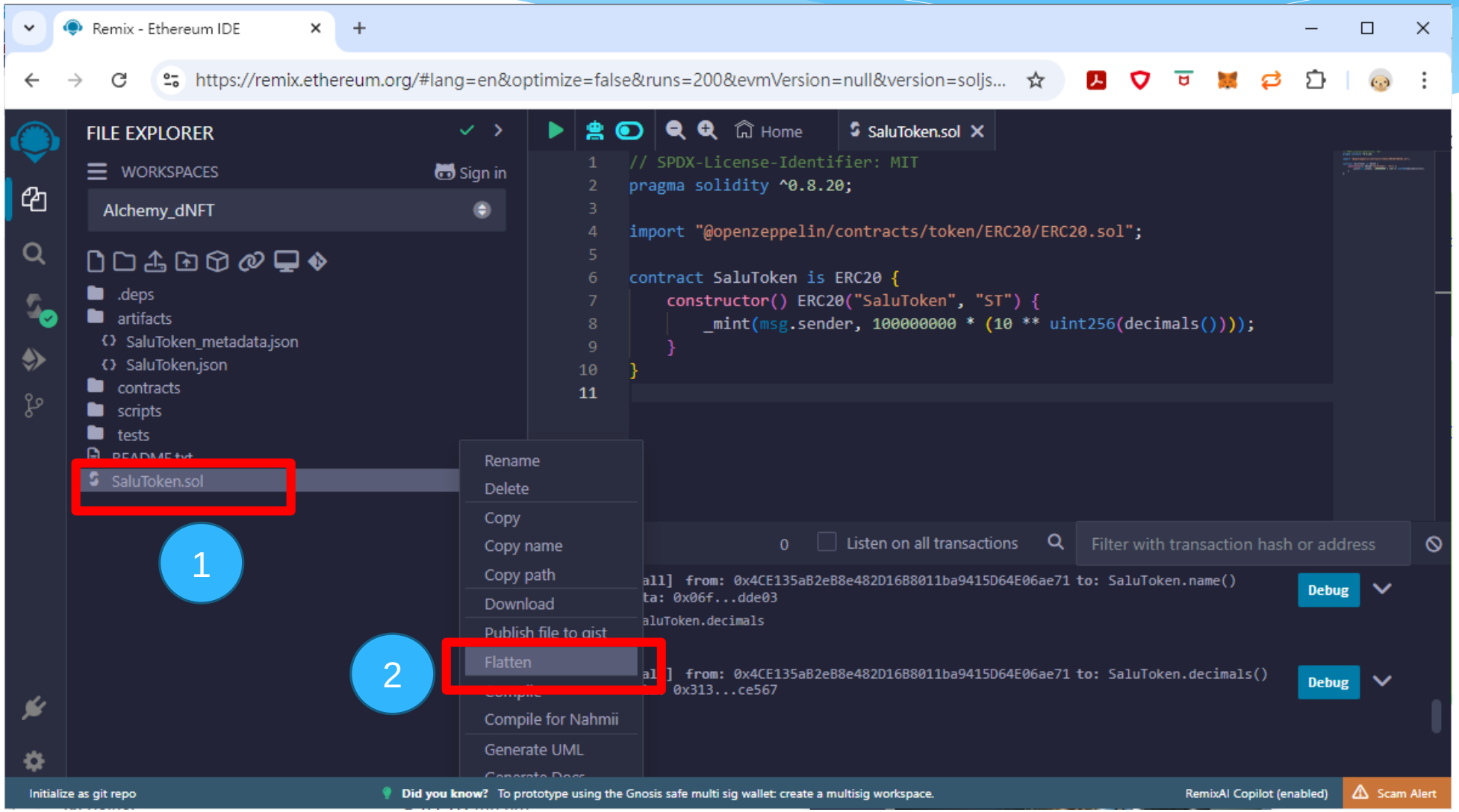
SaluToken.sol



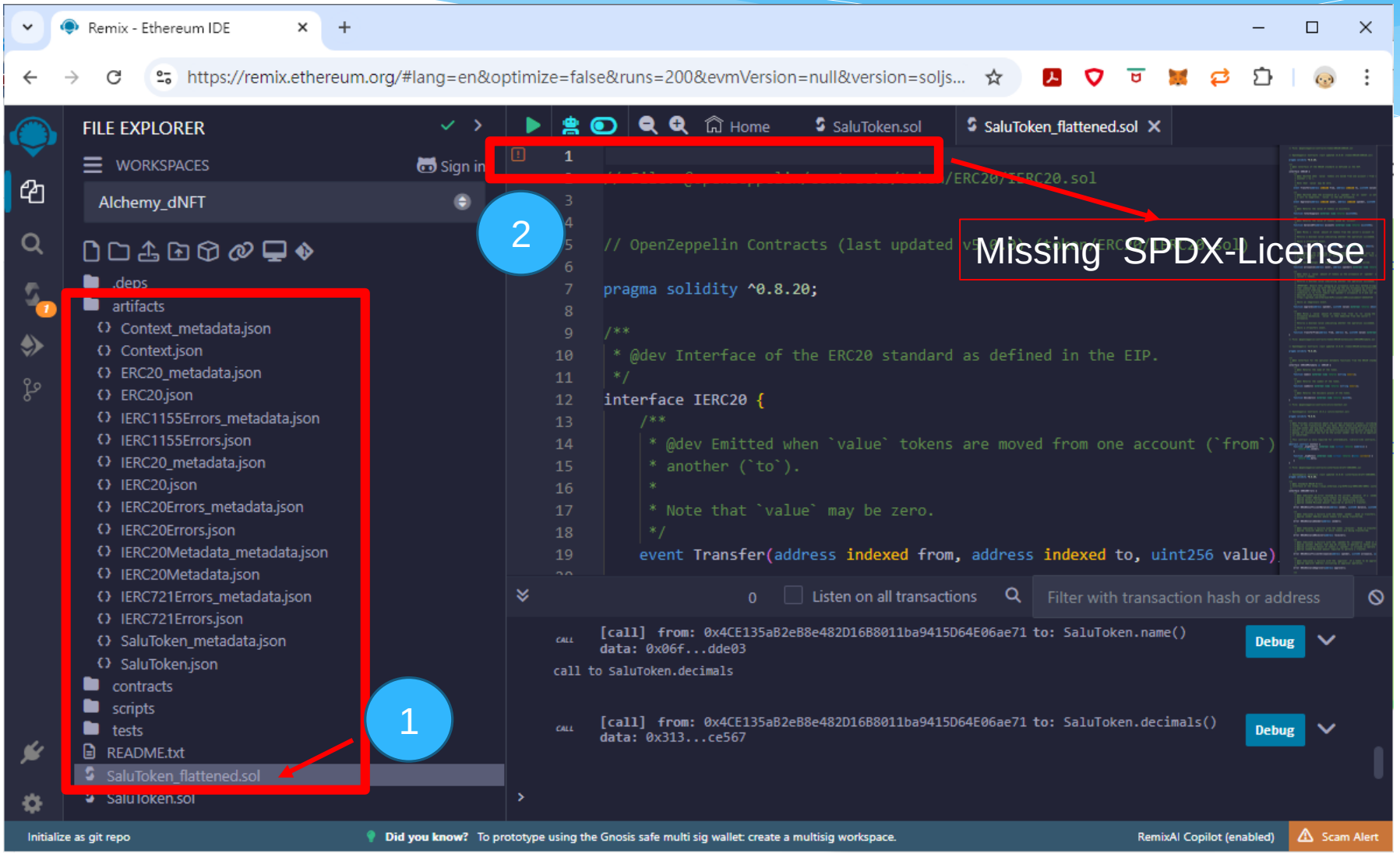
Flatten SaluToken.sol



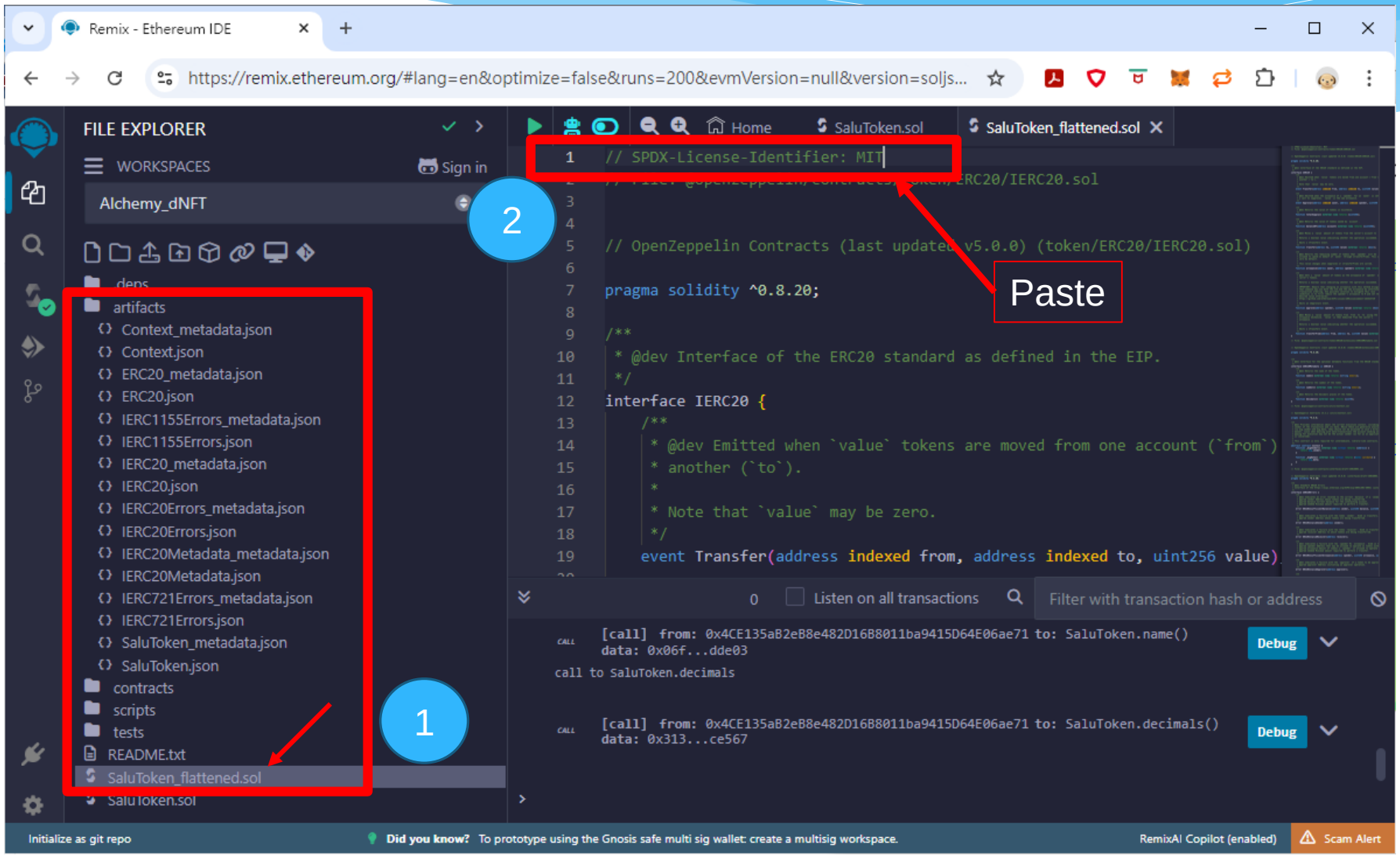
Flatten SaluToken.sol



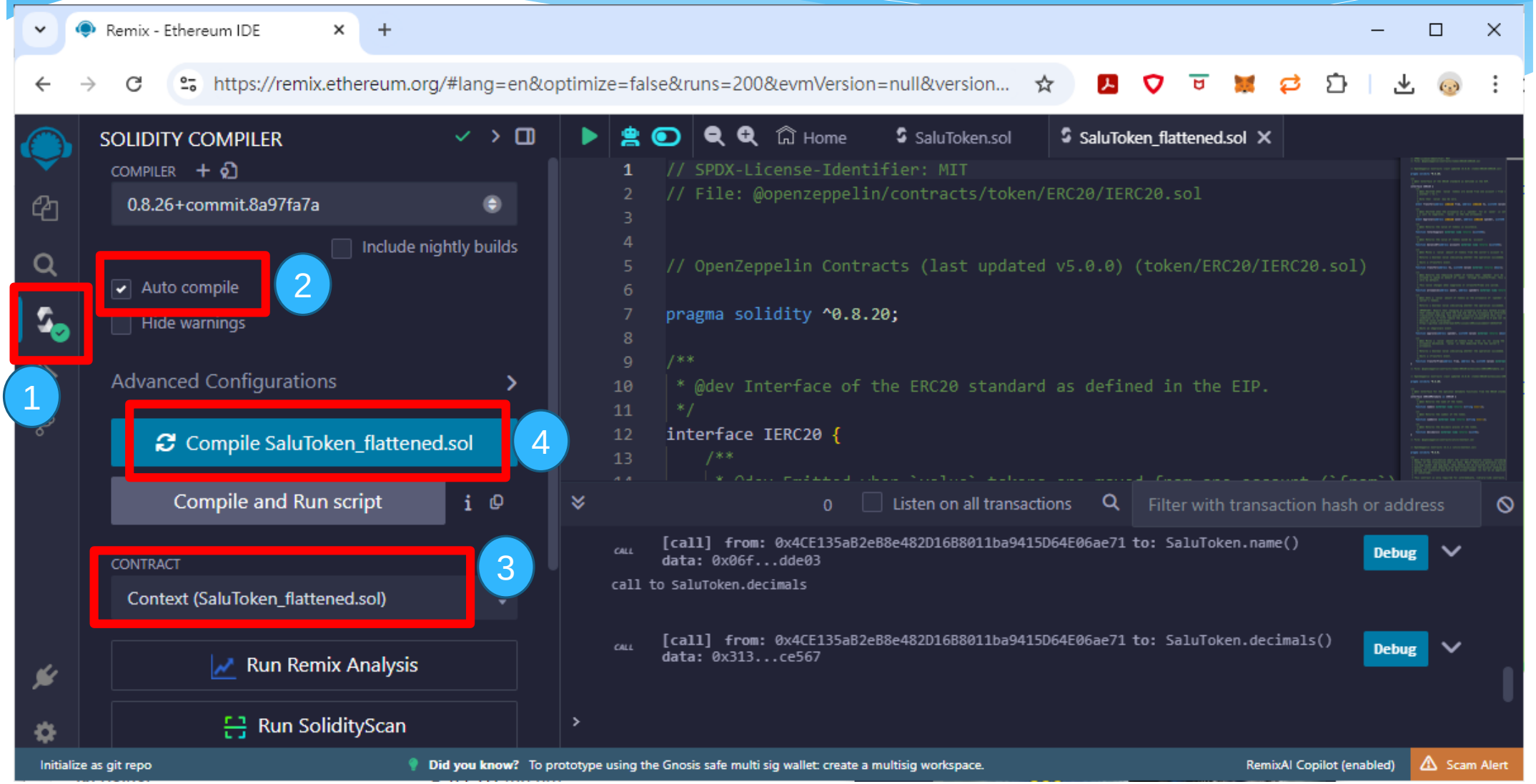
Flatten SaluToken.sol



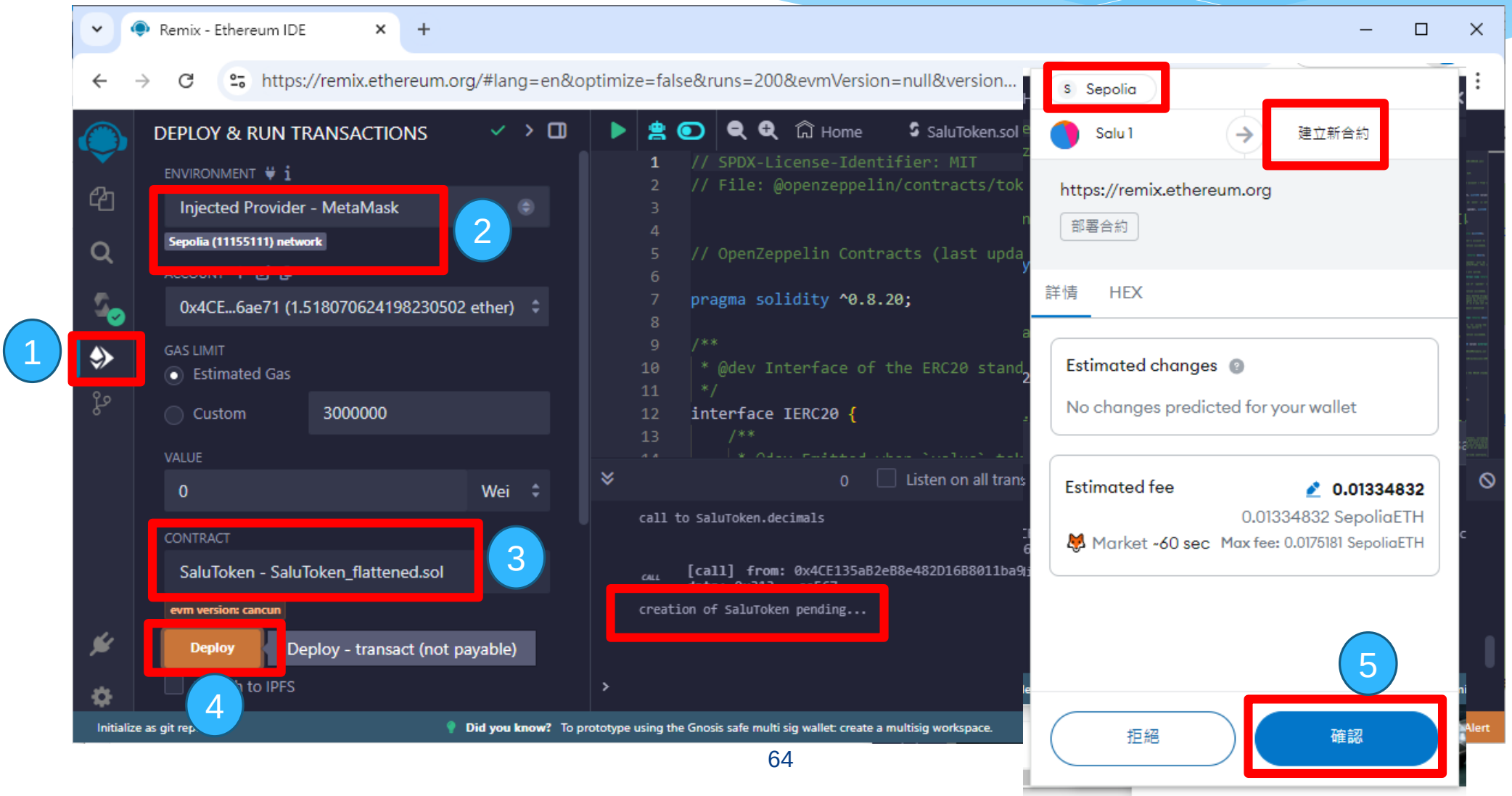
Flatten SaluToken.sol



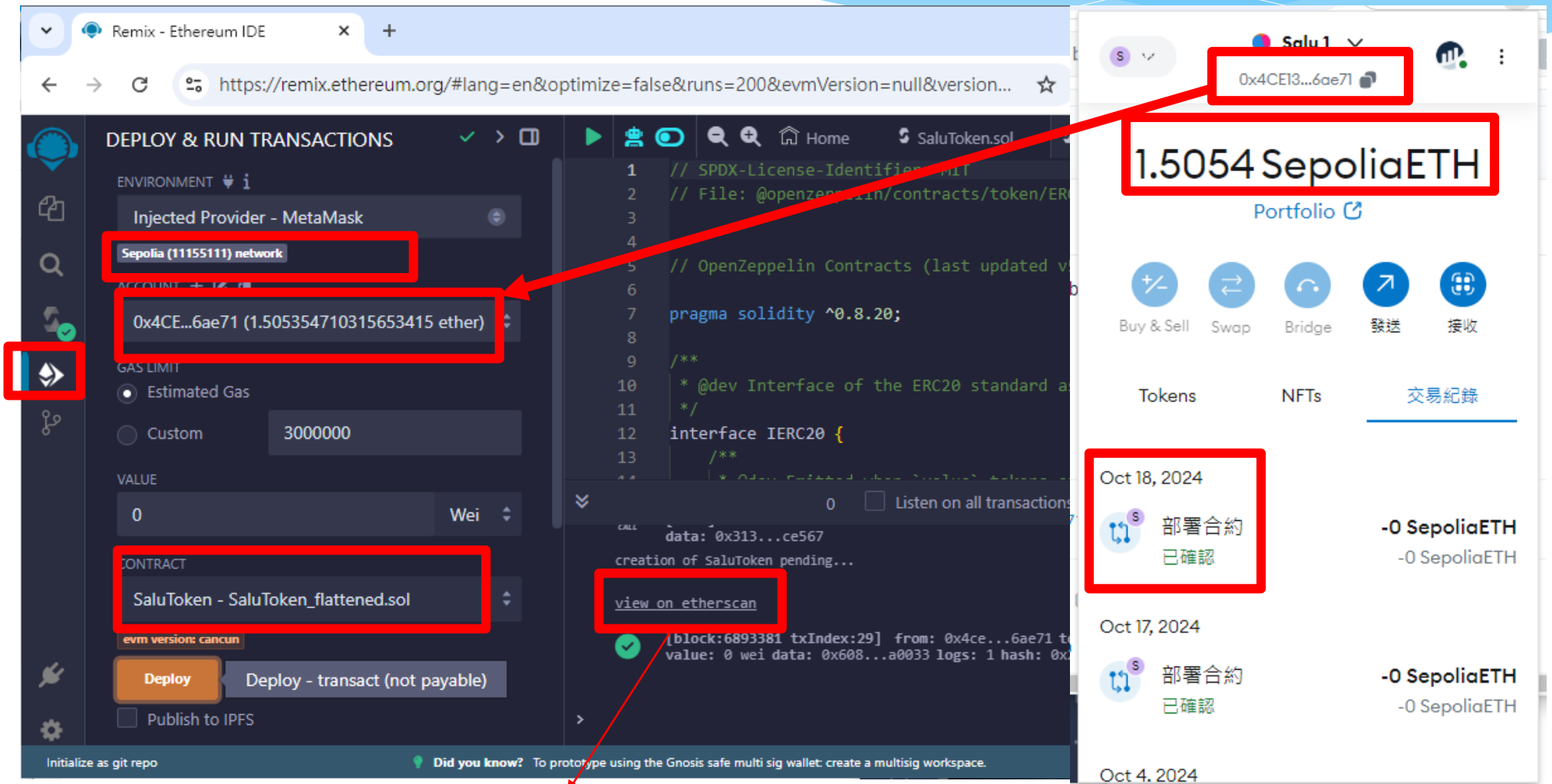
Compile Smart Contract



Deploy an ERC-20 Token

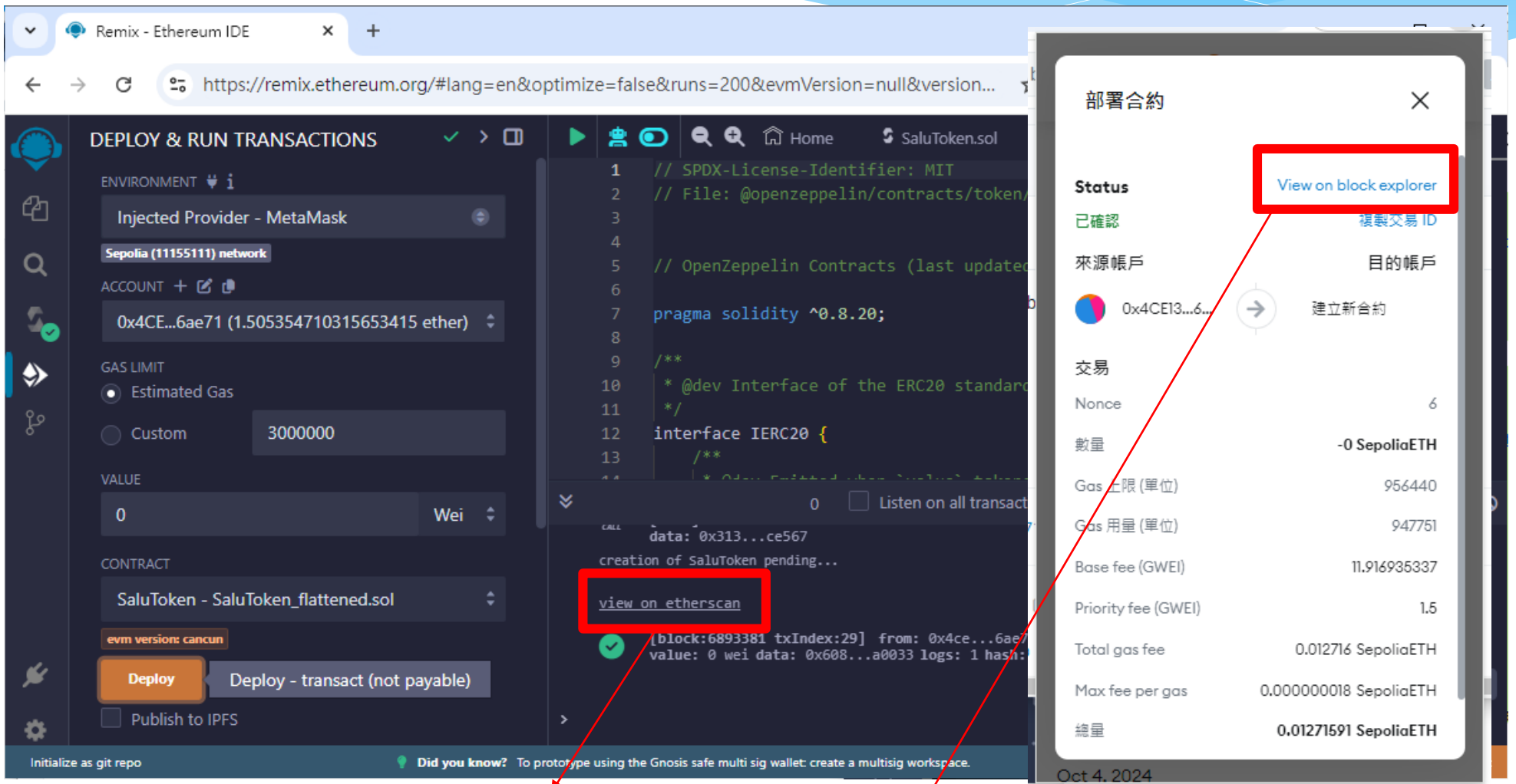


Deploy and Run Transaction



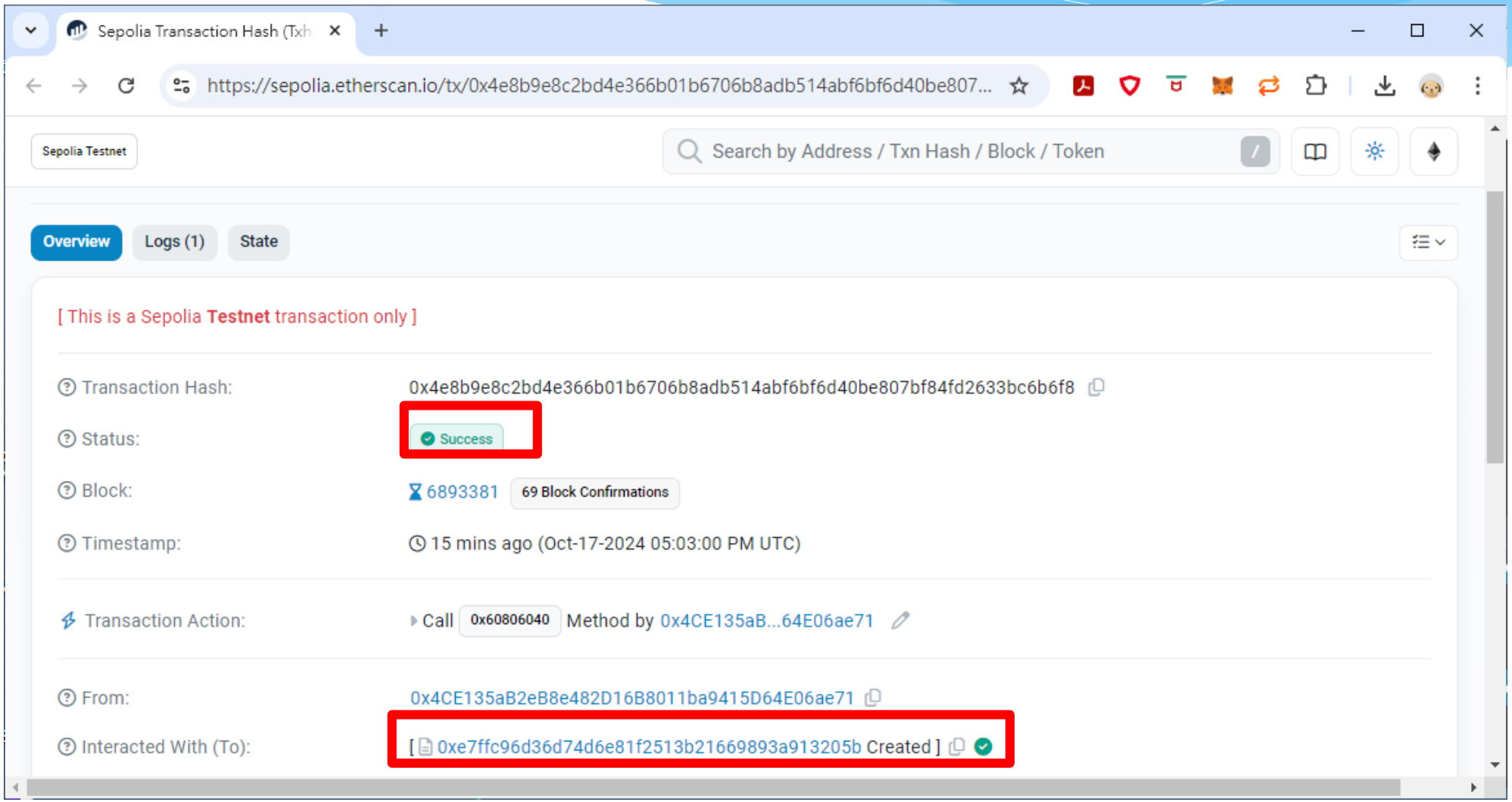
https://sepolia.etherscan.io/tx/0x4e8b9e8c2bd4e366b01b6706b8adb514abf6bf6d40be807bf84fd2633bc6b6f8

Deploy and Run Transaction



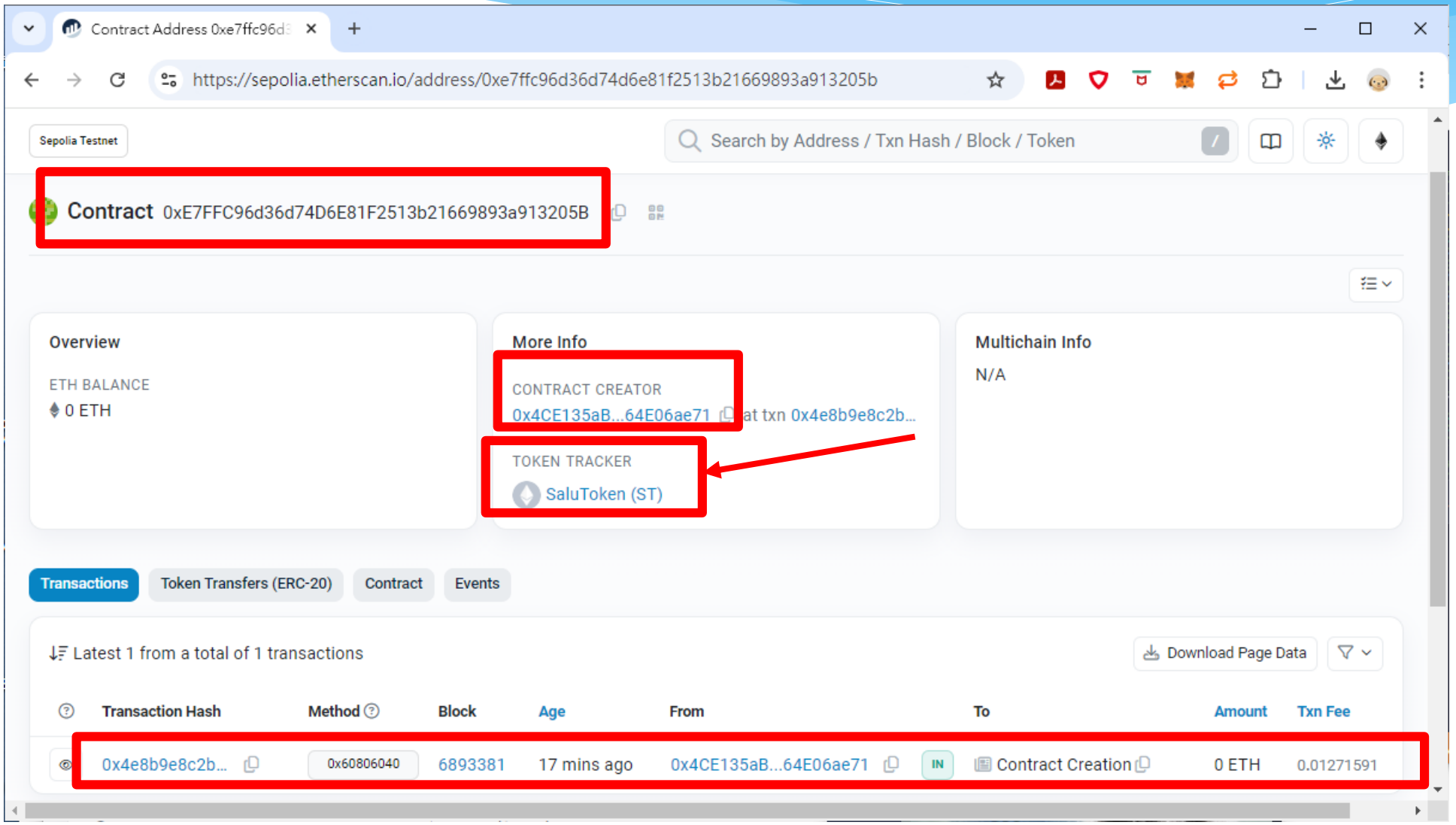
https://sepolia.etherscan.io/tx/0x4e8b9e8c2bd4e366b01b6706b8adb514abf6bf6d40be807bf84fd2633bc6b6f8

View Transaction on Sepolia



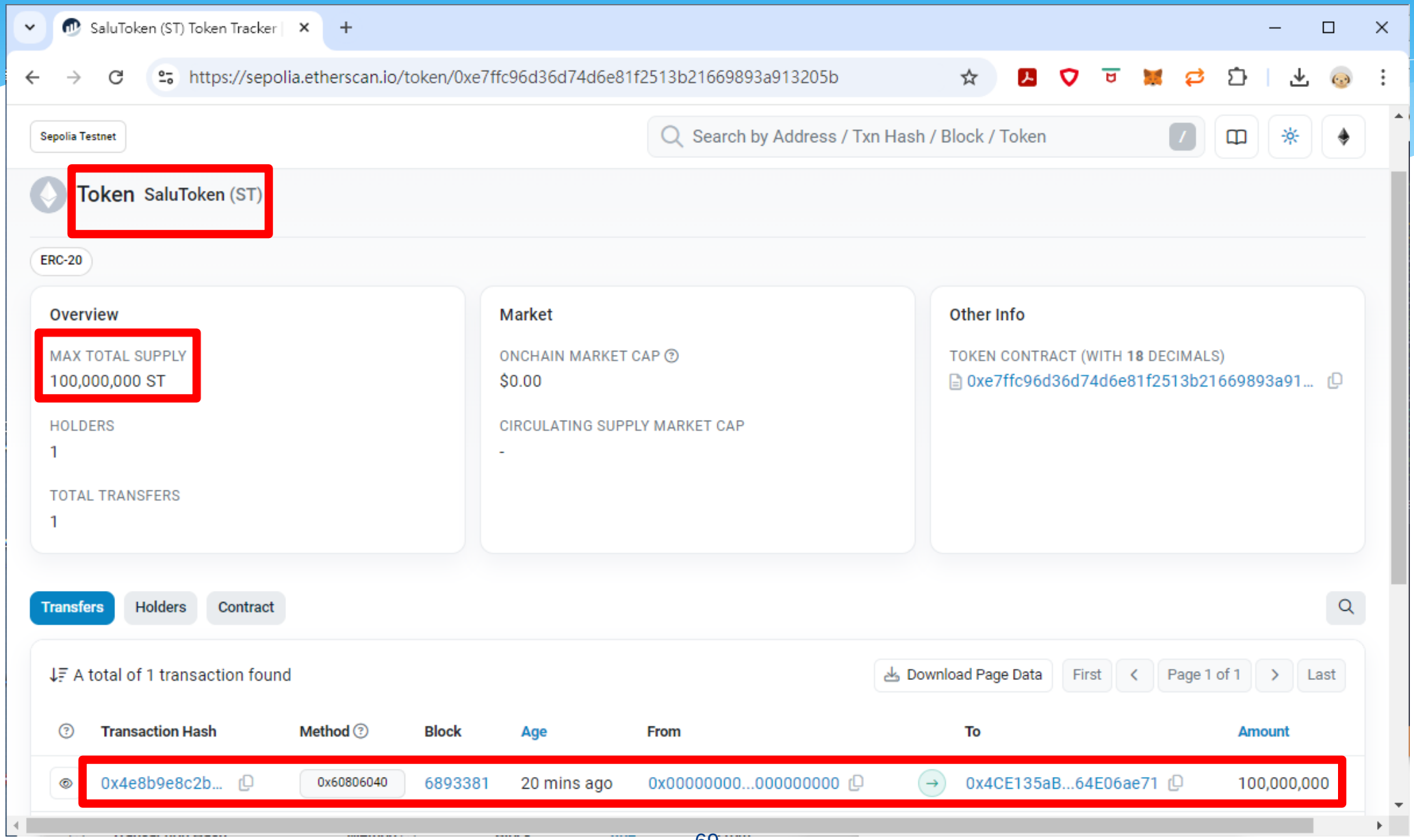
https://sepolia.etherscan.io/tx/0x4e8b9e8c2bd4e366b01b6706b8adb514abf6bf6d40be807bf84fd2633bc6b6f8

Smart Contract Information



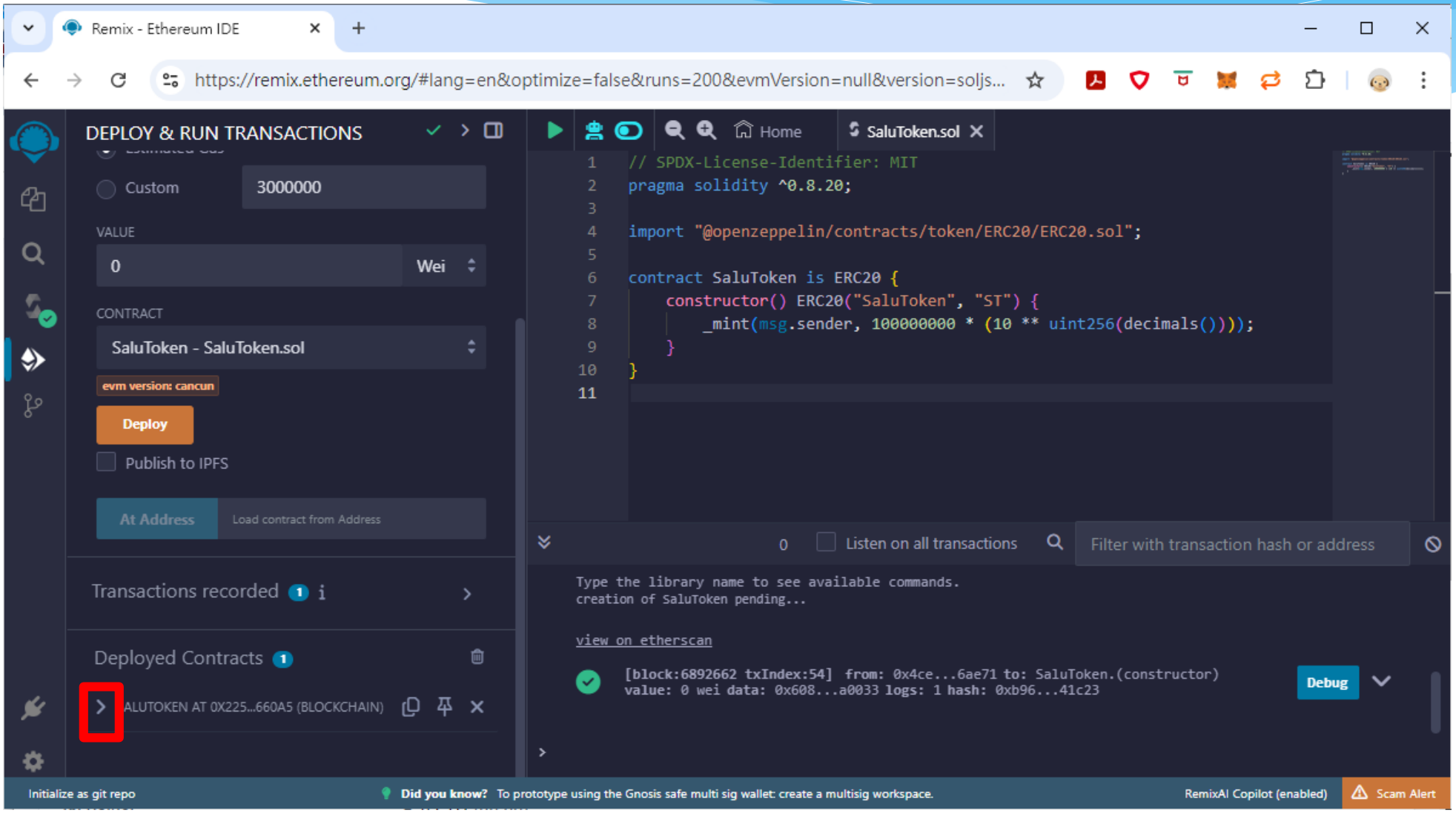
https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b

Smart Contract Information



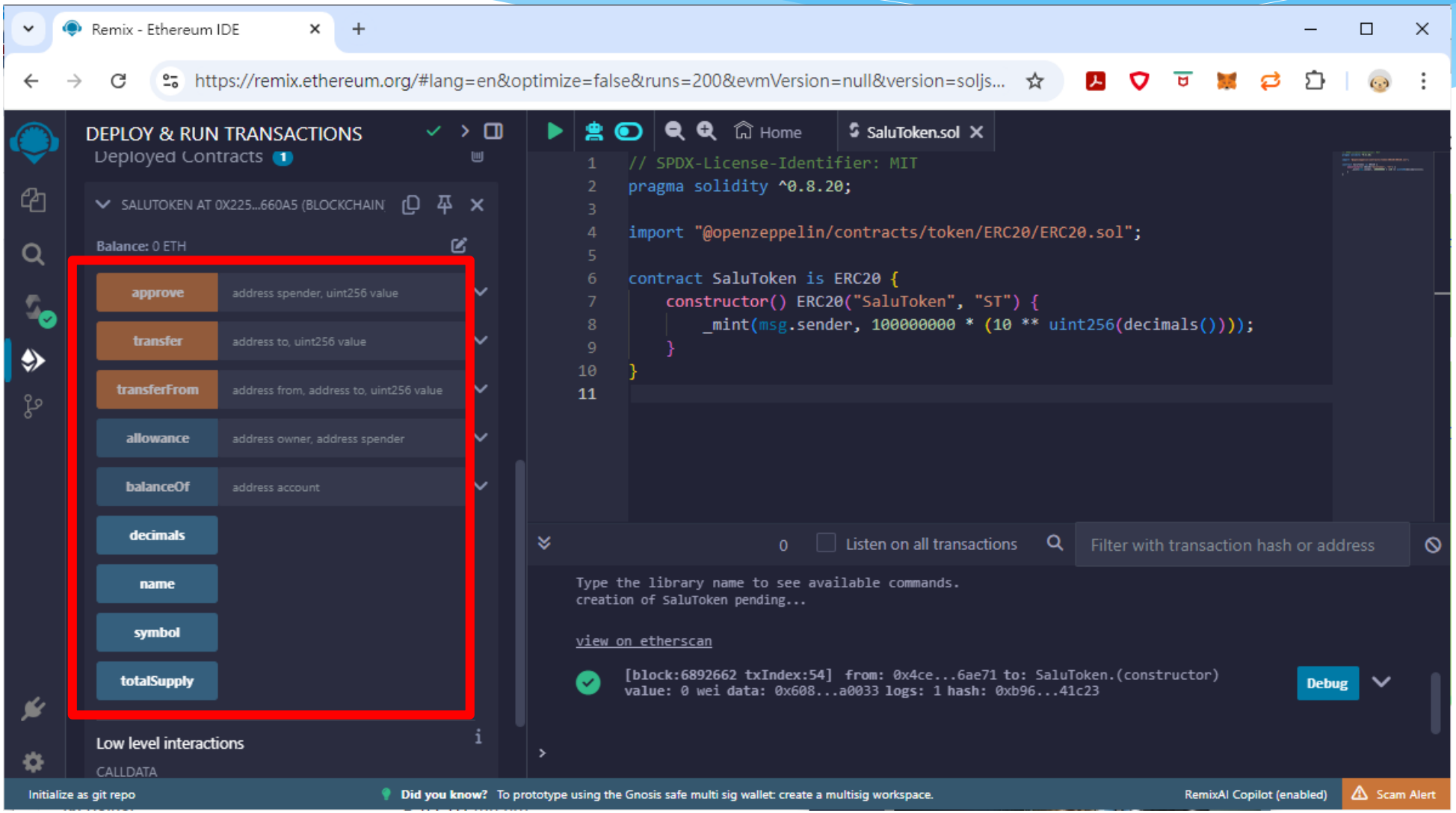
https://sepolia.etherscan.io/token/0xe7ffc96d36d74d6e81f2513b21669893a913205b

Deploy and Run Transaction

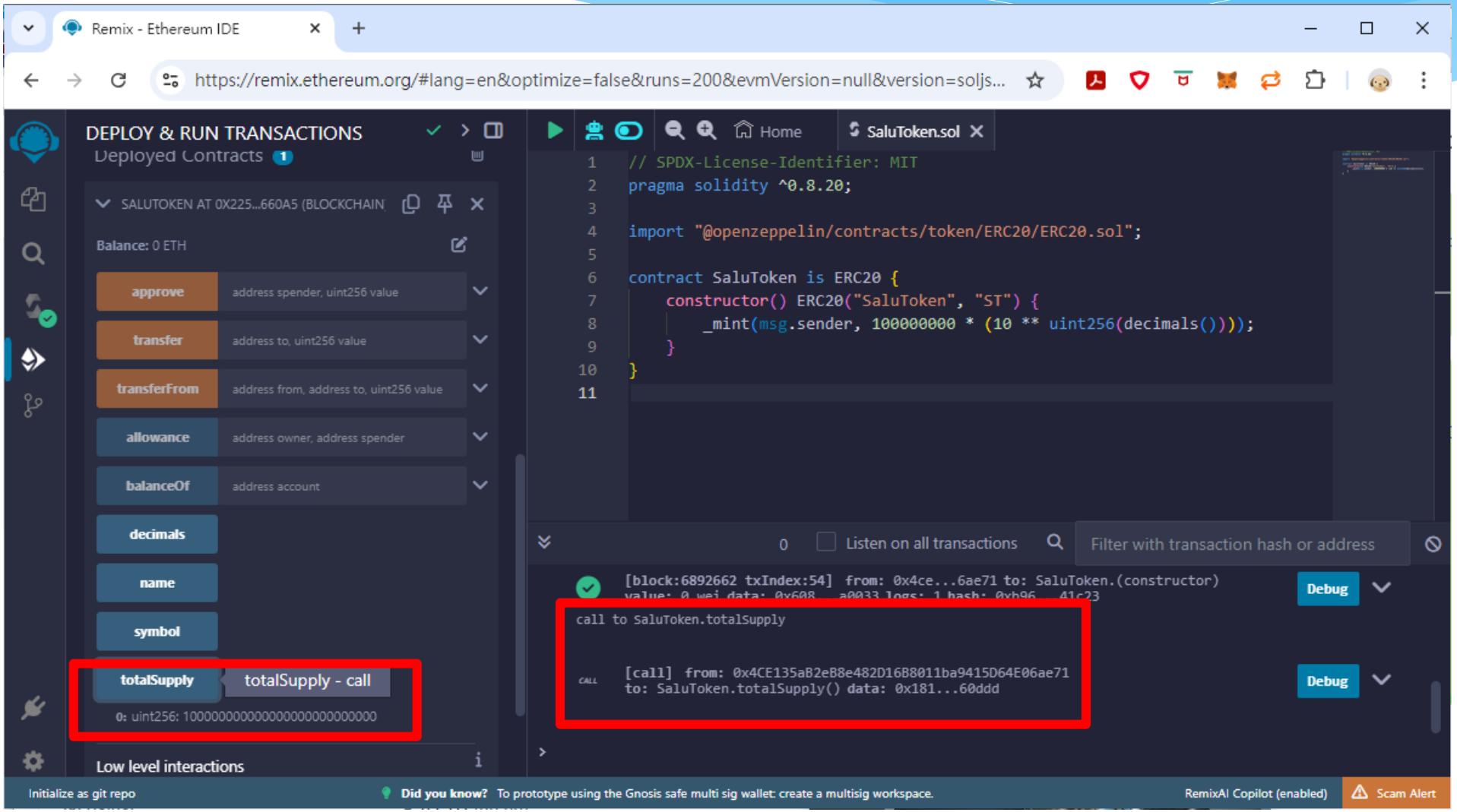


Contract address : 0x225b8F7ec3F056CF68FD5E9FEf82584E85A660A5

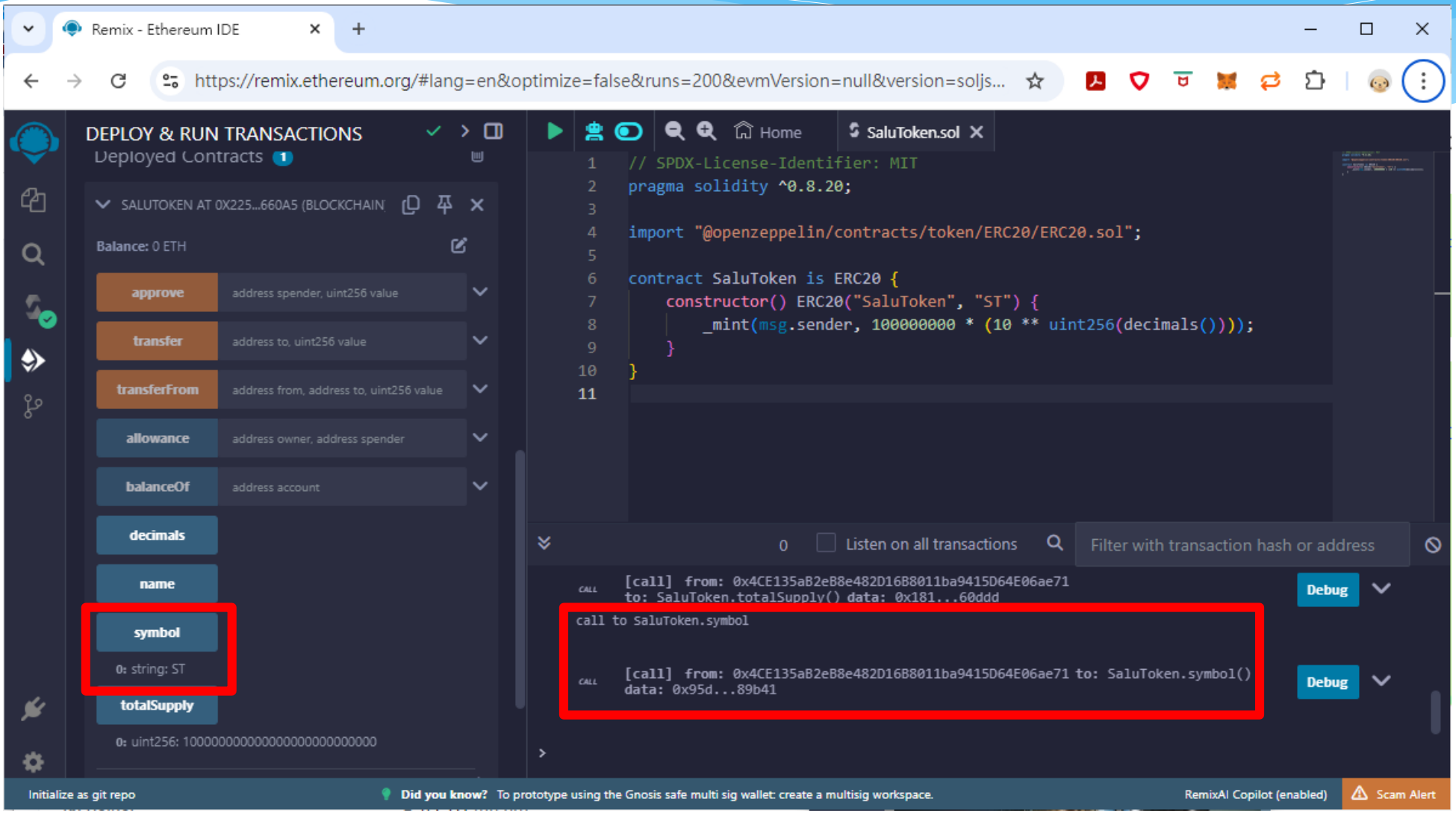
Deploy and Run Transaction



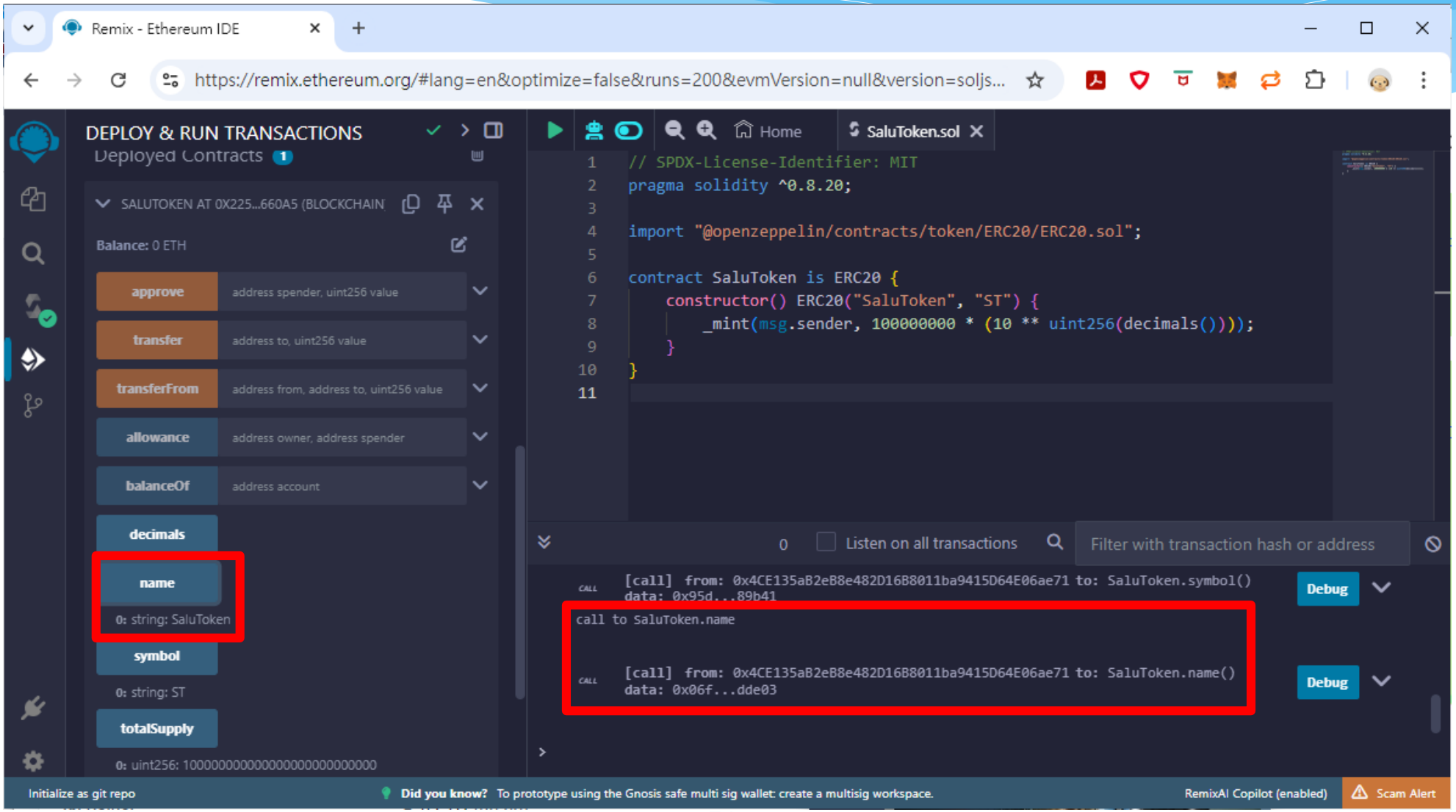
Call totalSupply()



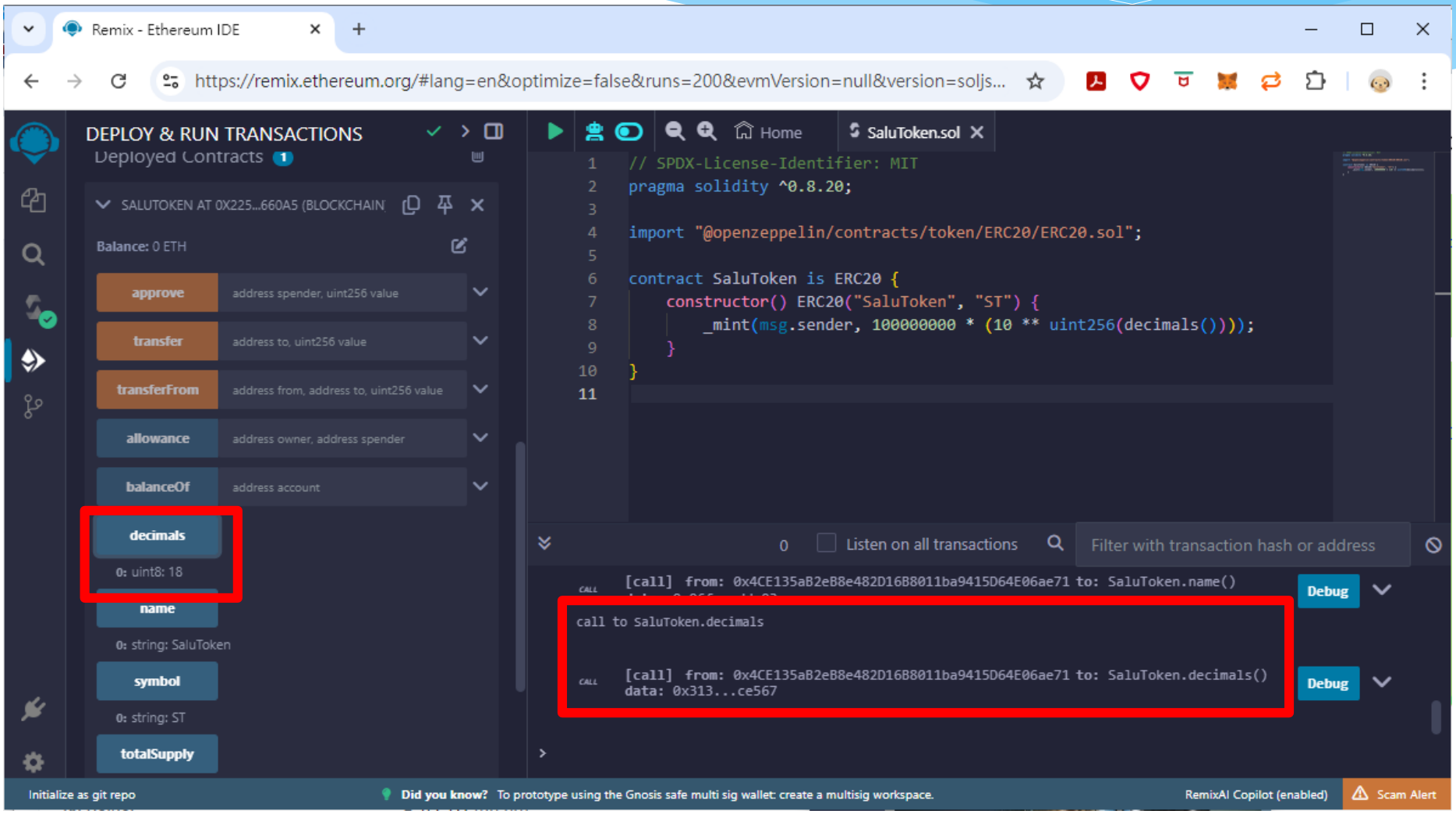
Call symbol()


Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljs...
DEPLOY & RUN TRANSACTIONS
Deployed Contracts 1
SALUTOKEN AT 0X225...660A5 (BLOCKCHAIN
Balance: 0 ETH
approve
address spender, uint256 value
transfer
address to, uint256 value
transferFrom
address from, address to, uint256 value
allowance
address owner, address spender
balanceOf
address account
decimals
name
symbol
0: string: ST
totalSupply
0: uint256: 100000000000000000000000000
Home
SaluToken.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6 contract SaluToken is ERC20 {
7 constructor() ERC20("SaluToken", "ST") {
8 _mint(msg.sender, 100000000 * (10 ** uint256(decimals())));
9 }
10 }
11
0
Listen on all transactions
Filter with transaction hash or address
CALL [call] from: 0x4CE135aB2eB8e482D16B8011ba9415D64E06ae71 to: SaluToken.totalSupply() data: 0x181...60ddd
Debug
call to SaluToken.symbol
CALL [call] from: 0x4CE135aB2eB8e482D16B8011ba9415D64E06ae71 to: SaluToken.symbol() data: 0x95d...89b41
Debug
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

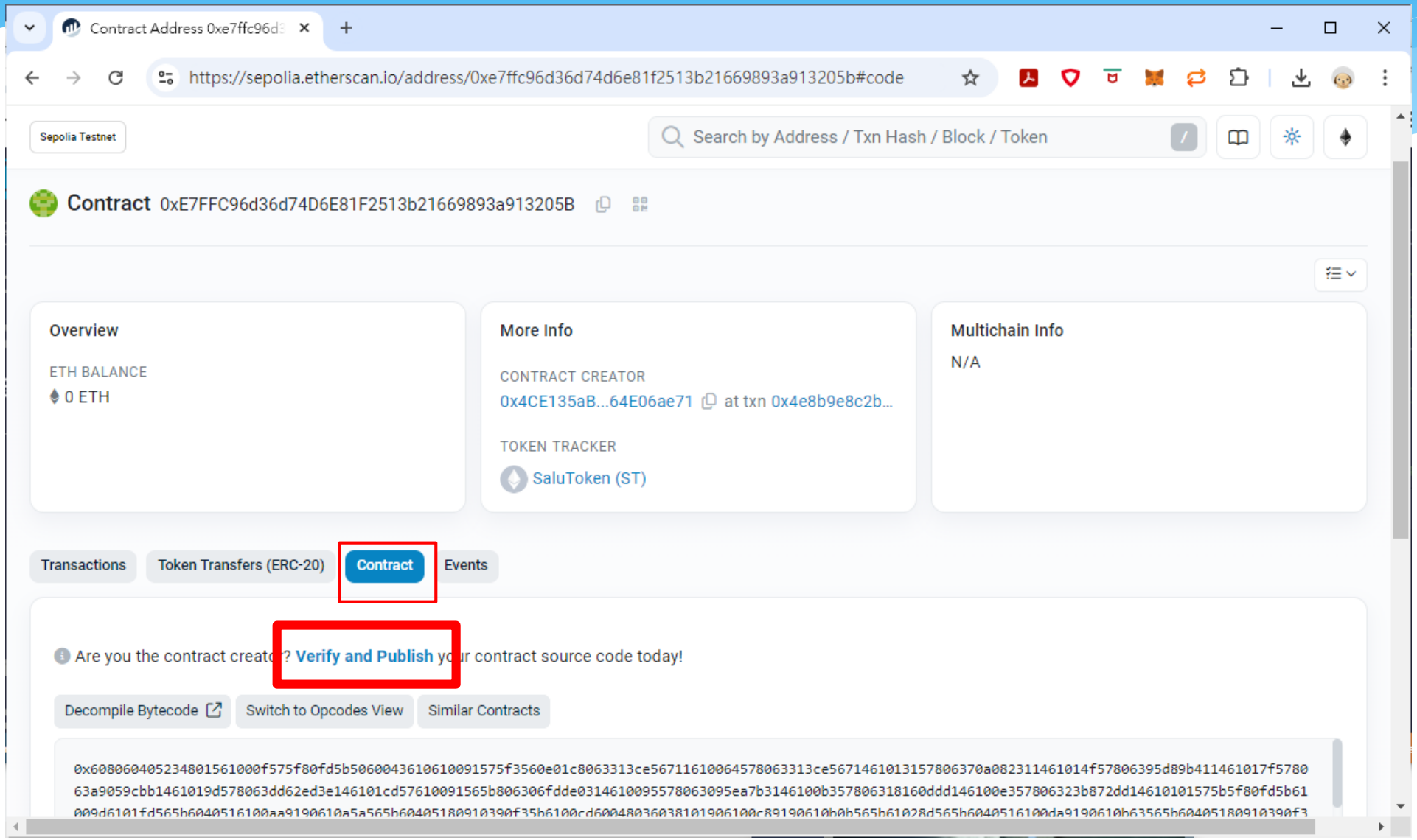
Call name()


Remix - Ethereum IDE
https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljs...
DEPLOY & RUN TRANSACTIONS
Deployed Contracts 1
SALUTOKEN AT 0X225...660A5 (BLOCKCHAIN
Balance: 0 ETH
approve
address spender, uint256 value
transfer
address to, uint256 value
transferFrom
address from, address to, uint256 value
allowance
address owner, address spender
balanceOf
address account
decimals
name
0: string: SaluToken
symbol
0: string: ST
totalSupply
0: uint256: 100000000000000000000000000
Home
SaluToken.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5
6 contract SaluToken is ERC20 {
7 constructor() ERC20("SaluToken", "ST") {
8 _mint(msg.sender, 100000000 * (10 ** uint256(decimals())));
9 }
10 }
11
0
Listen on all transactions
Filter with transaction hash or address
[call] from: 0x4CE135aB2eB8e482D16B8011ba9415D64E06ae71 to: SaluToken.symbol()
data: 0x95d...89b41
Debug
call to SaluToken.name
[call] from: 0x4CE135aB2eB8e482D16B8011ba9415D64E06ae71 to: SaluToken.name()
data: 0x06f...dde03
Debug
Initialize as git repo
Did you know? To prototype using the Gnosis safe multi sig wallet: create a multisig workspace.
RemixAI Copilot (enabled)
Scam Alert

Call decimals()

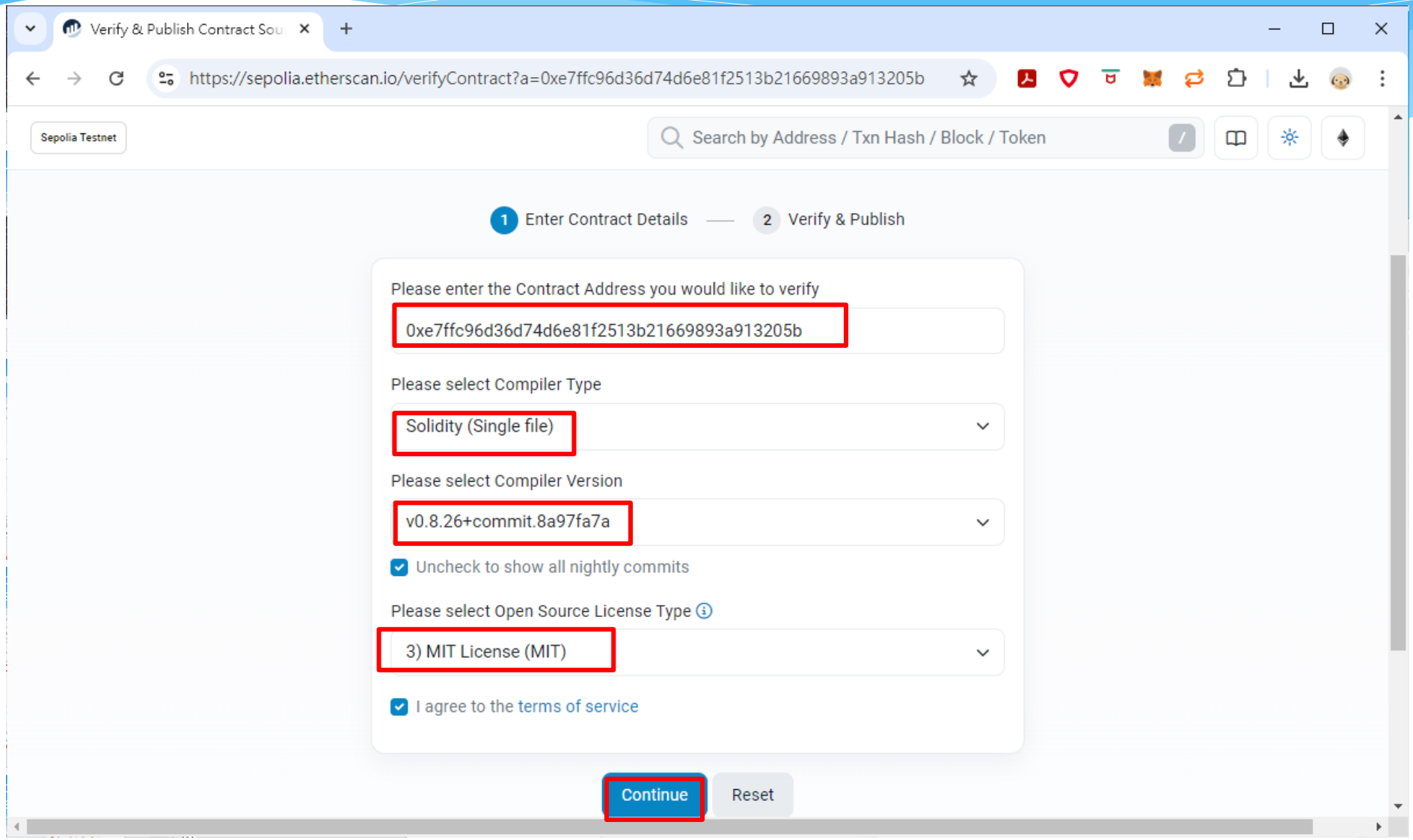


View & Publish



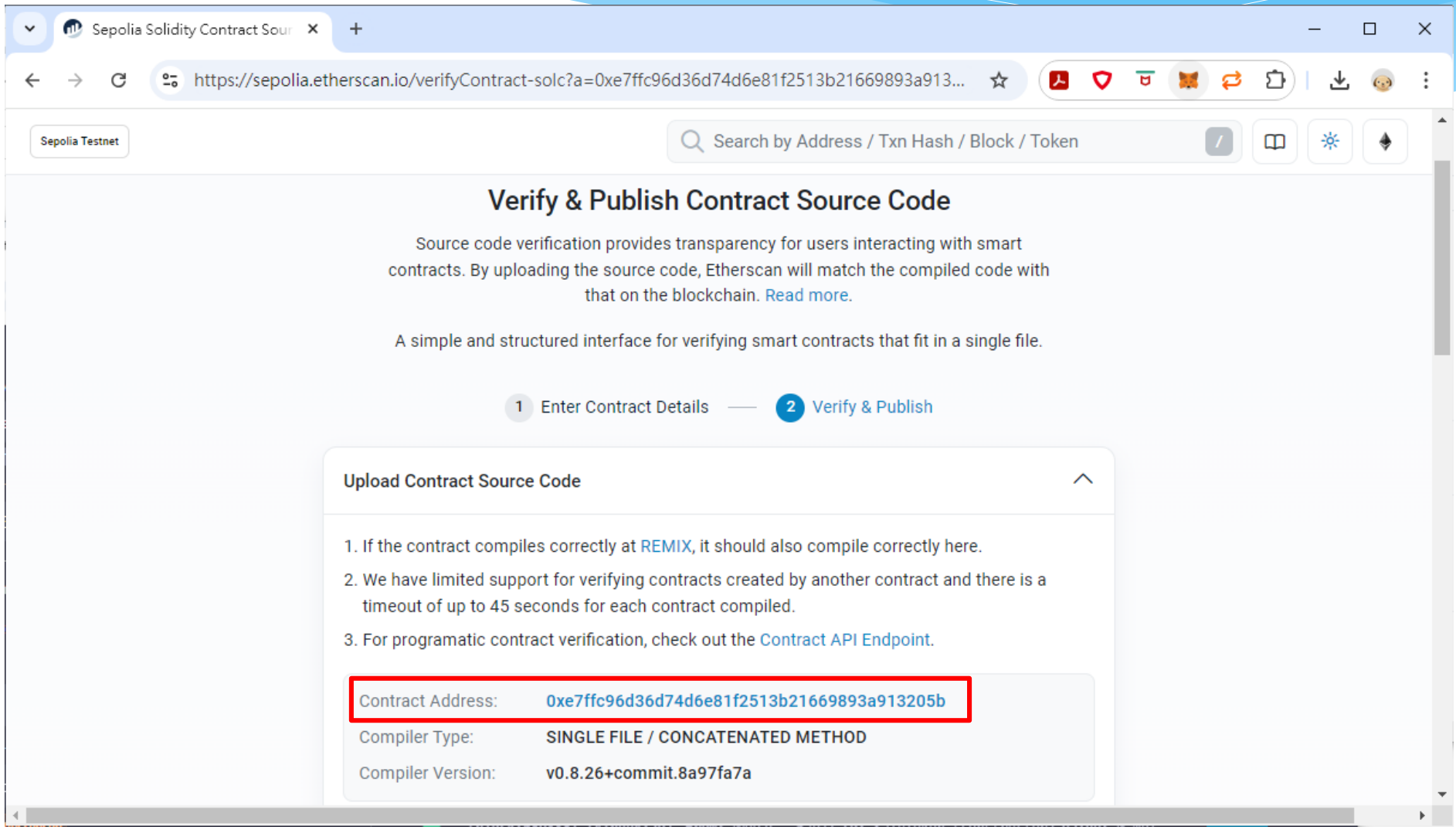
https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#code

View & Publish Contract Source Code

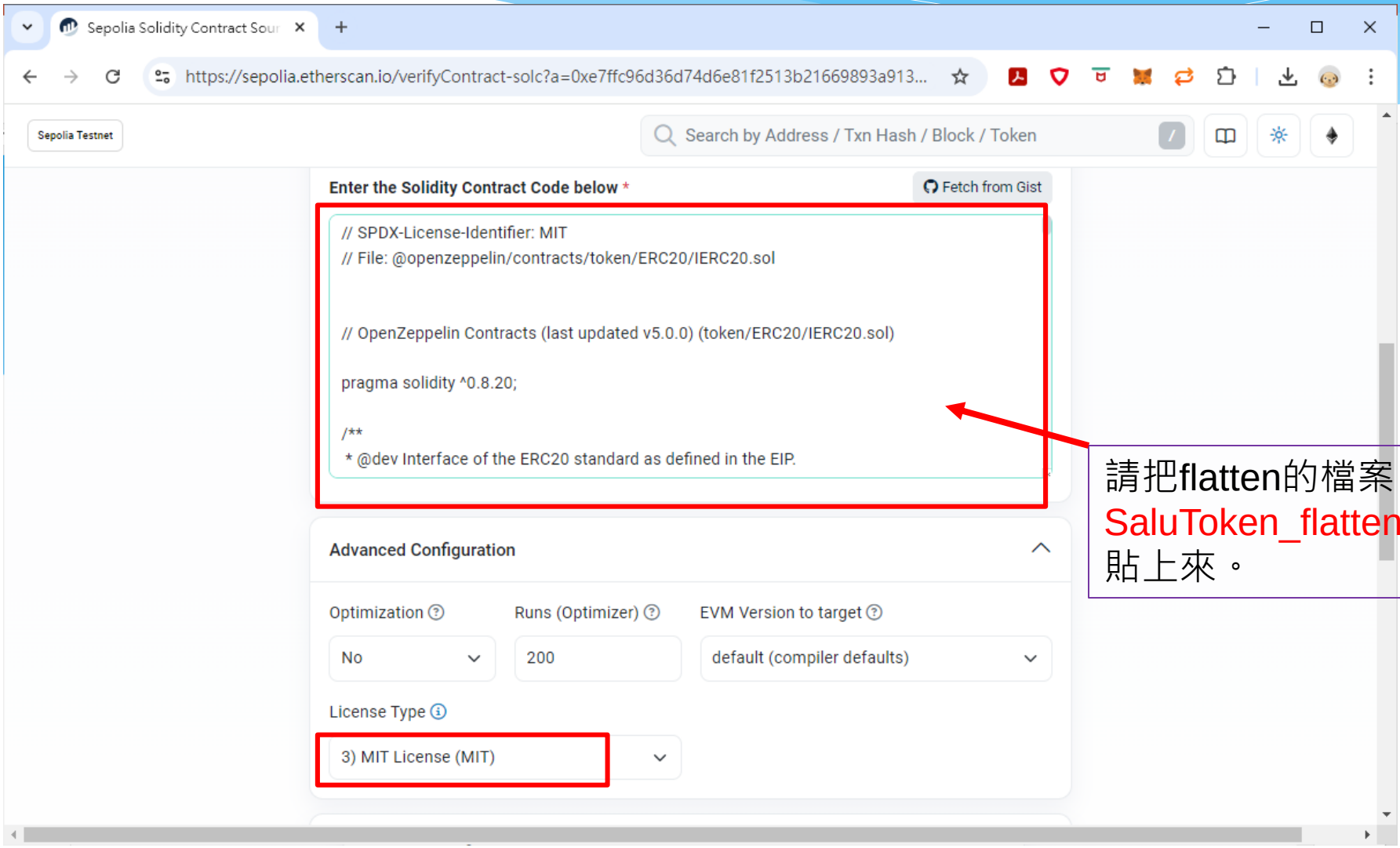


https://sepolia.etherscan.io/verifyContract?a=0xe7ffc96d36d74d6e81f2513b21669893a913205b

View & Publish Contract Source Code

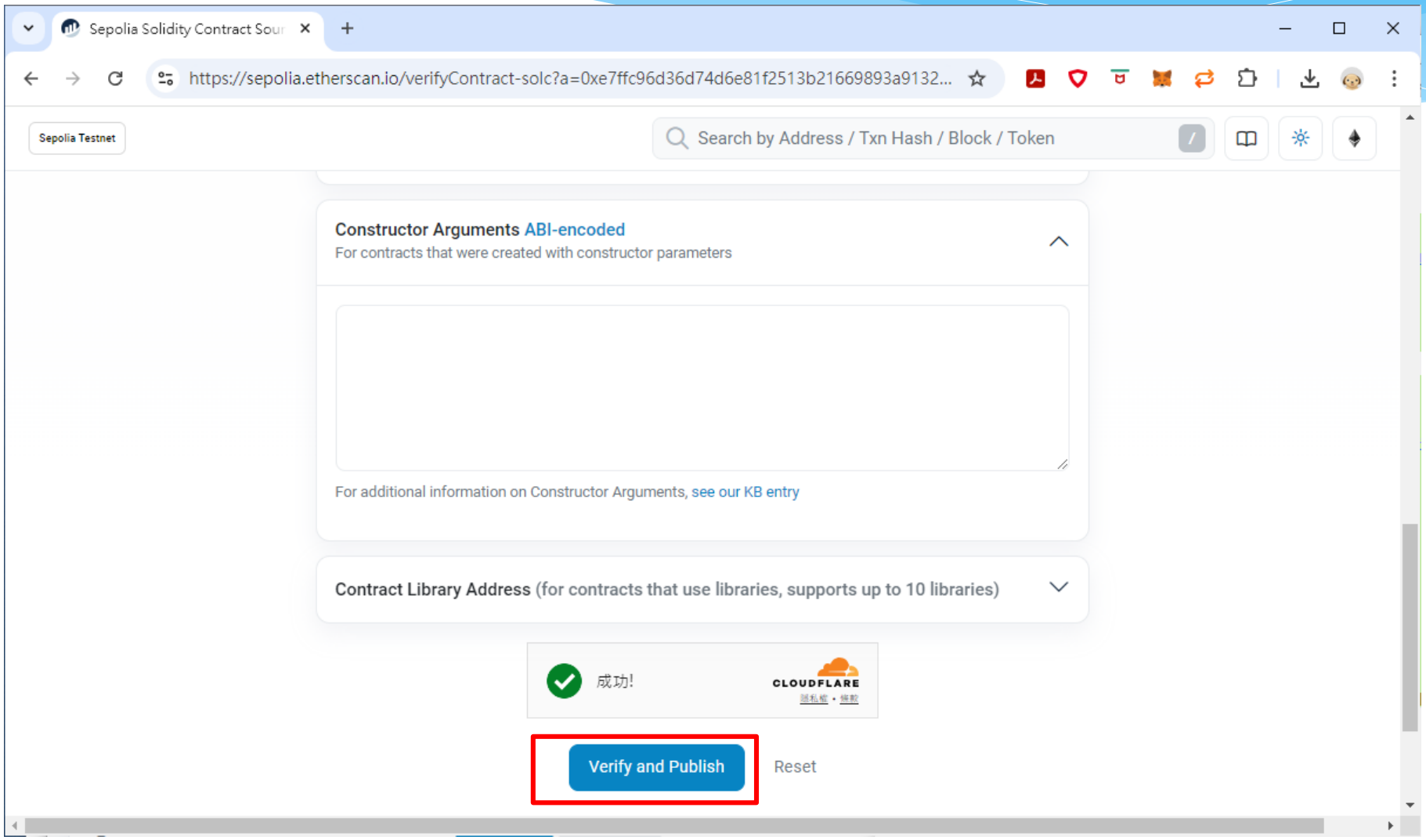


View & Publish Contract Source Code

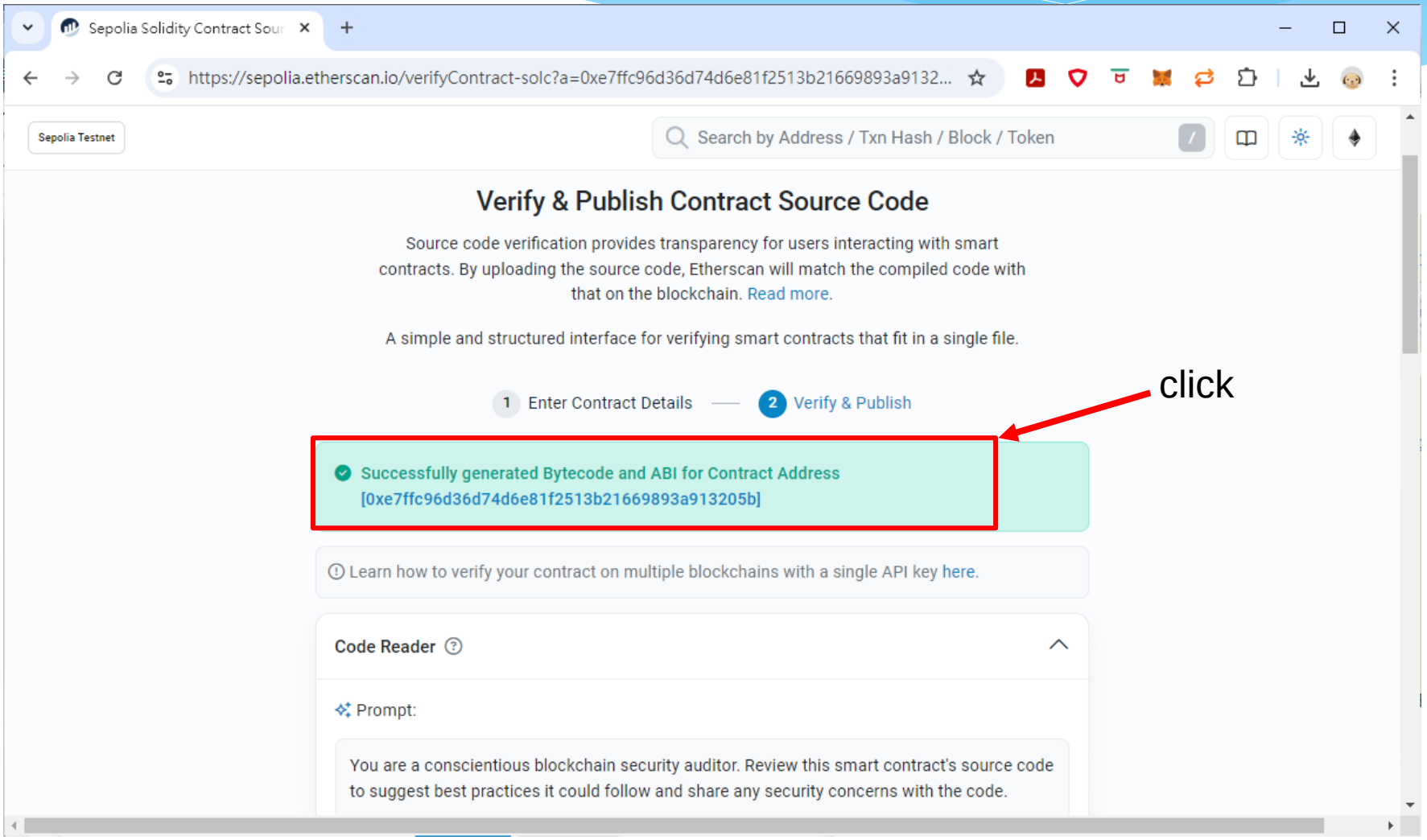


請把flatten的檔案 SaluToken_flattened.sol 貼上來。

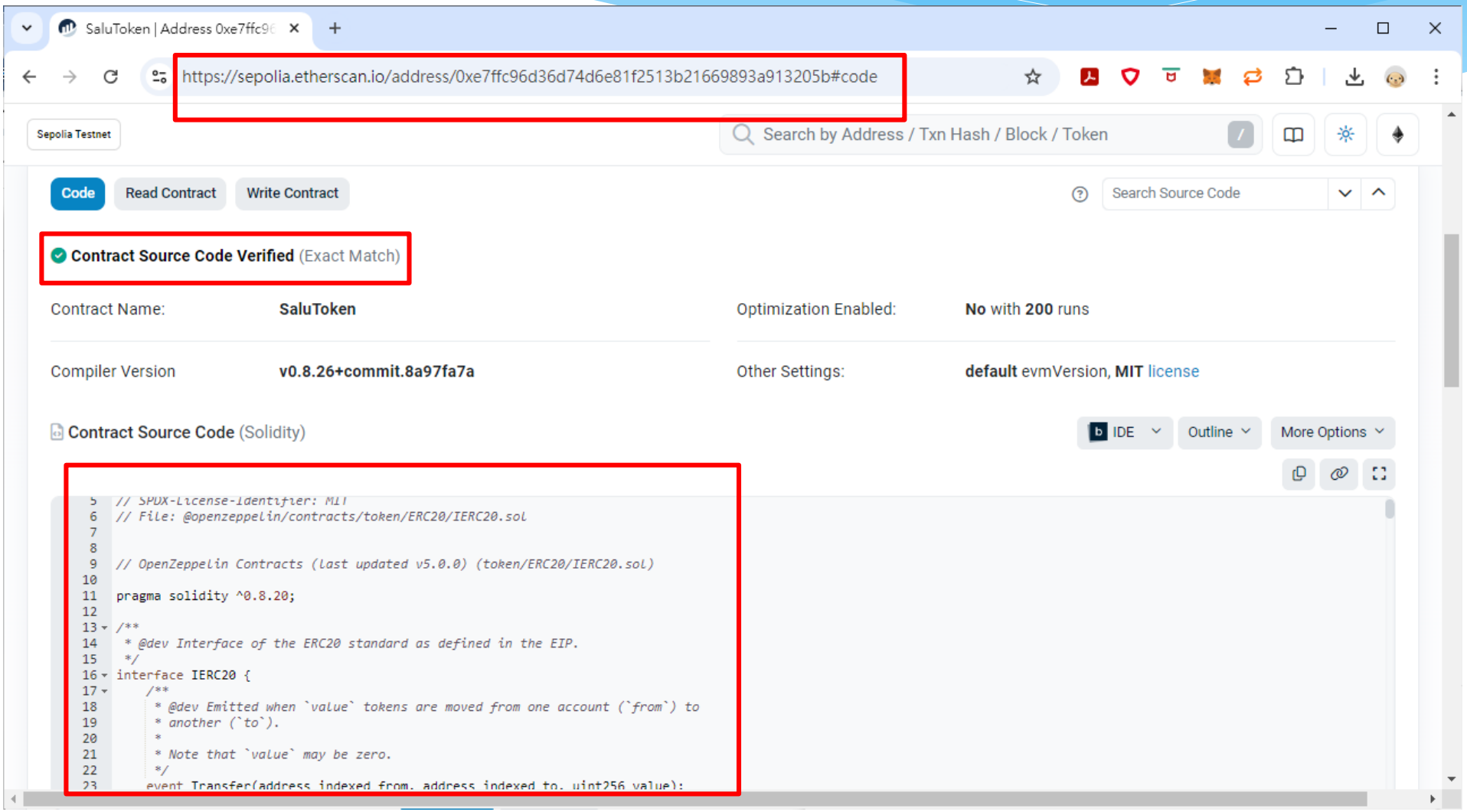
View & Publish Contract Source Code



View & Publish Contract Source Code

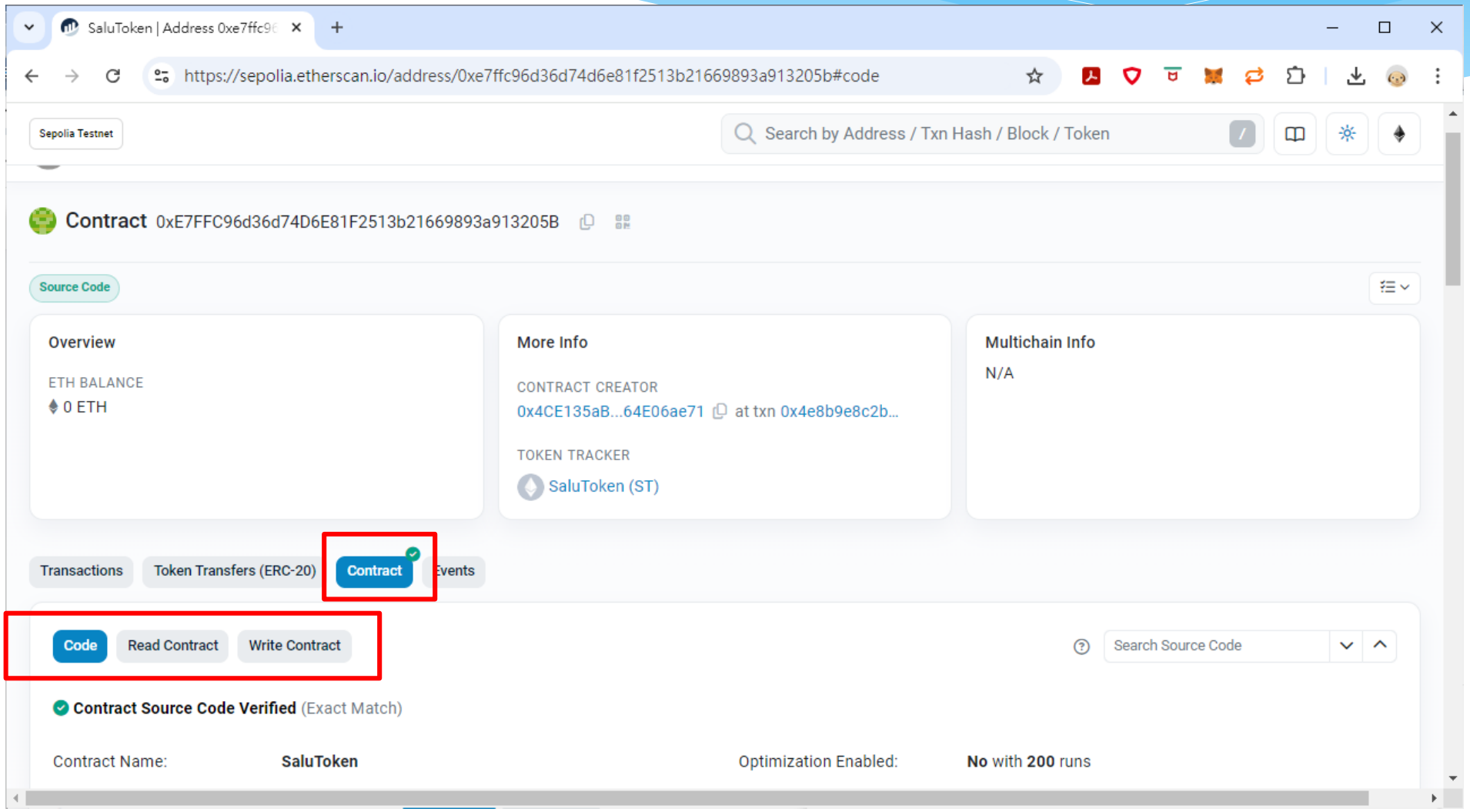


View Contract Source Code



https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#code

View & Publish Contract Source Code

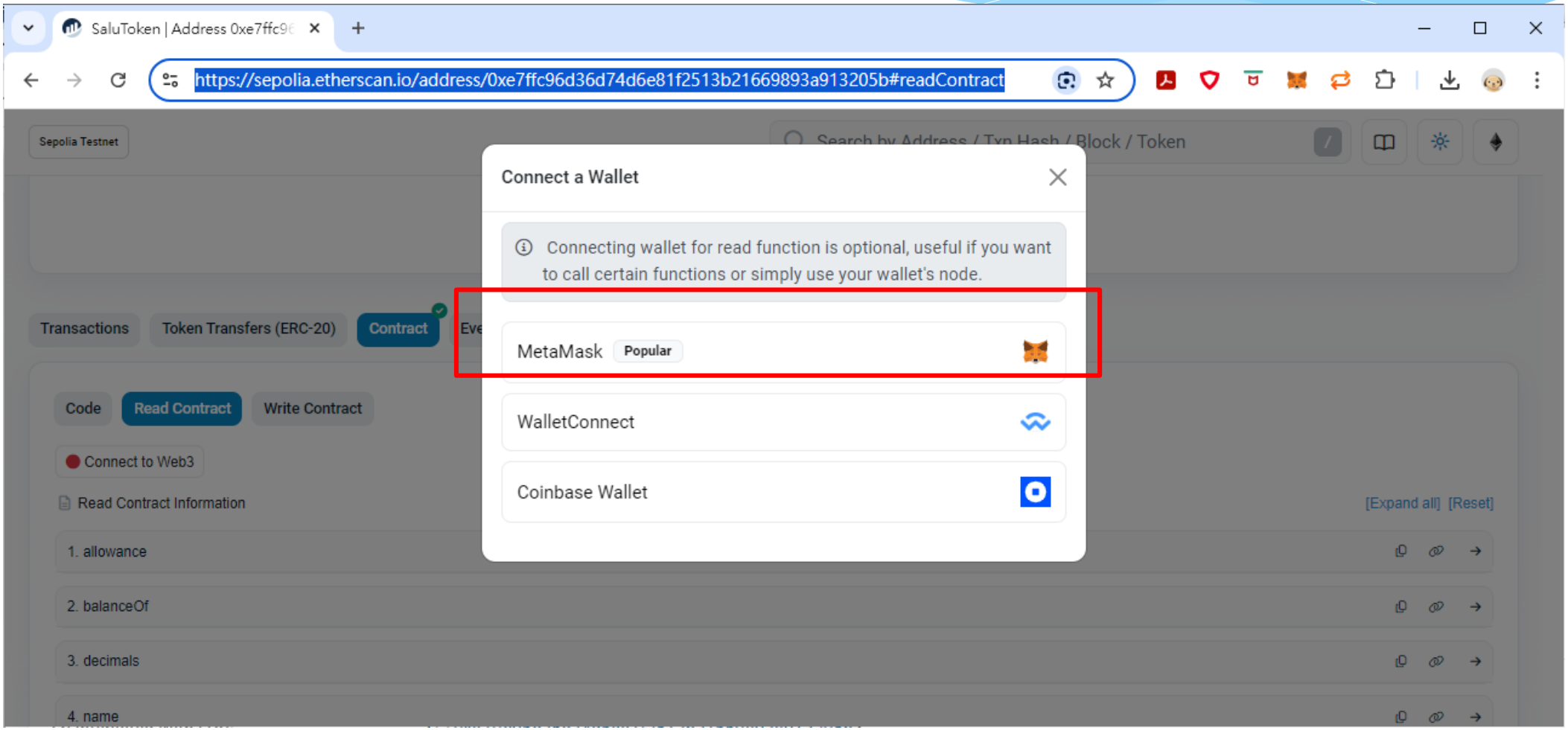


https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#code

Connect to Web3

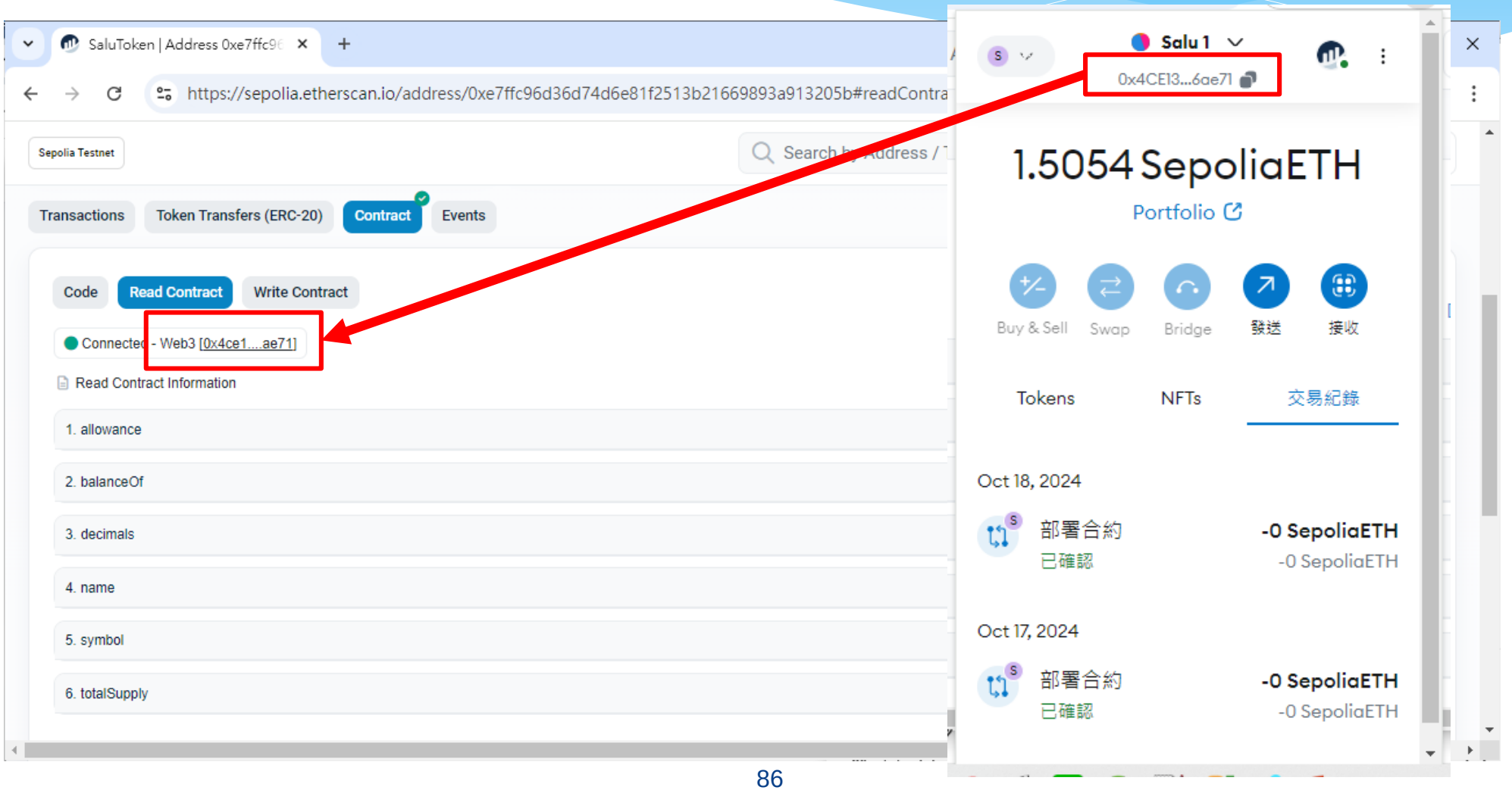
https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#readContract

Connect a Wallet-MetaMask



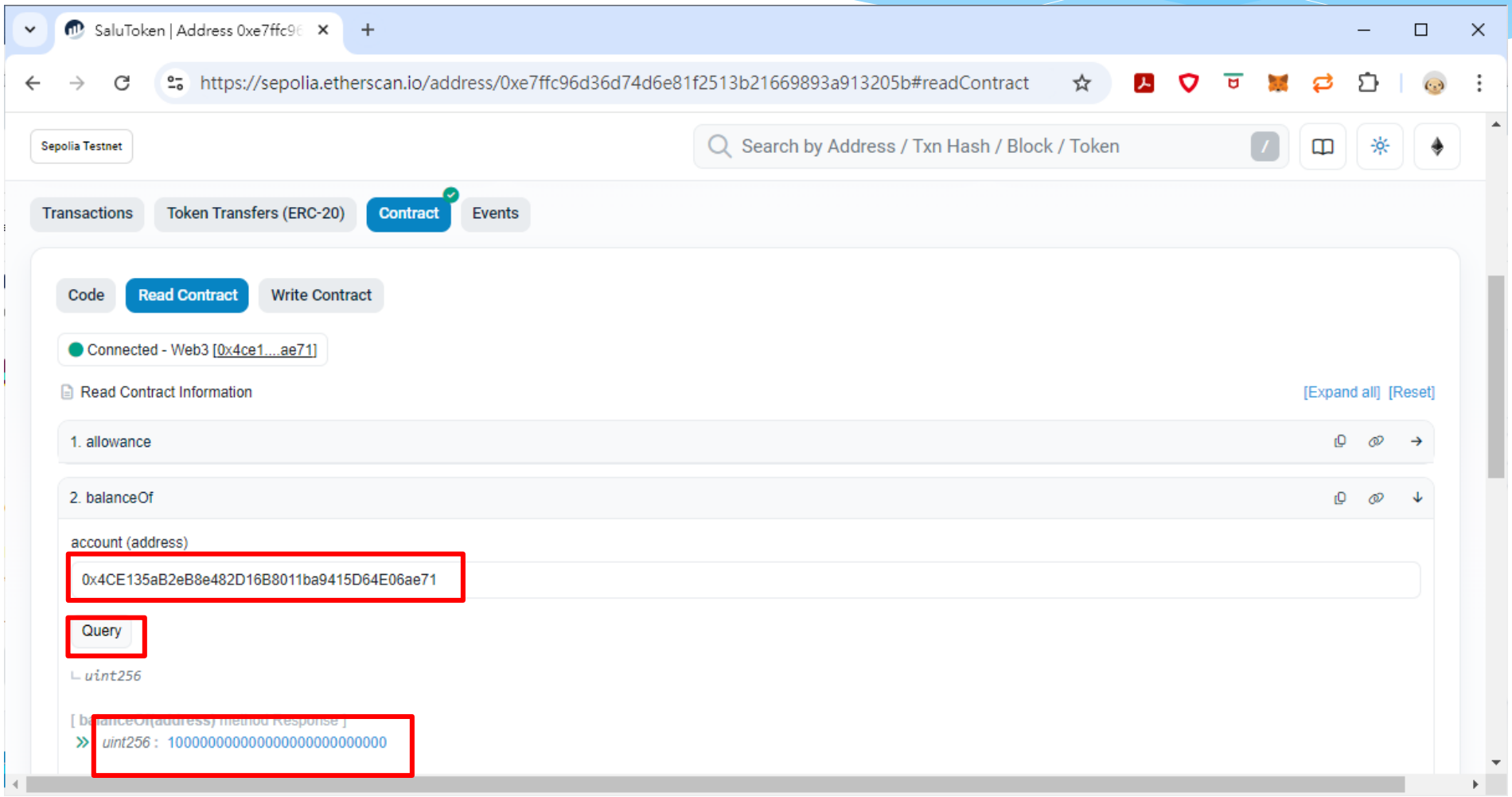
https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#readContract

Connected Web3 using account



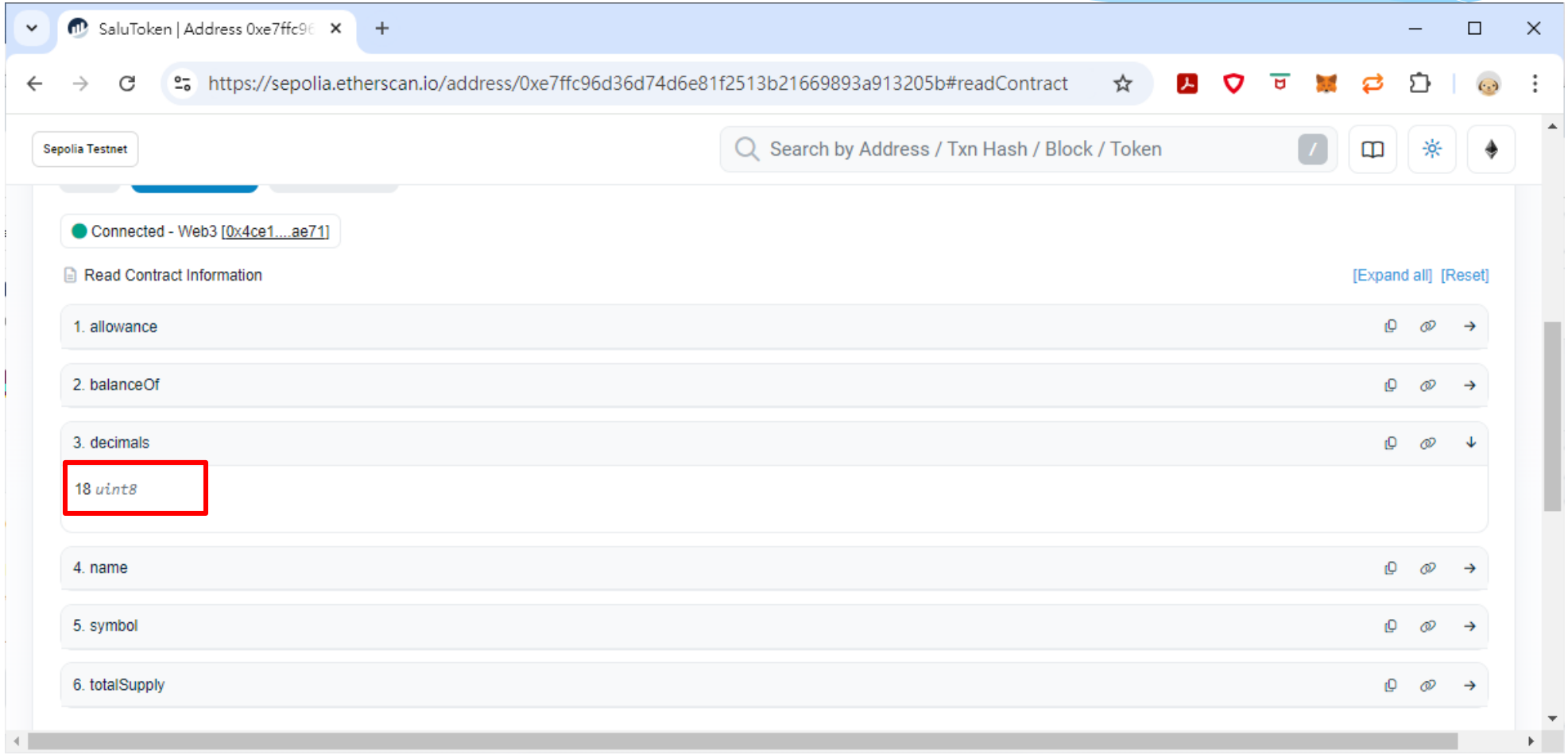
https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#readContract

Read Contract - balanceOf



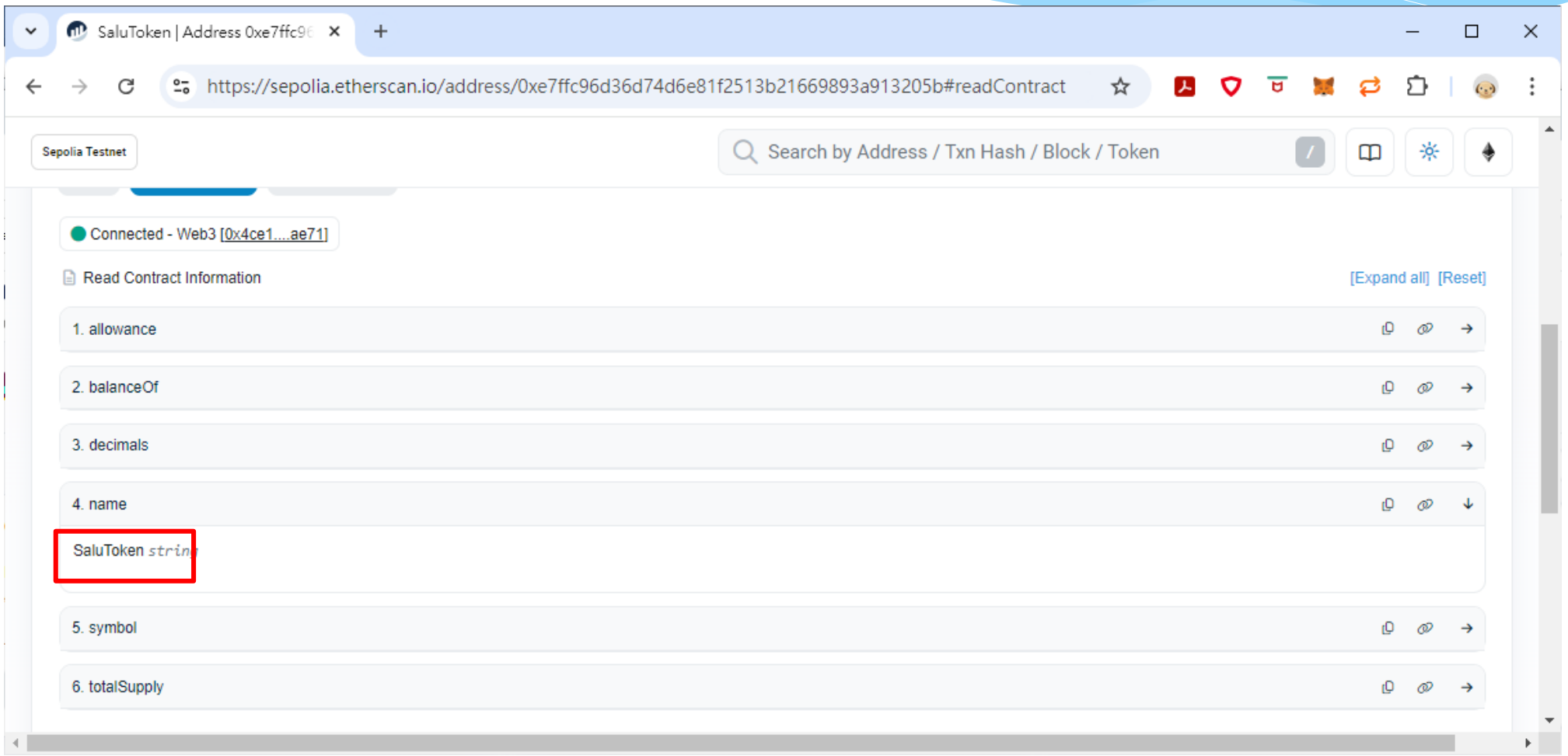
https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#readContract

Read Contract-decimals



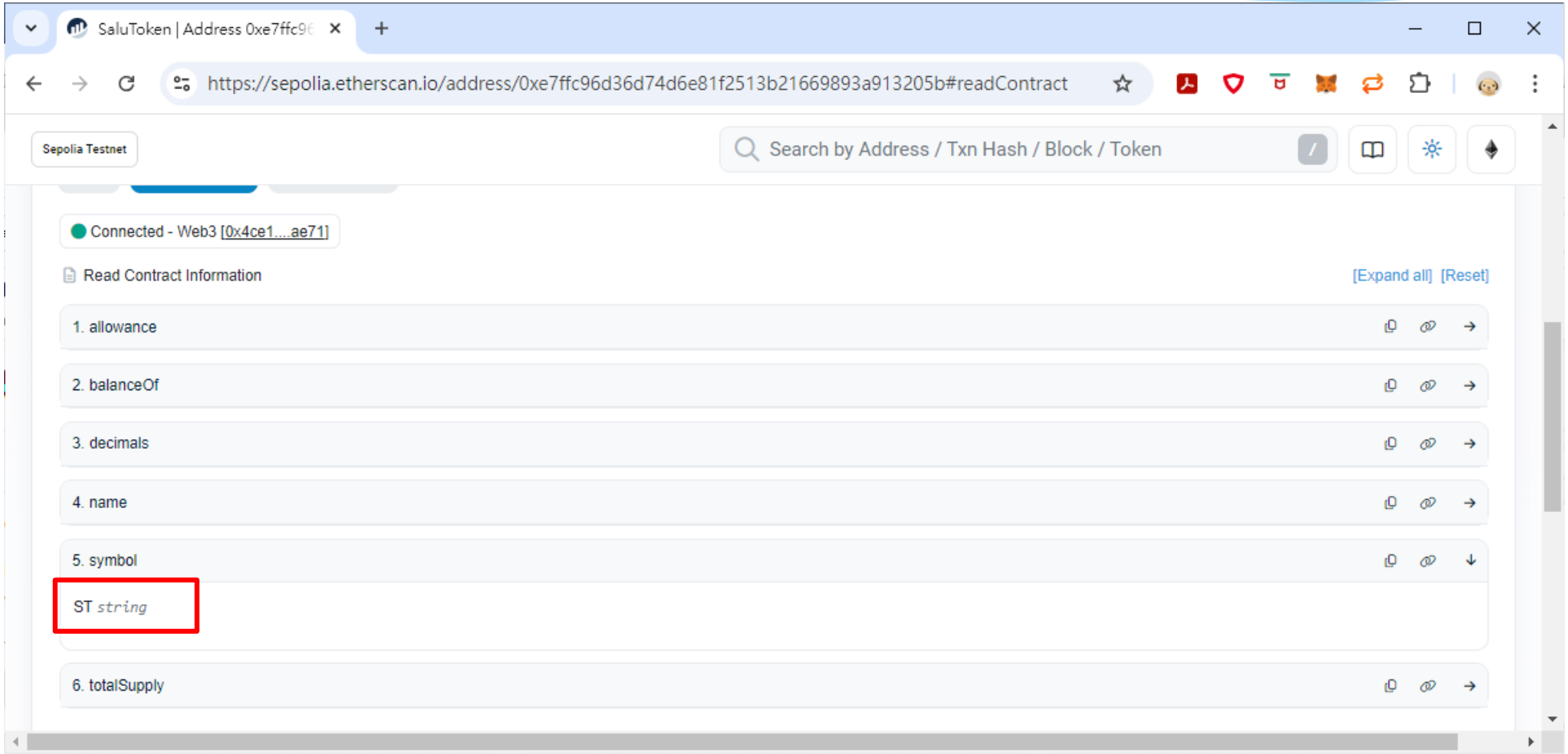
https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#readContract

Read Contract-name



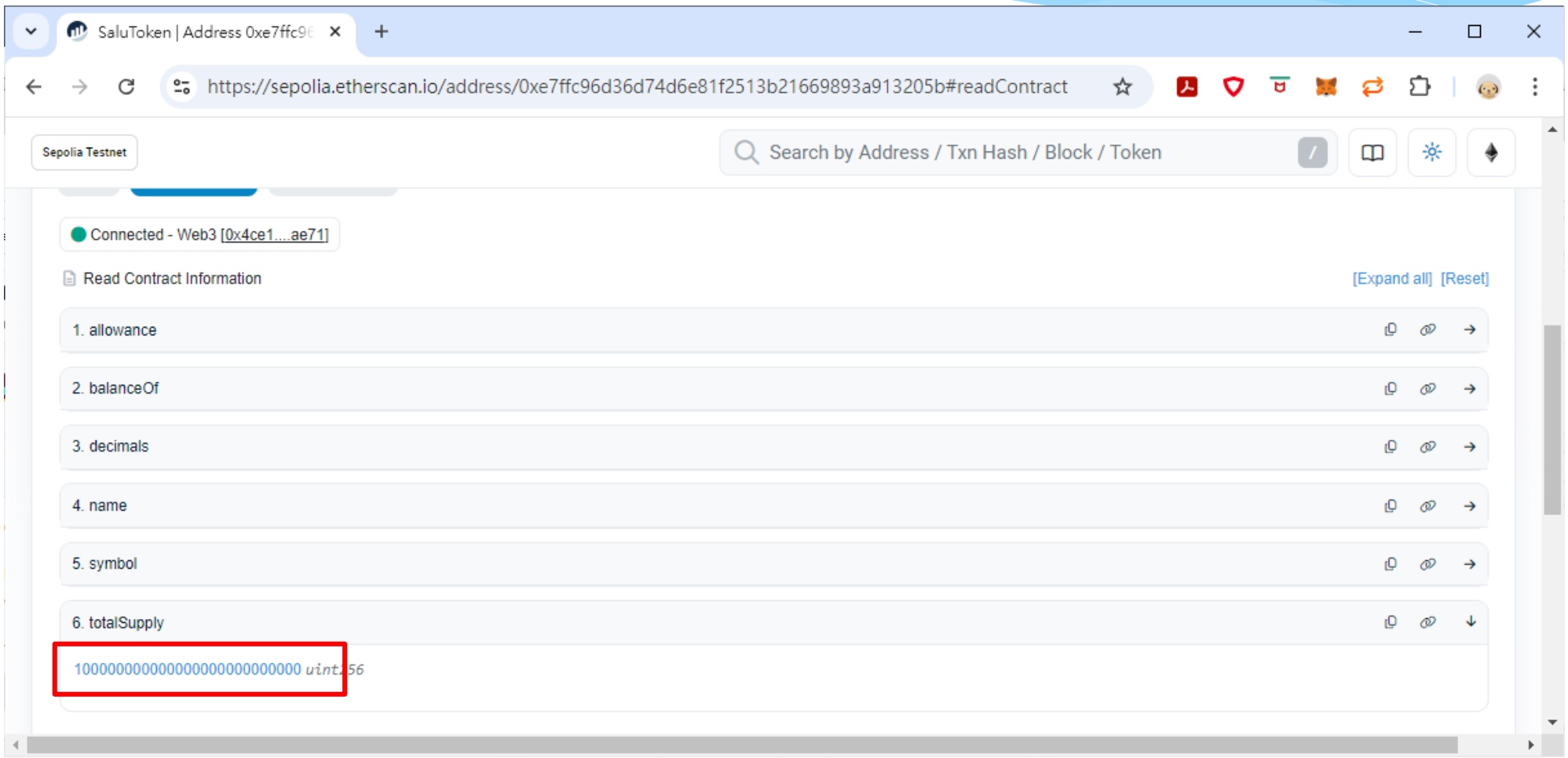
https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#readContract

Read Contract-symbol



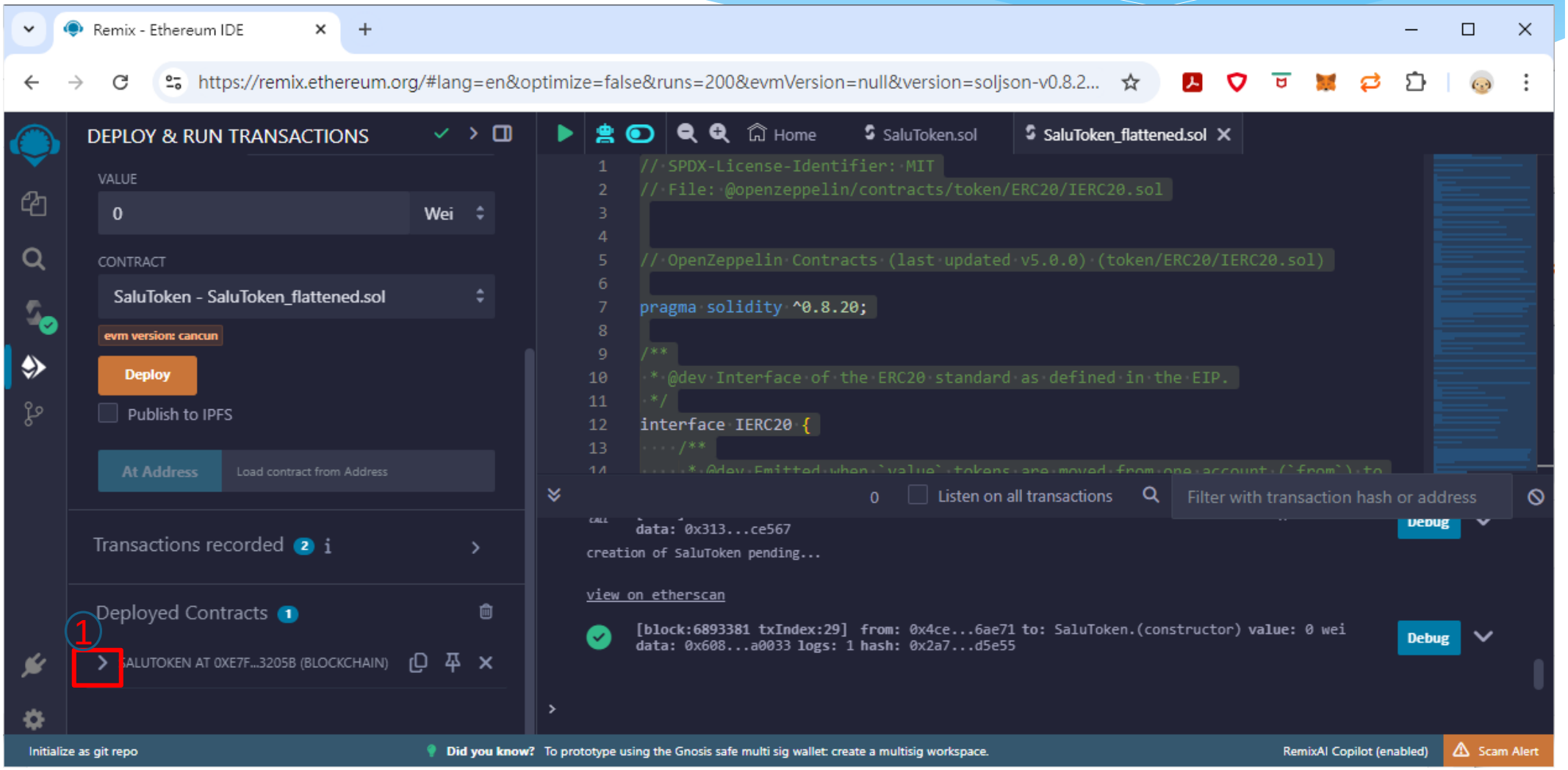
https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#readContract

Read Contract-totalSupply

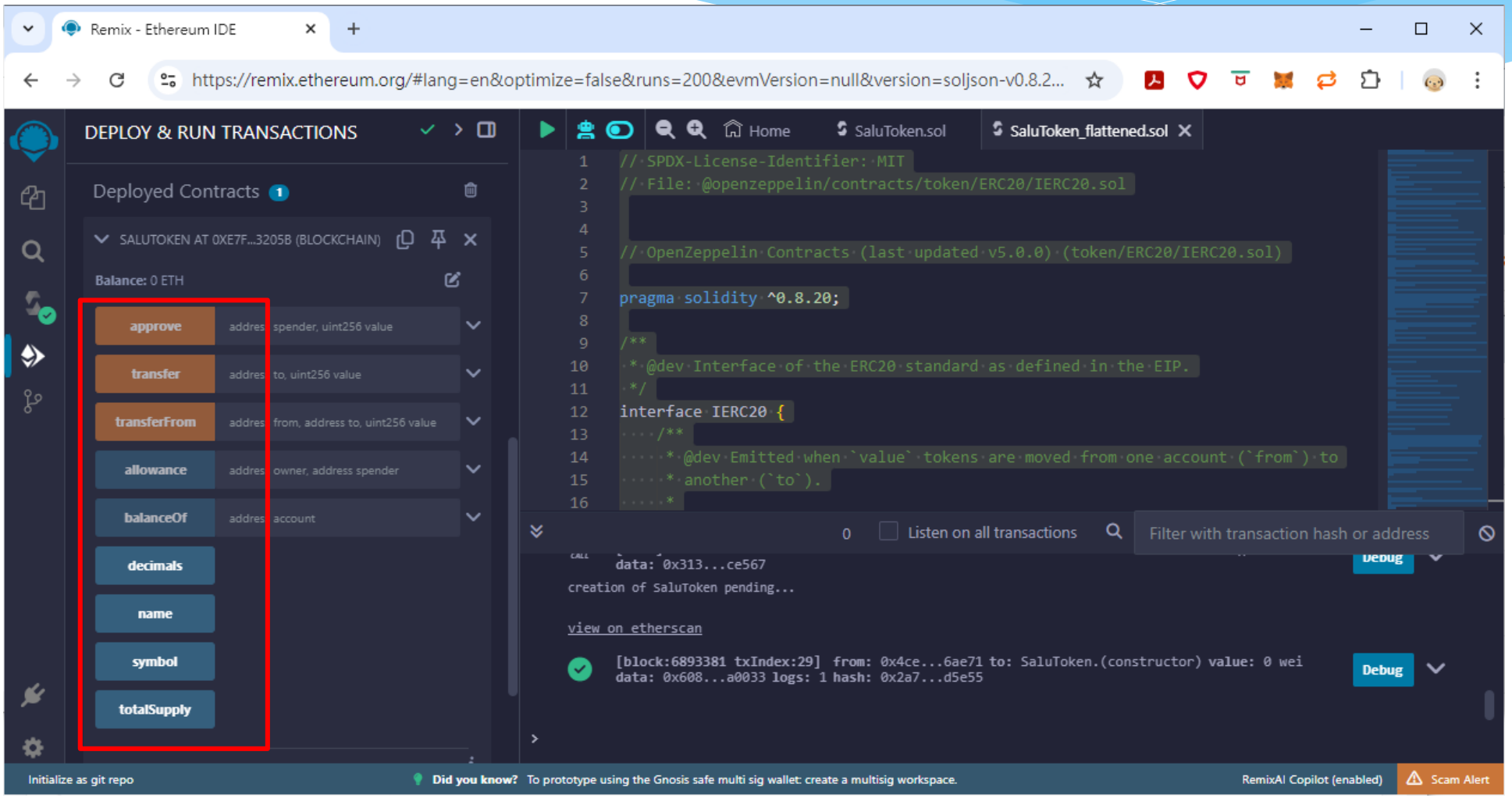


https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#readContract

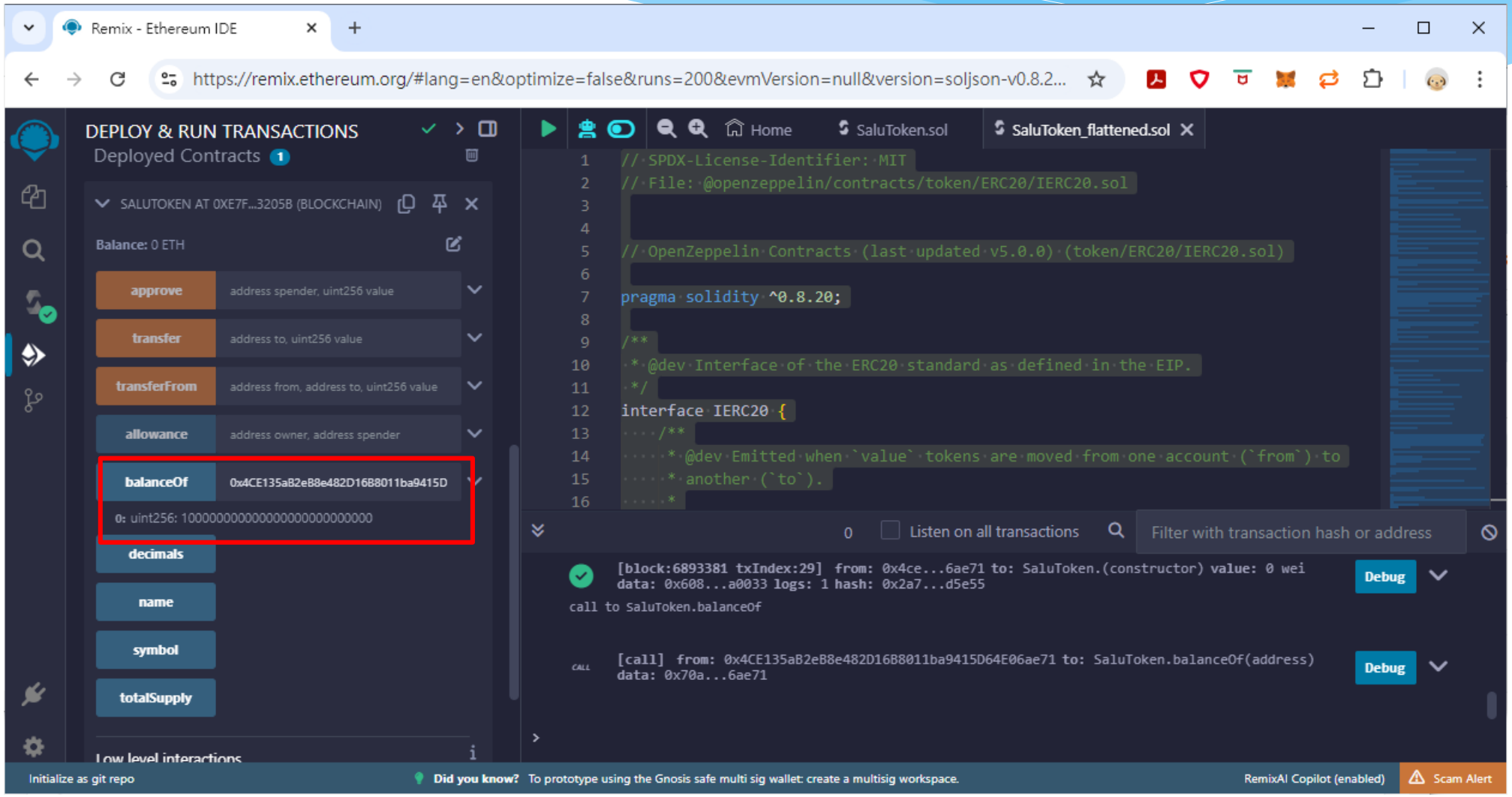
List Contract using Remix



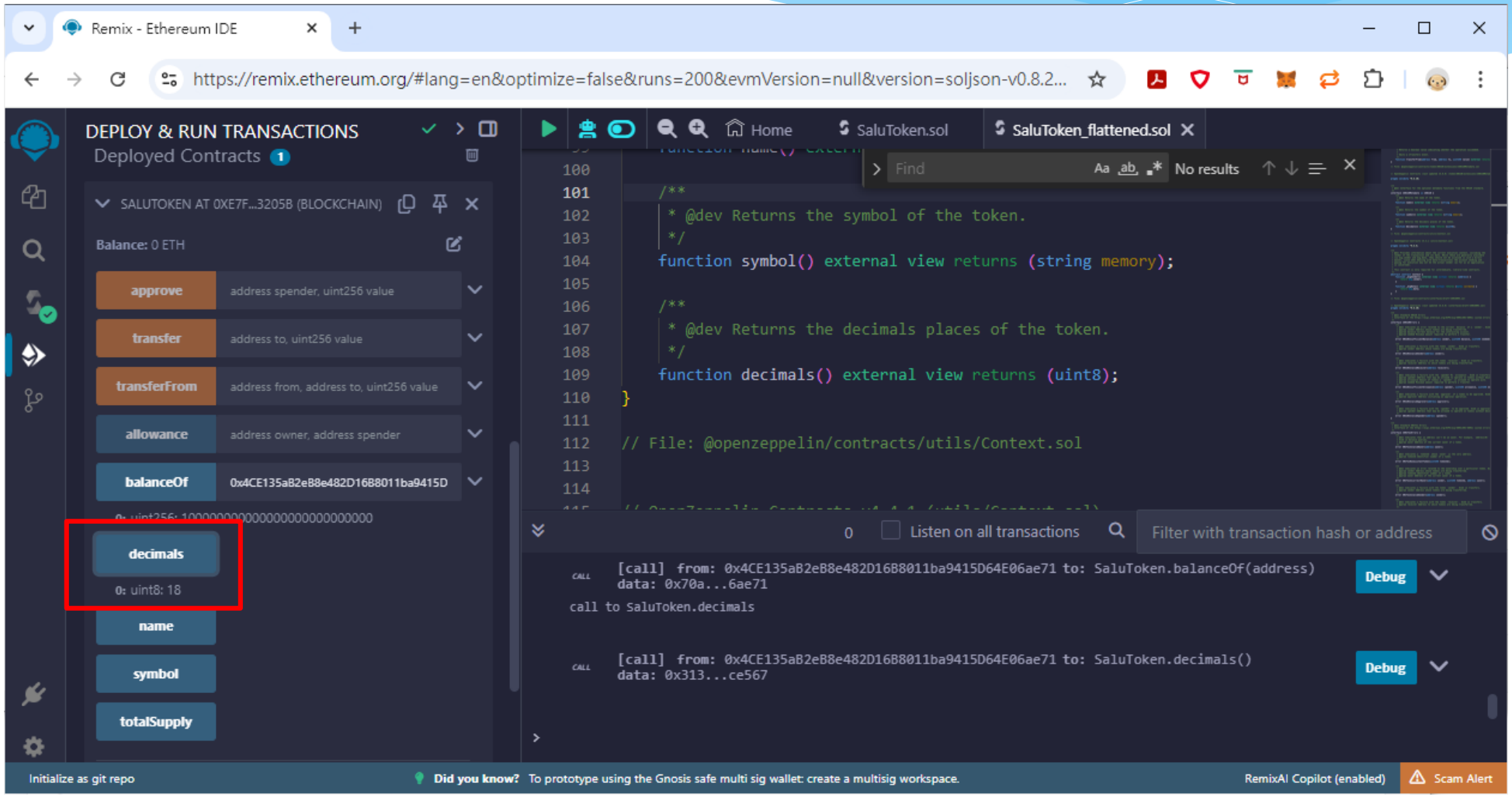
List Contract using Remix



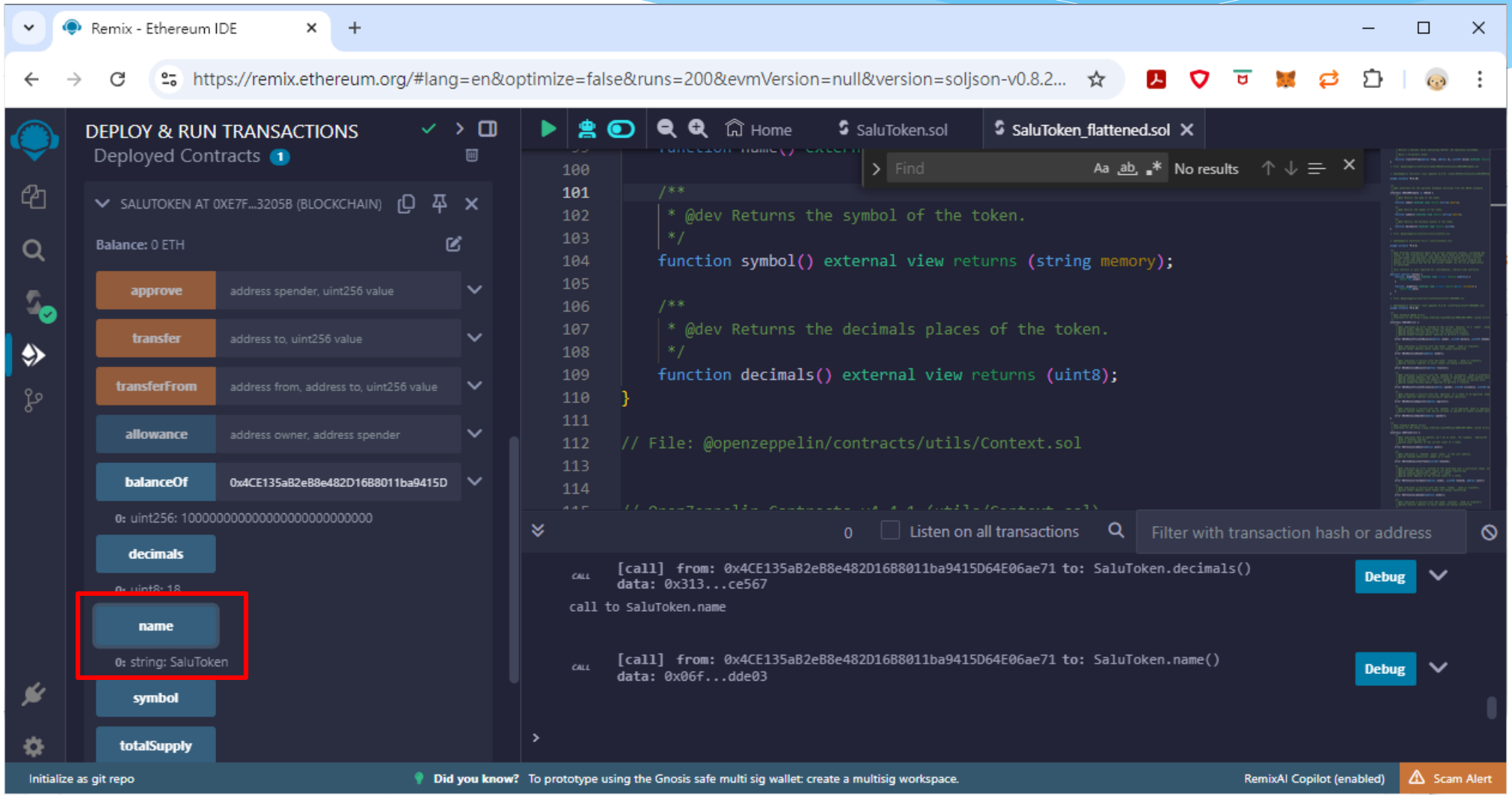
balanceOf function



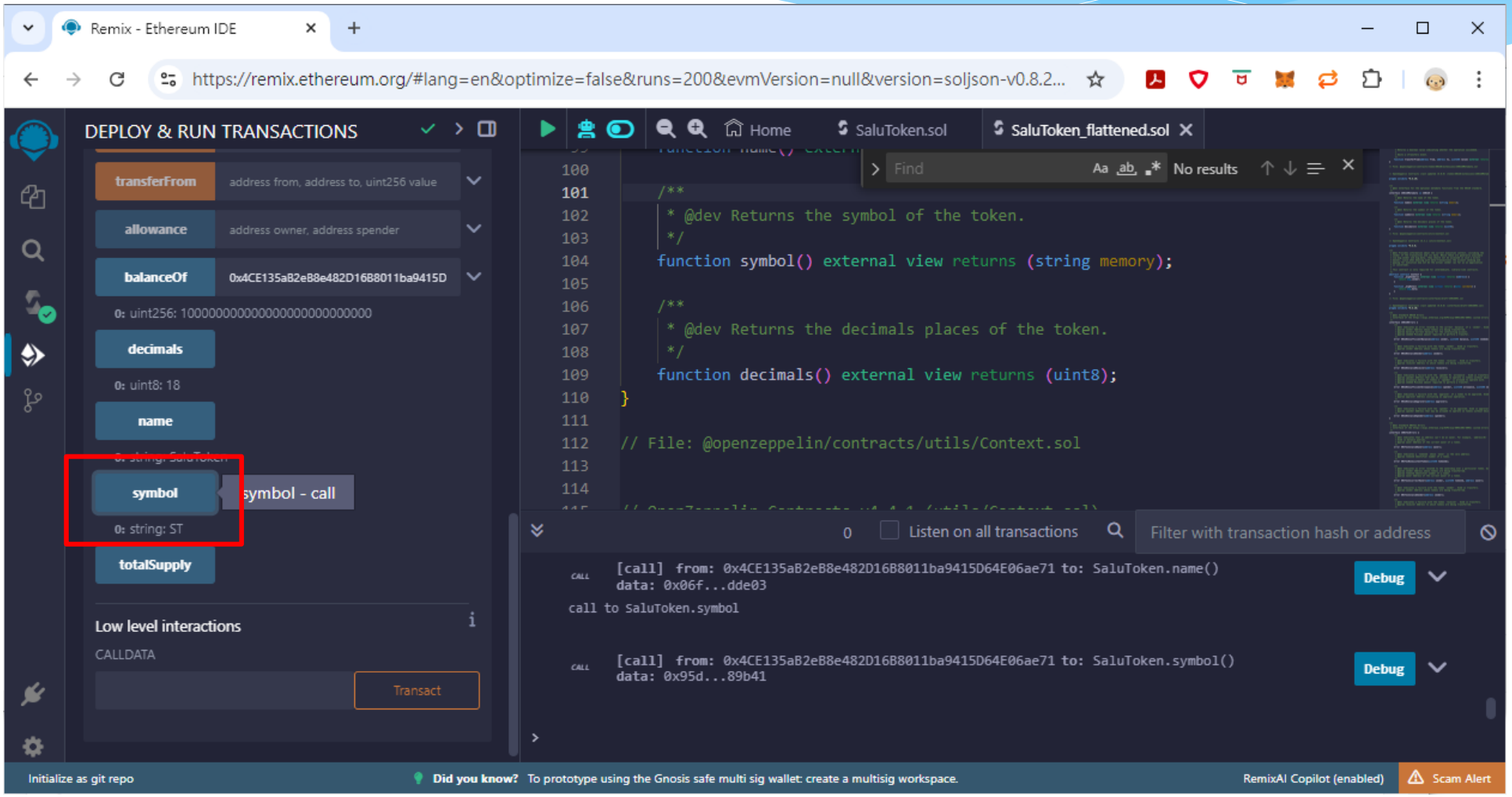
decimals function



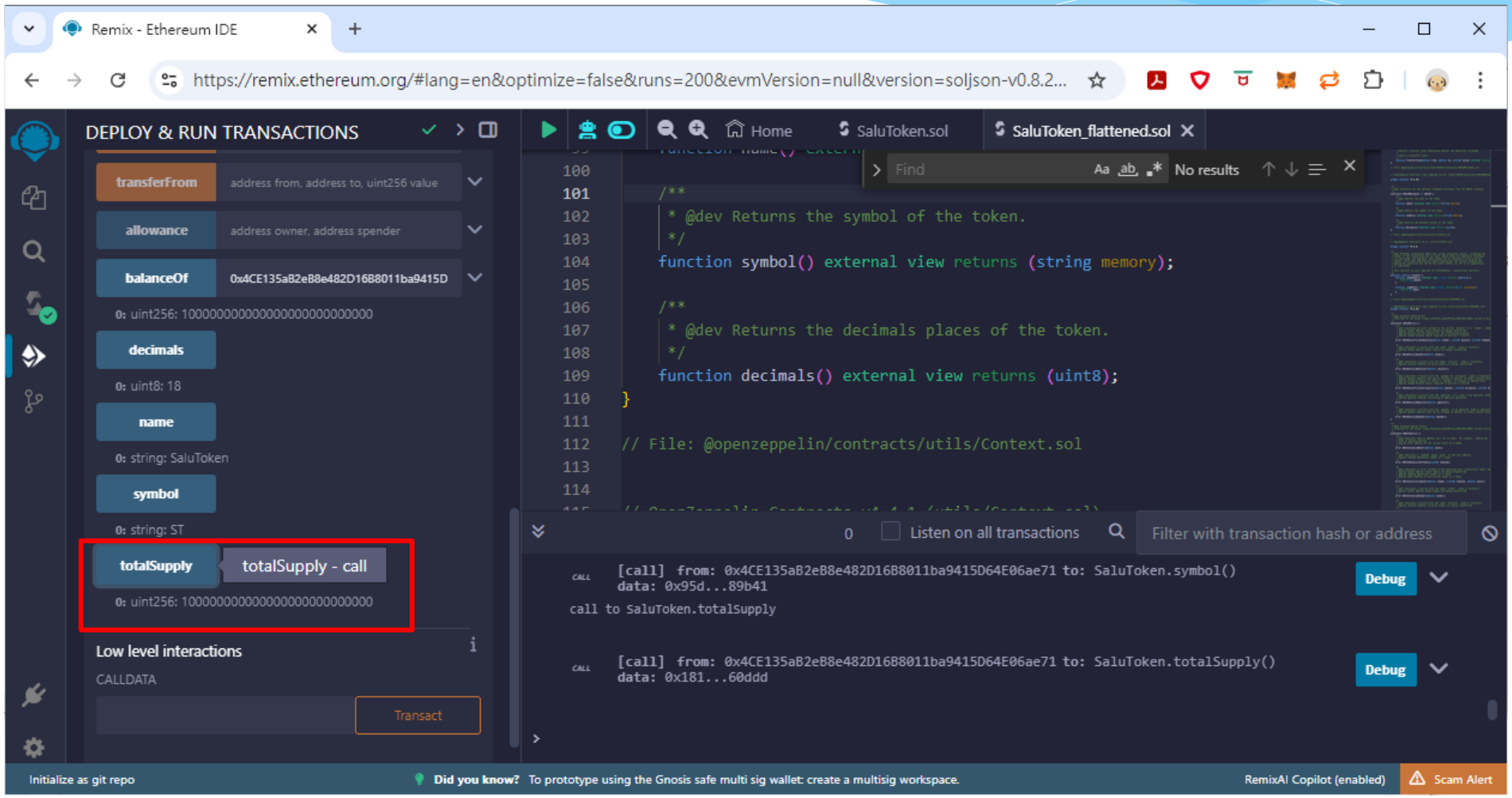
name function



symbol function

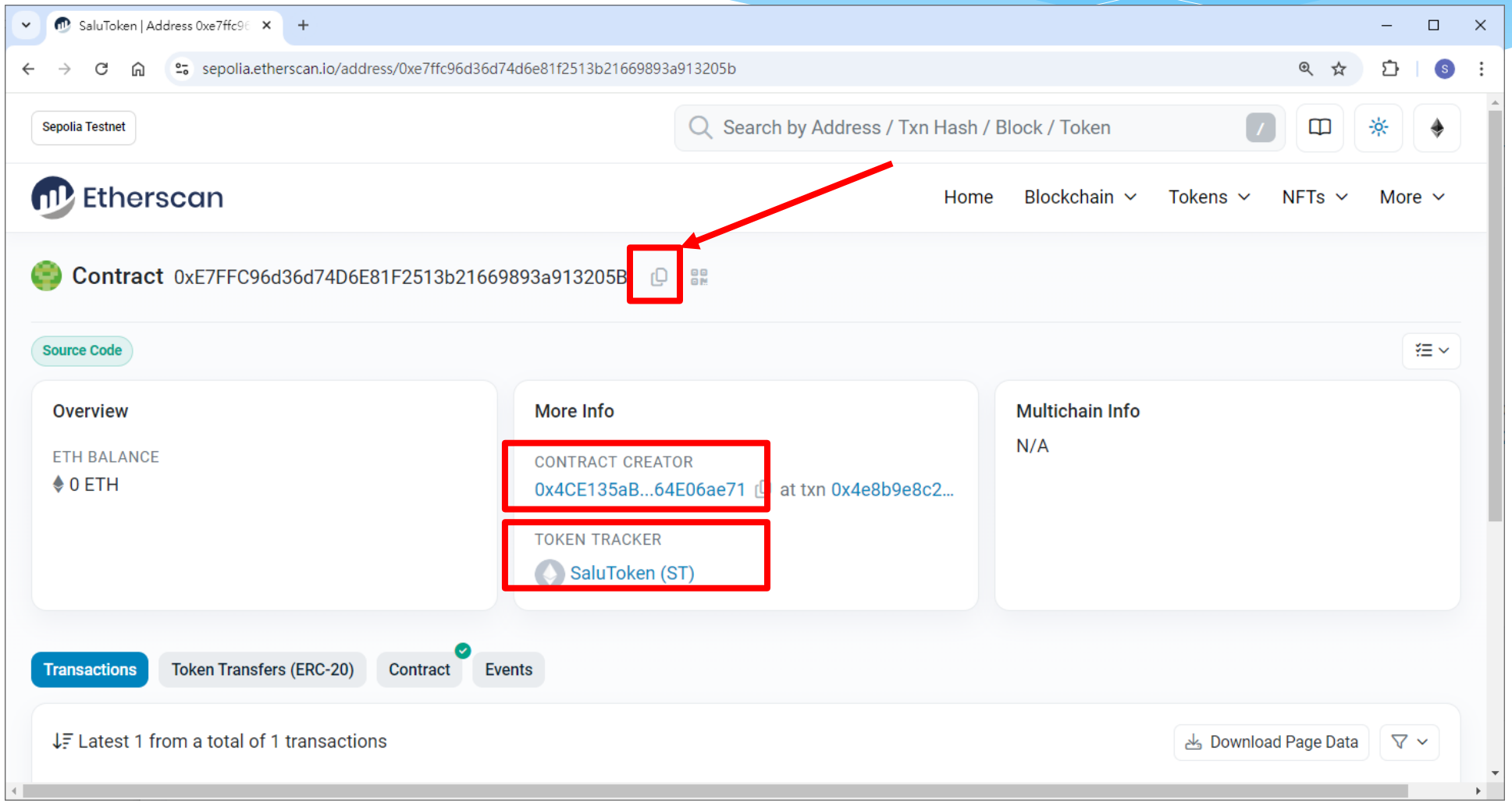


totalSupply function



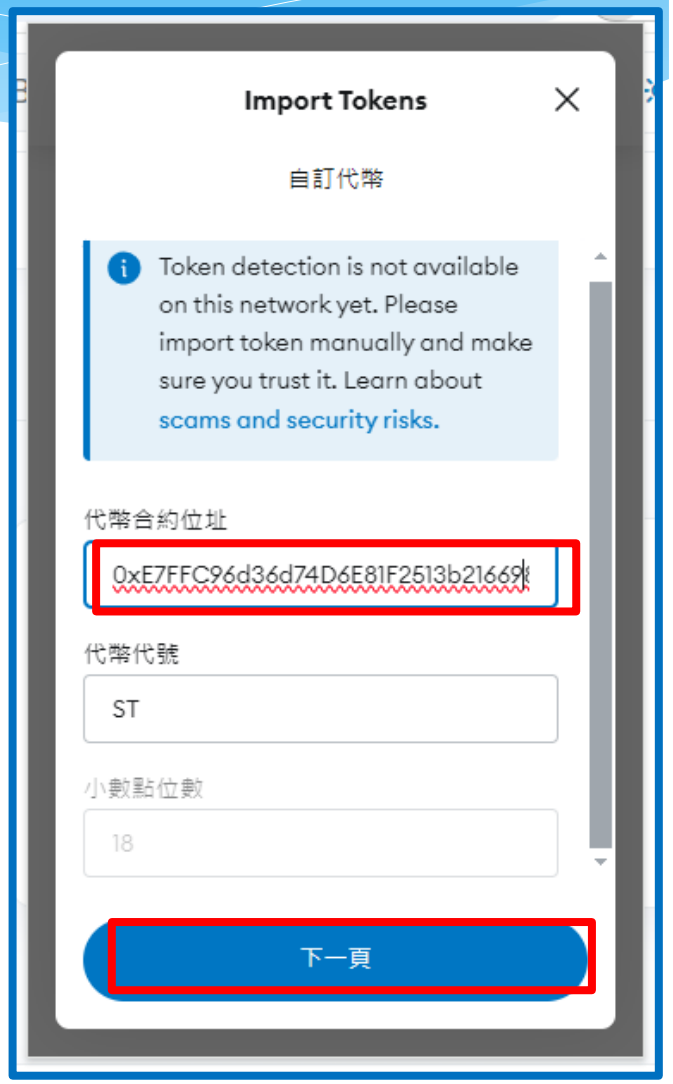
Import Tokens

Copy Contract address

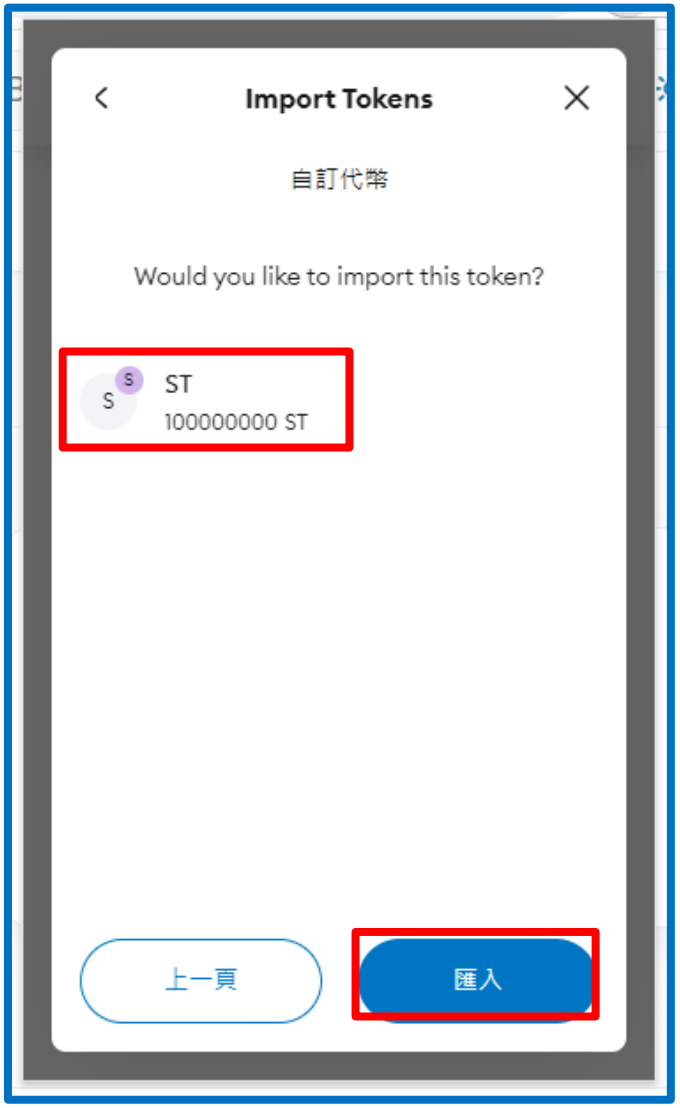


https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b

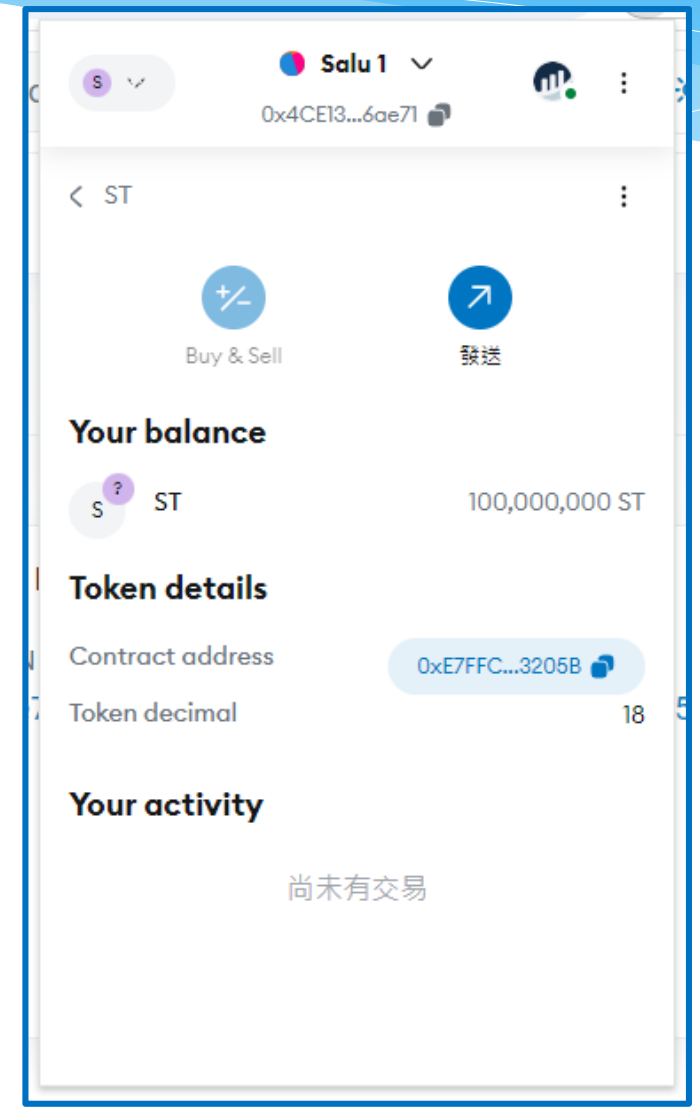
Import Tokens



Import Tokens

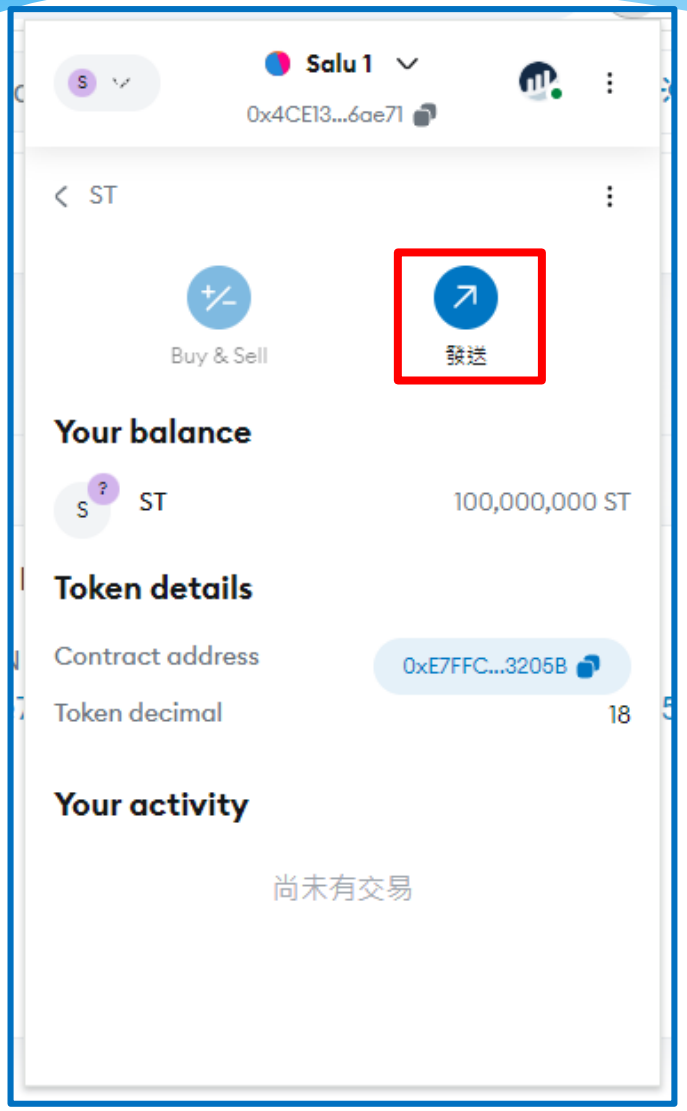


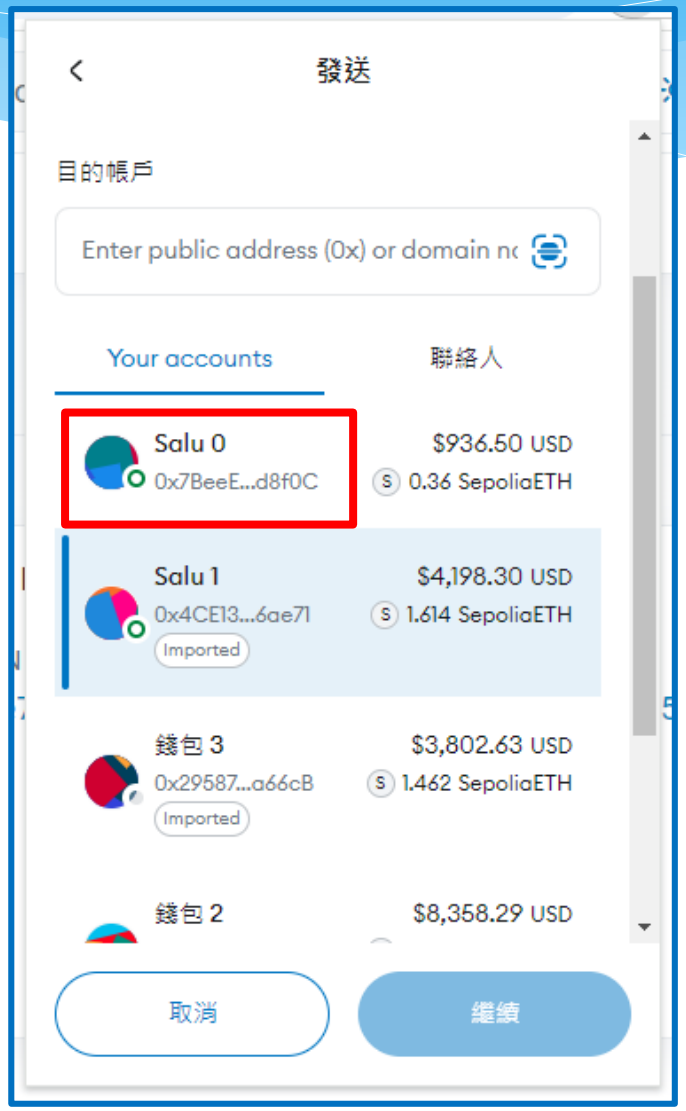
Token details



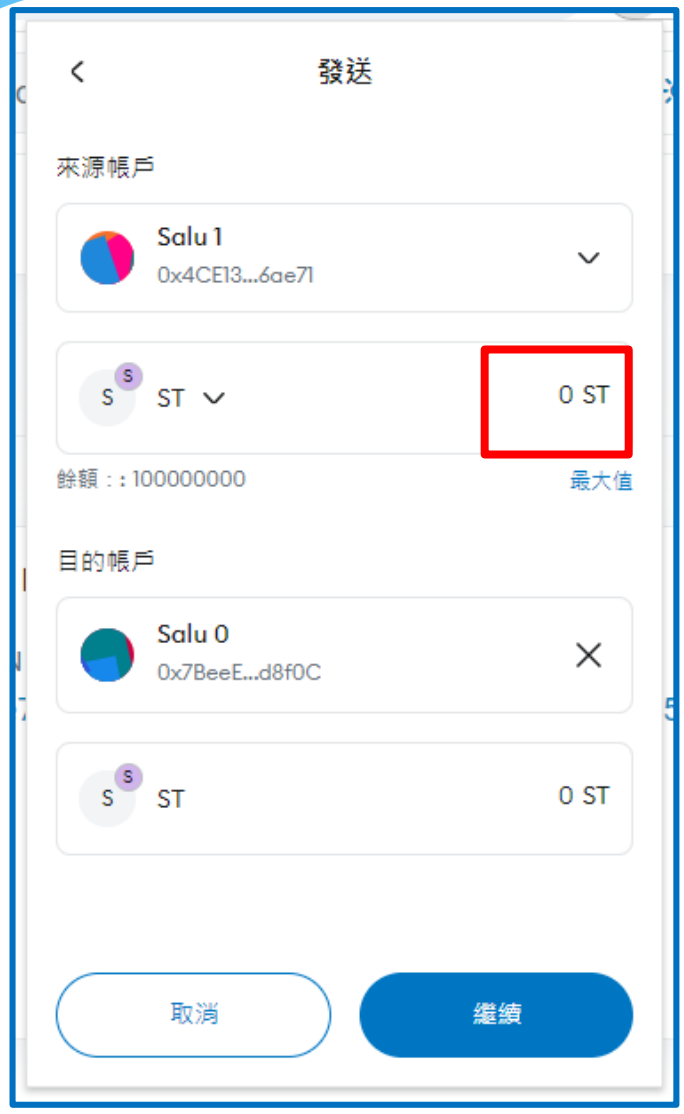
Transfer Token

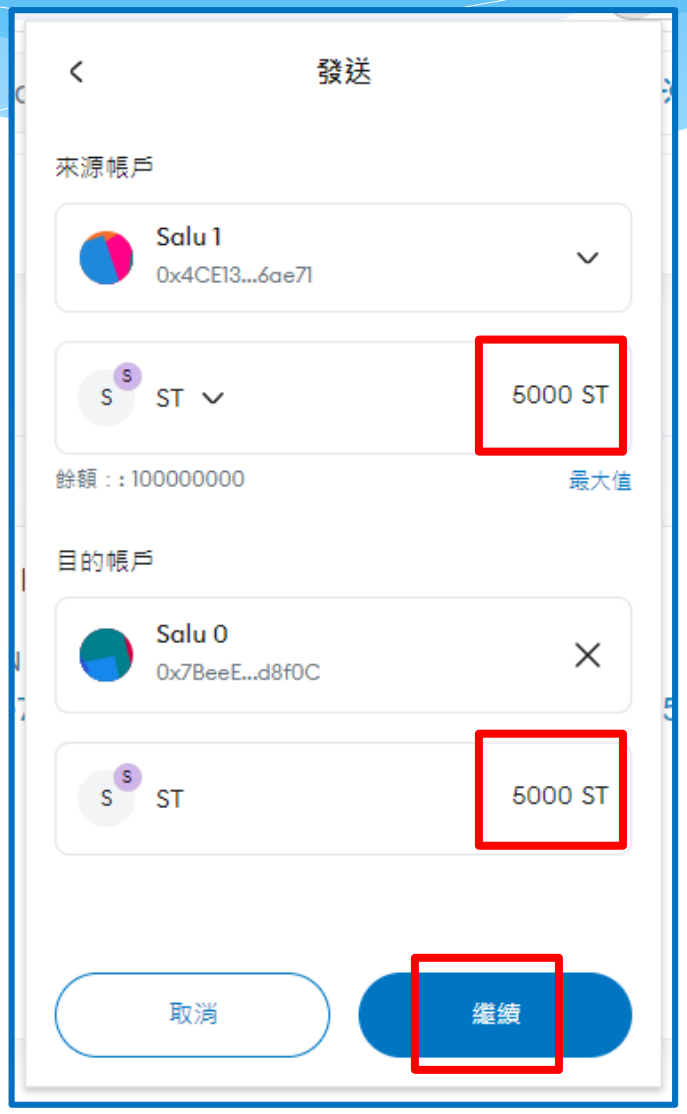
Transfer Token



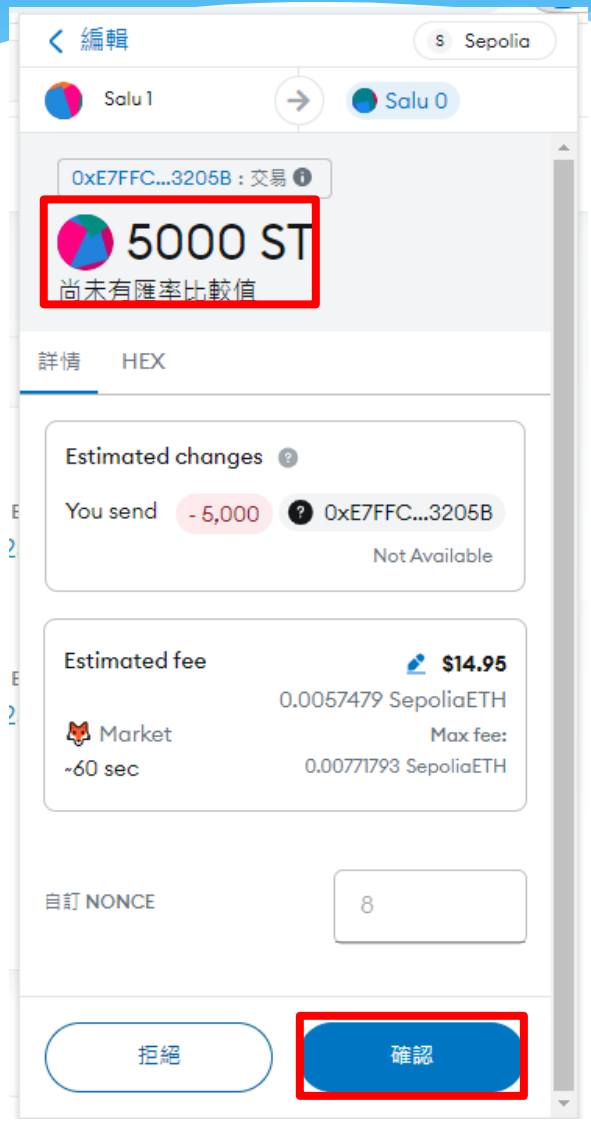


Transfer Token



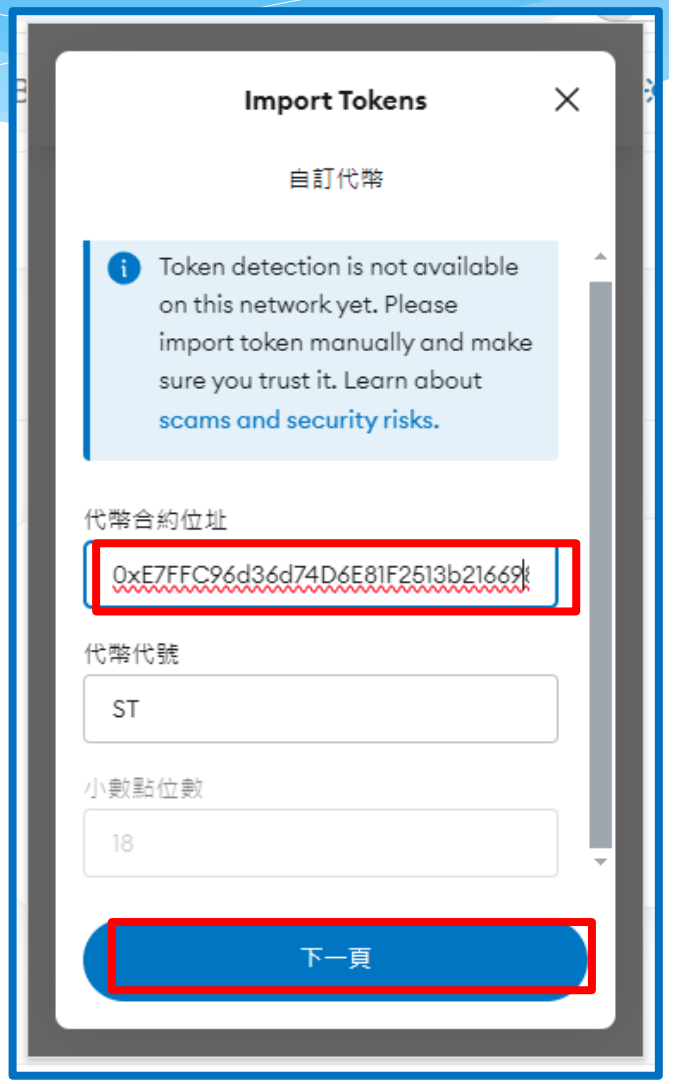


Transfer Token

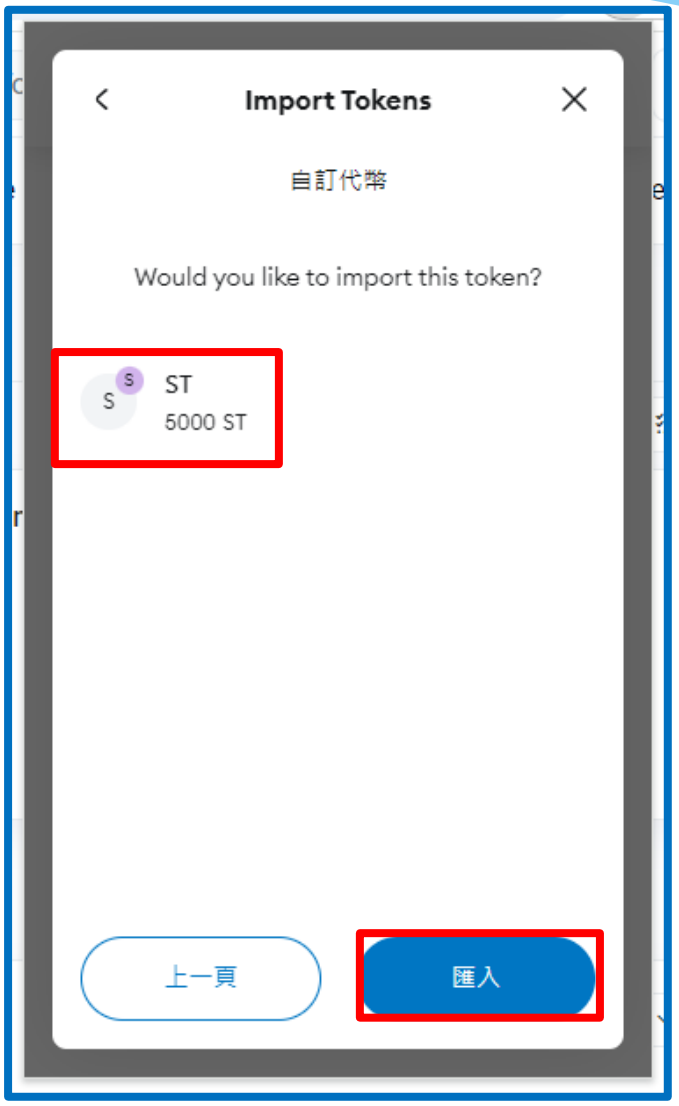


Transfer Token

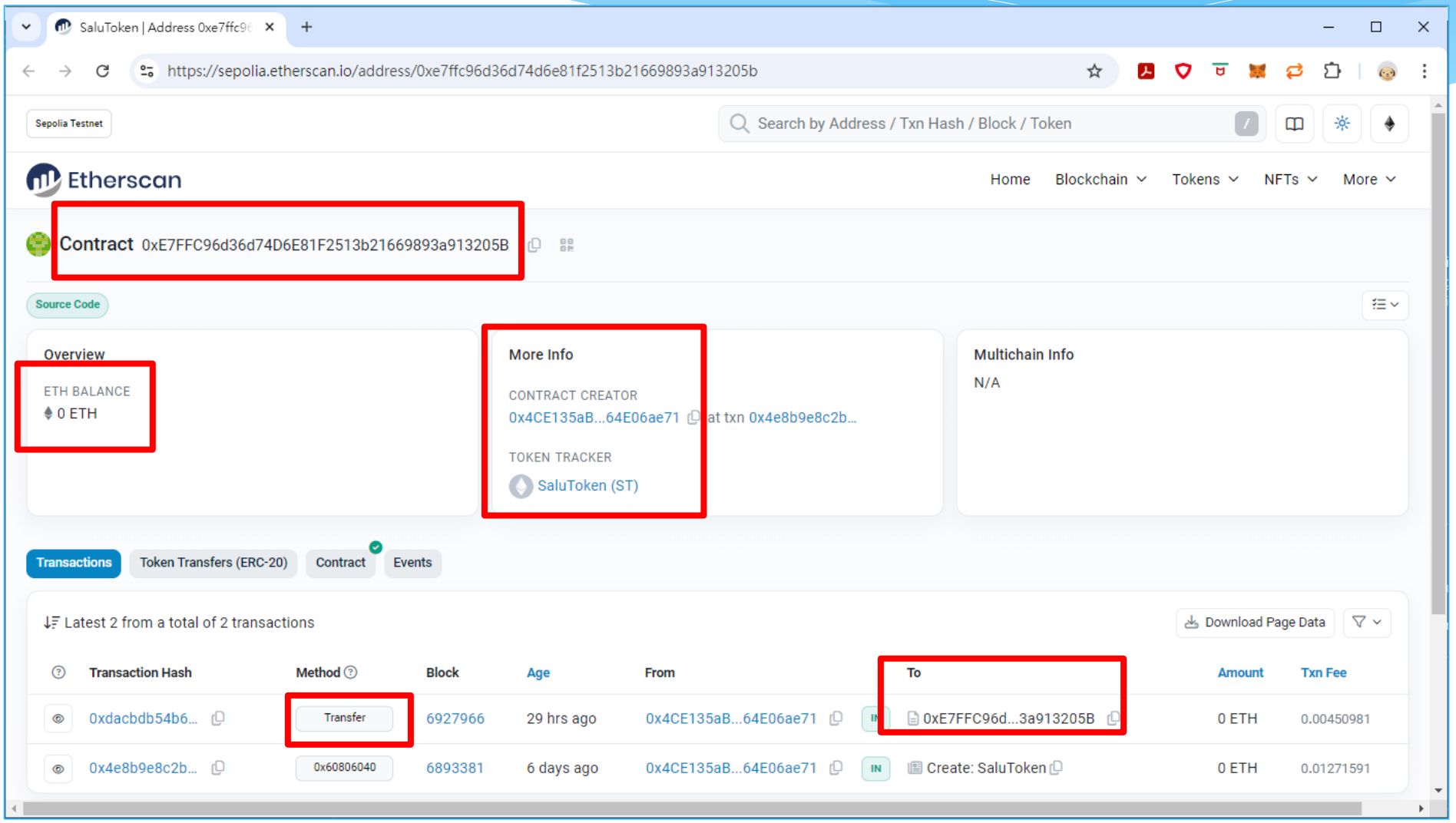
Import Tokens



Import Tokens

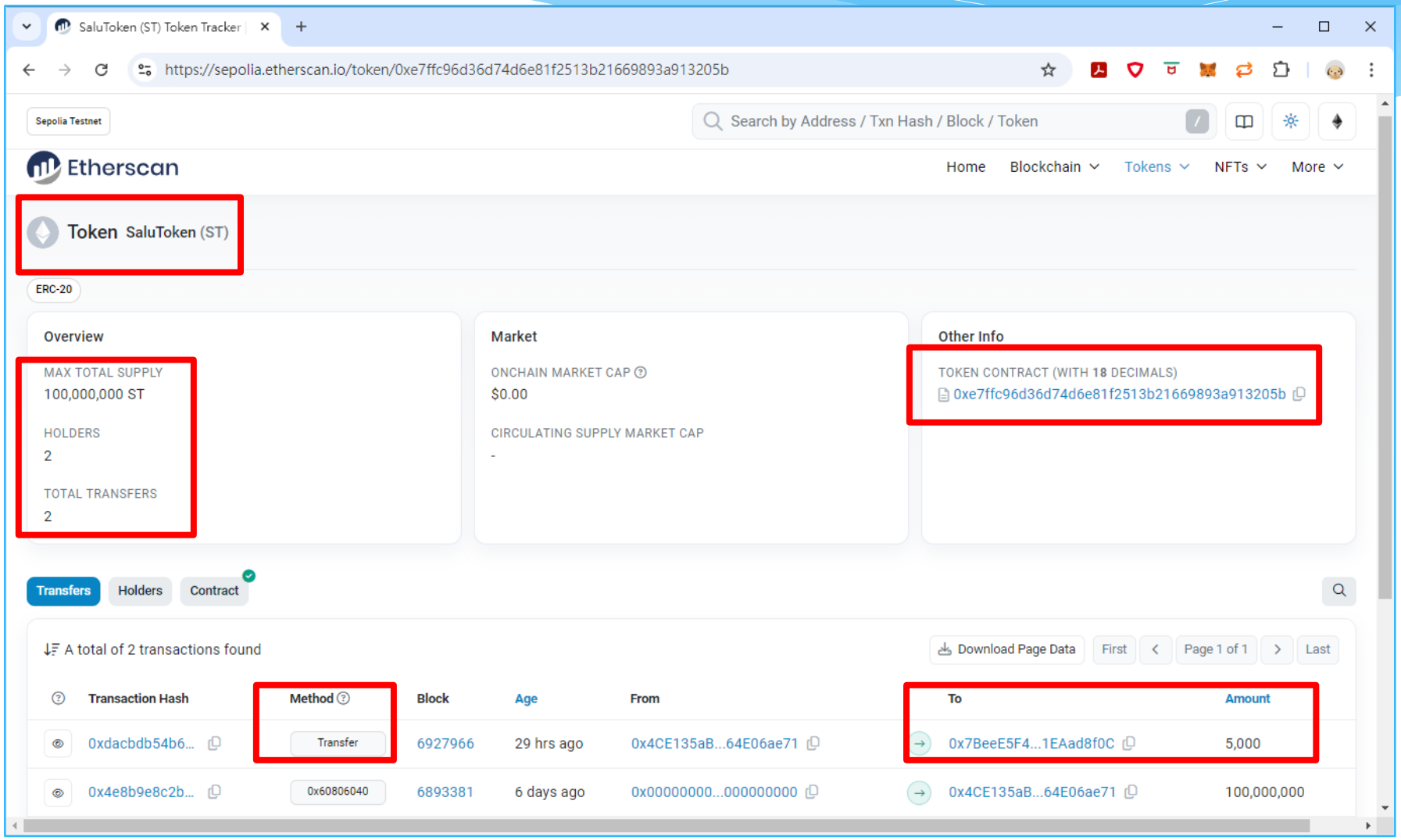


Token Overview



https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b

Token Overview



https://sepolia.etherscan.io/token/0xe7ffc96d36d74d6e81f2513b21669893a913205b

Modify and Extend Token

Buy function

- Allow users to mint new tokens by depositing some ether.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract SaluToken is ERC20 {
    uint256 price = 0.0000001 ether; // price of 1 token in ether

    constructor() ERC20("SaluToken", "ST") {
        _mint(msg.sender, 100000000 * (10 ** uint256(decimals())));
    }

    function buy() external payable {
    require(msg.value > 0, "You must send some ether");
    _mint(msg.sender, msg.value * 10 ** decimals() / price);
   }
}
```

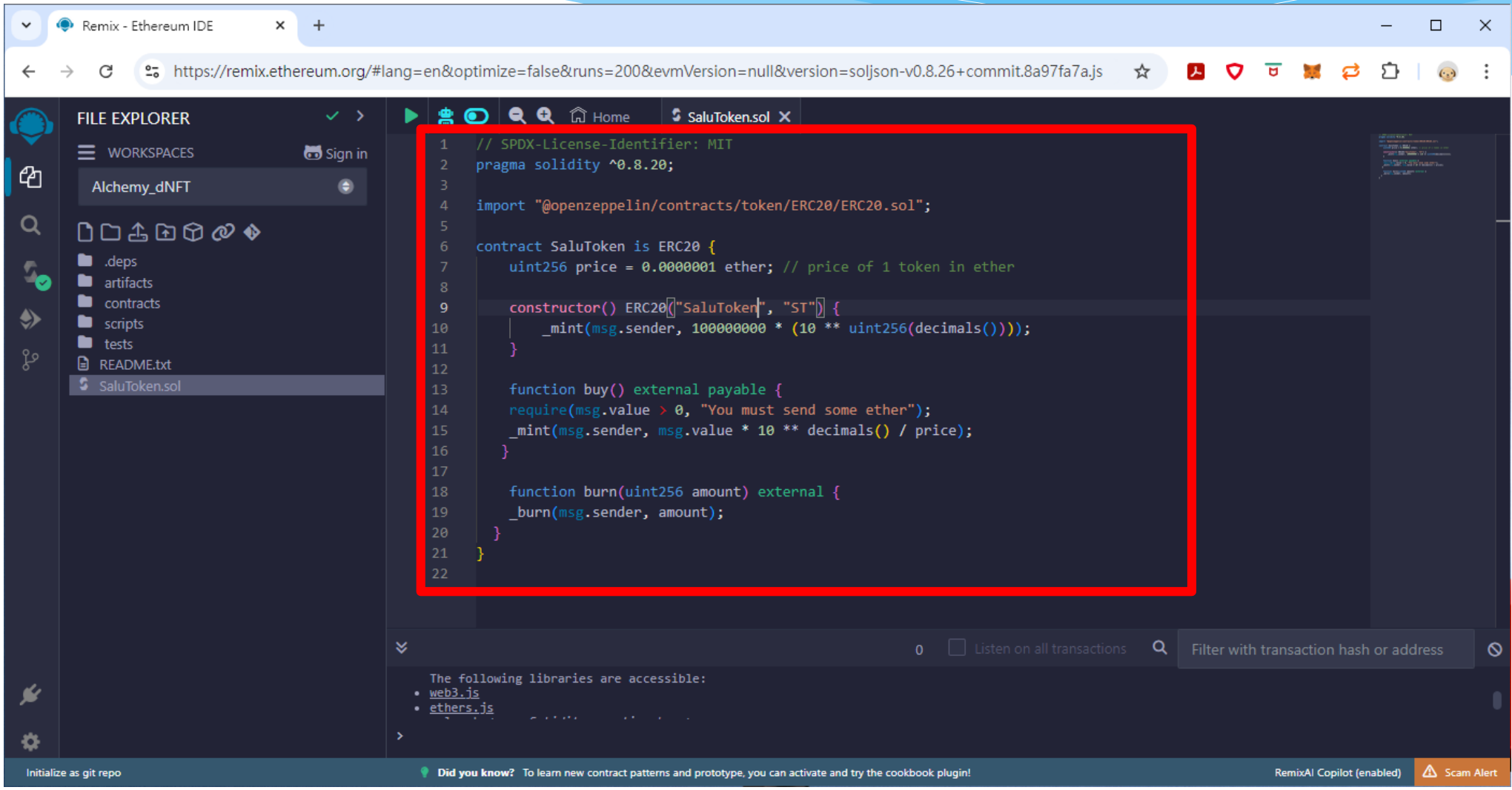
Burn function

- Allowing users to burn their tokens.

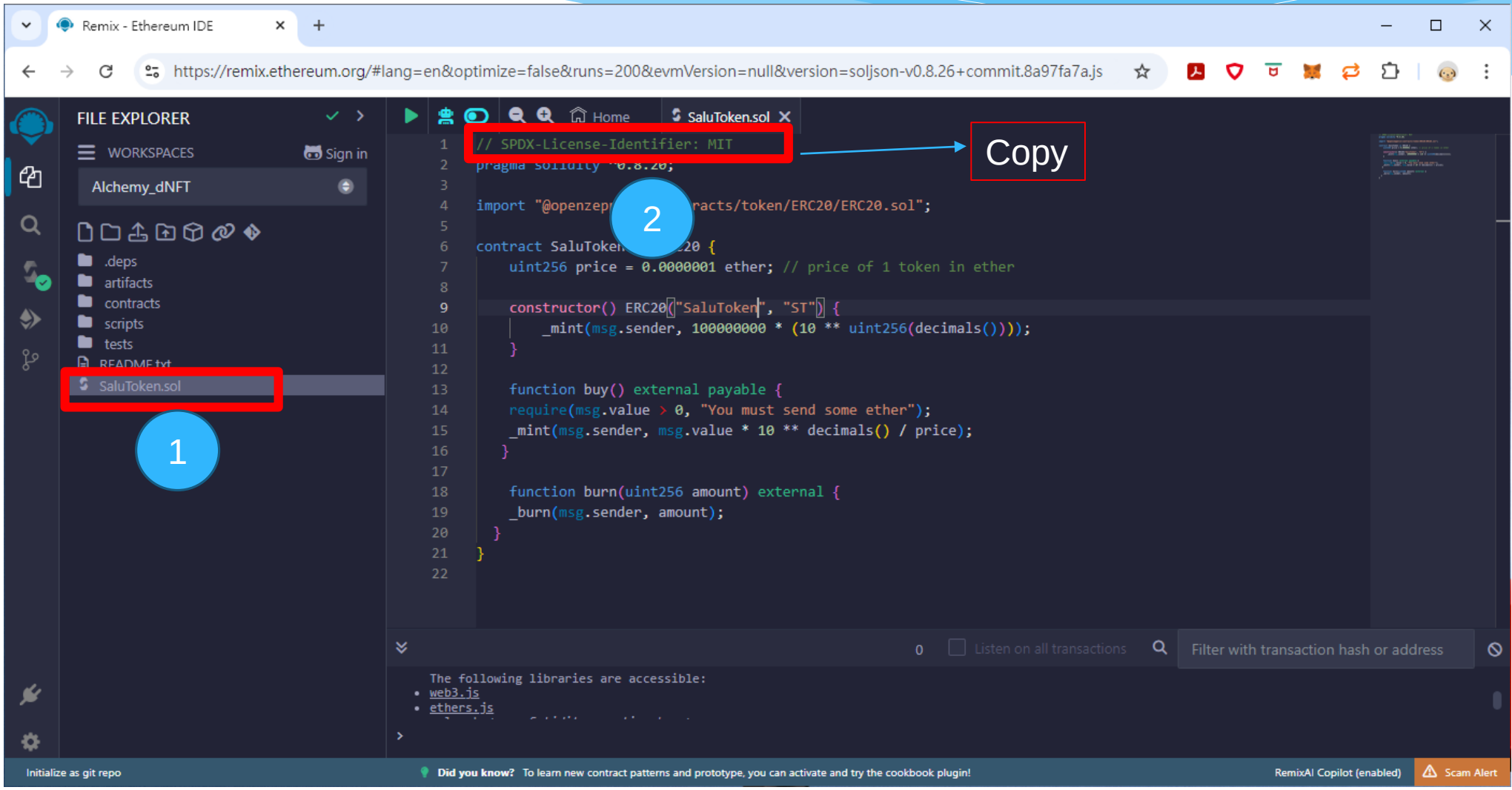
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract SaluToken is ERC20 {
    uint256 price = 0.0000001 ether; // price of 1 token in ether

    constructor() ERC20("SaluToken", "ST") {
        _mint(msg.sender, 100000000 * (10 ** uint256(decimals())));
    }

    function buy() external payable {
    require(msg.value > 0, "You must send some ether");
    _mint(msg.sender, msg.value * 10 ** decimals() / price);
   }

    function burn(uint256 amount) external {
    _burn(msg.sender, amount);
  }
}
```

1. Minting and burnings are treated as transfers, so they should emit a Transfer event.
2. When minting new tokens, _from is 0x0.
3. When burning, _to is 0x0.

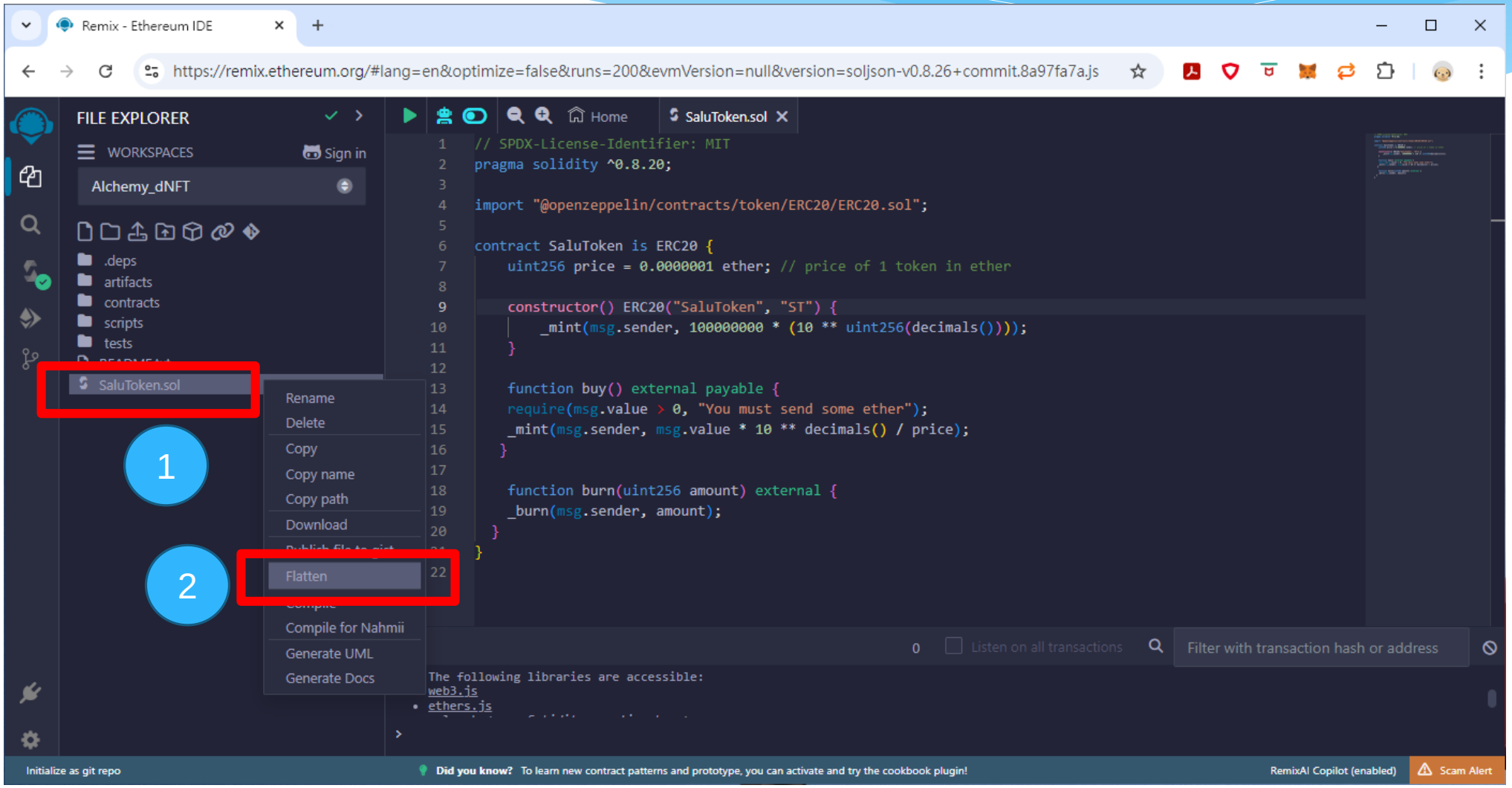
Modify SaluToken.sol



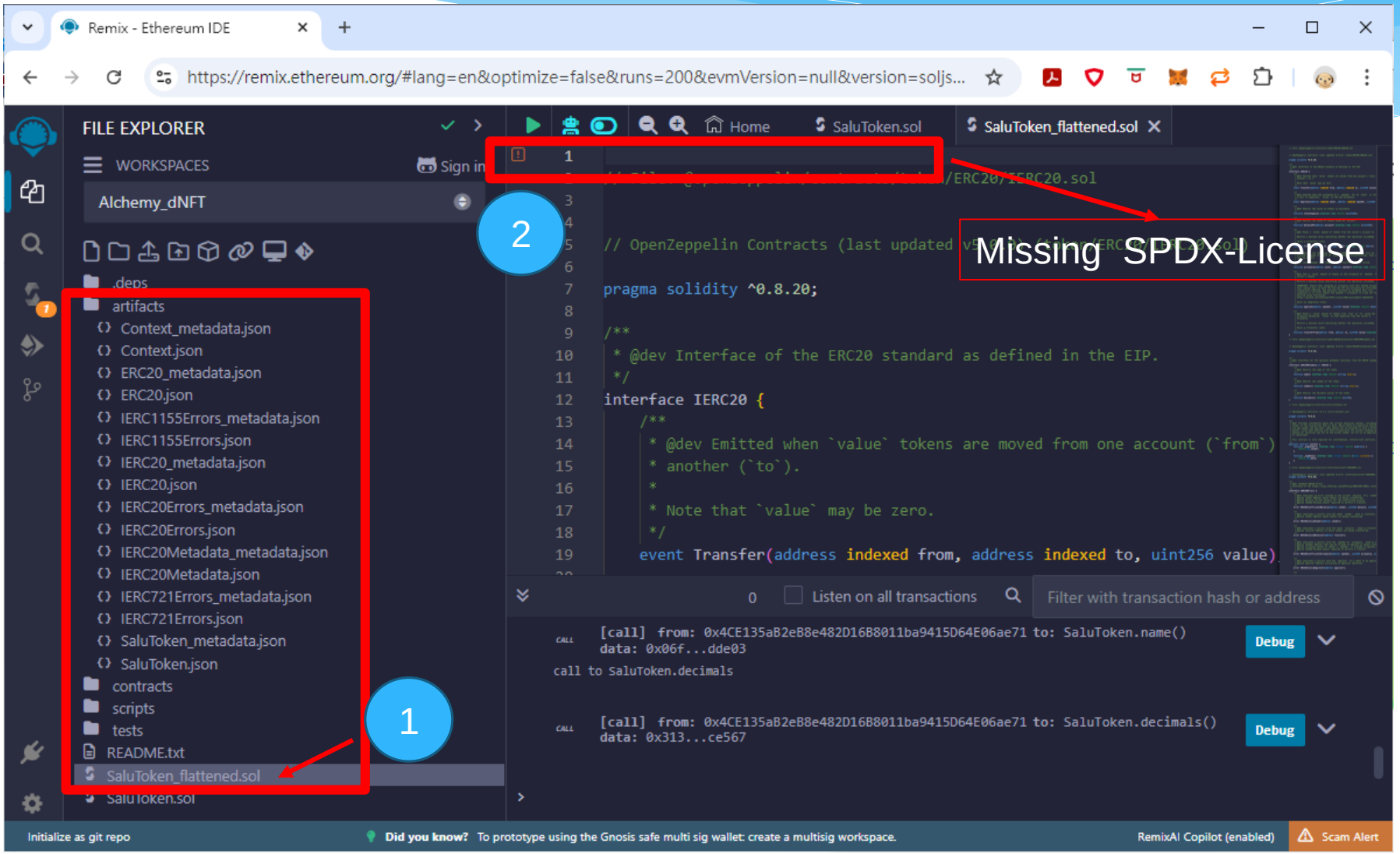
Copy SPDX-License



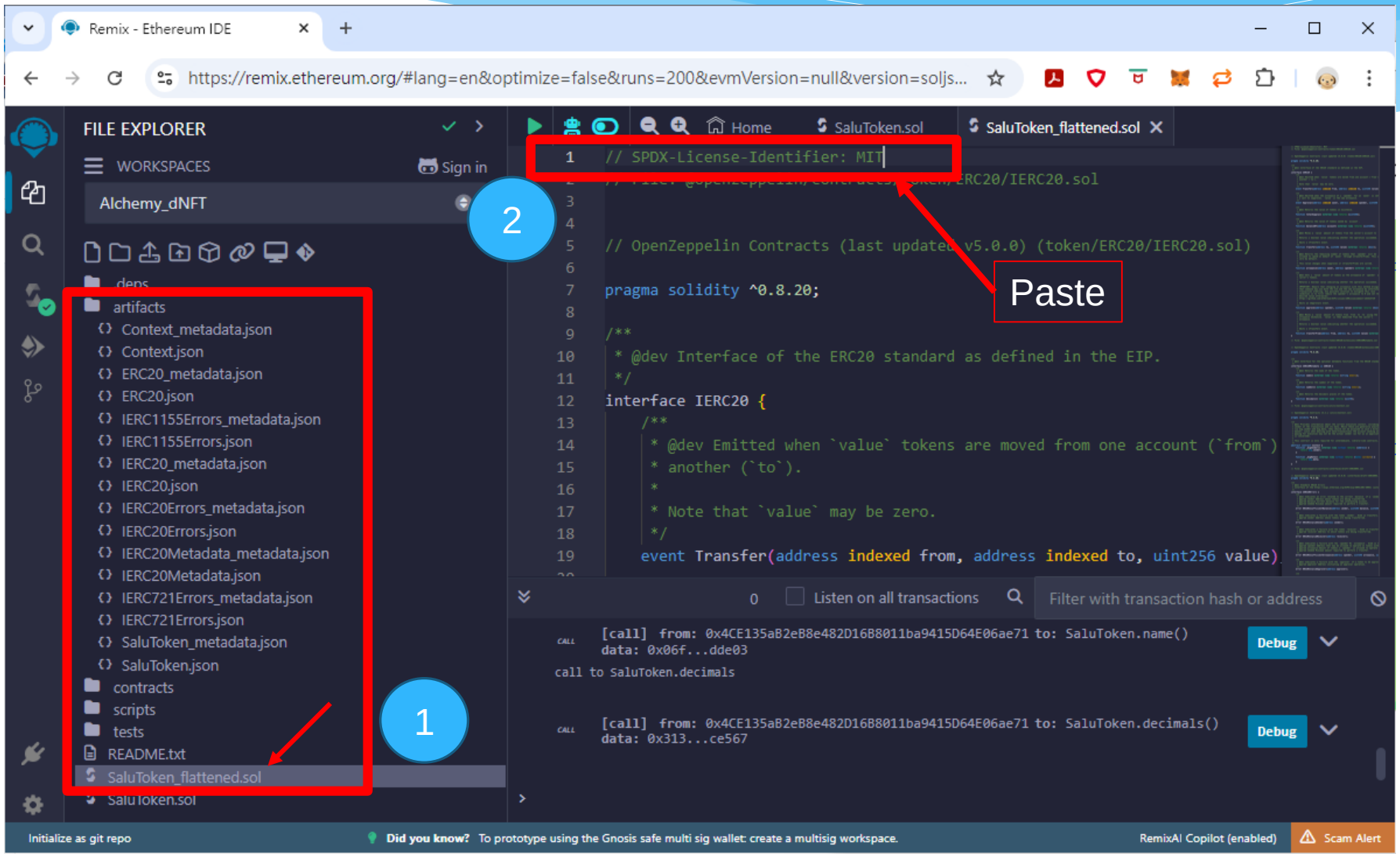
Flatten SaluToken.sol



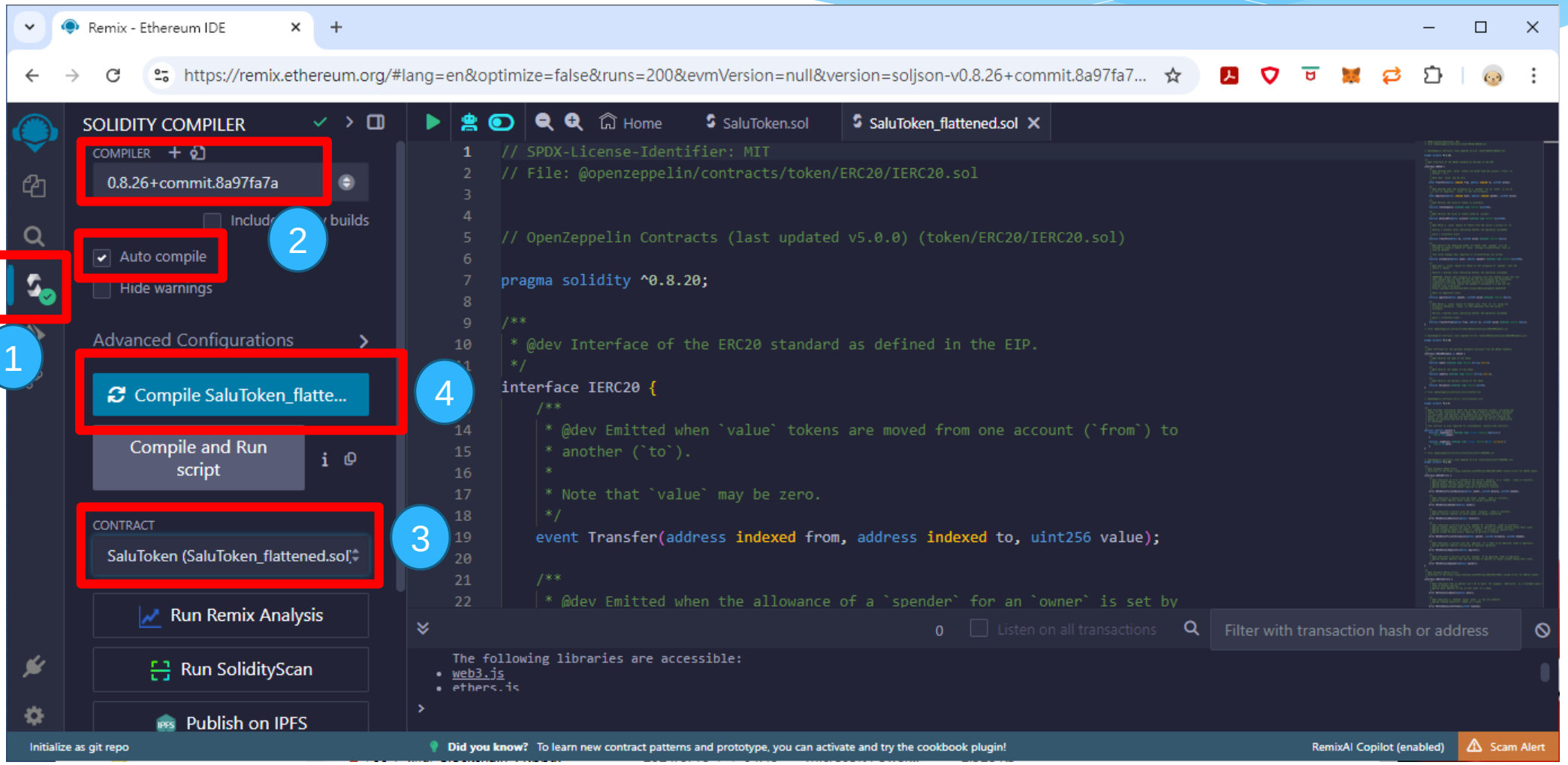
SaluToken_flattened.sol



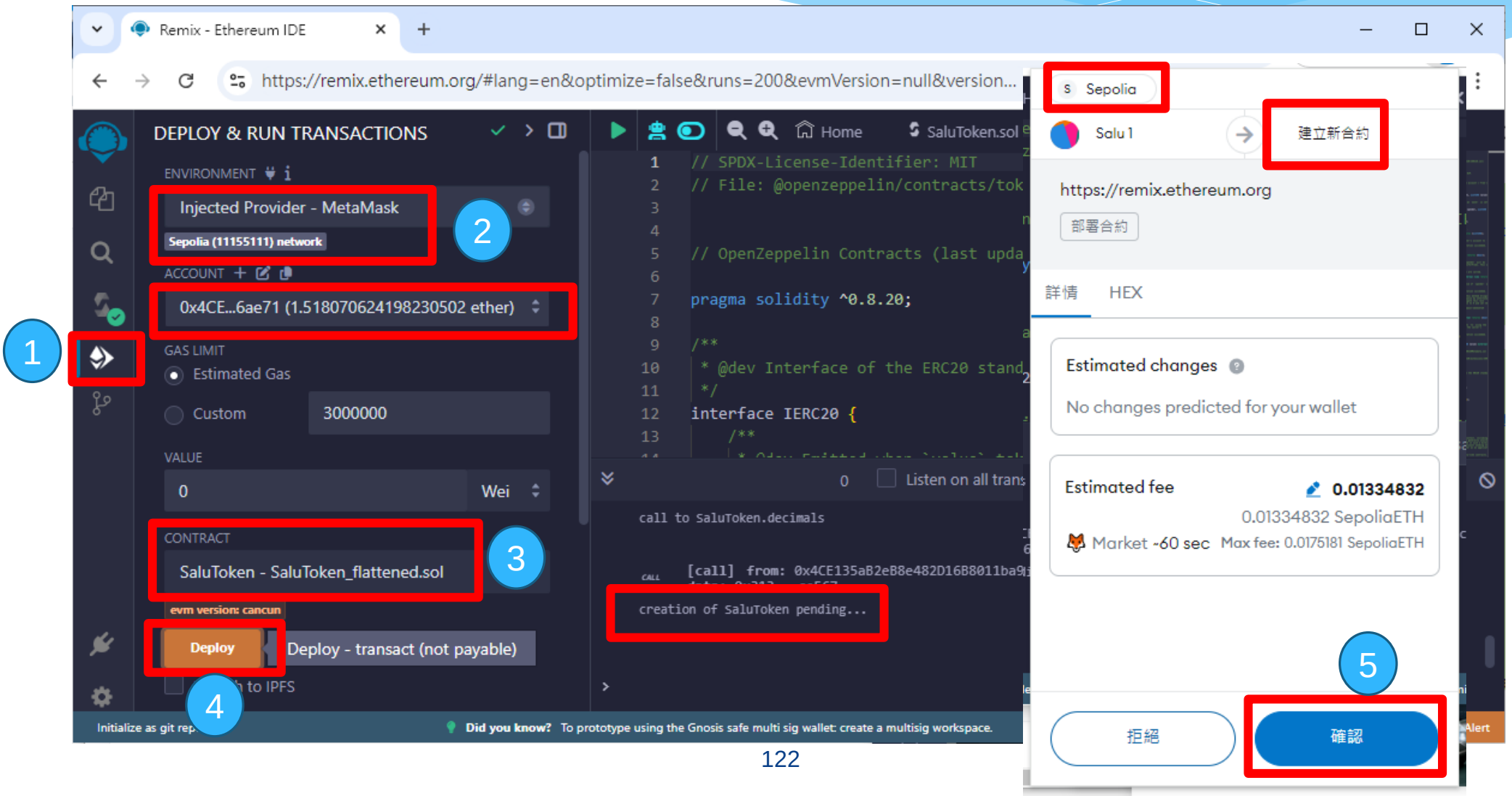
SaluToken_flattened.sol



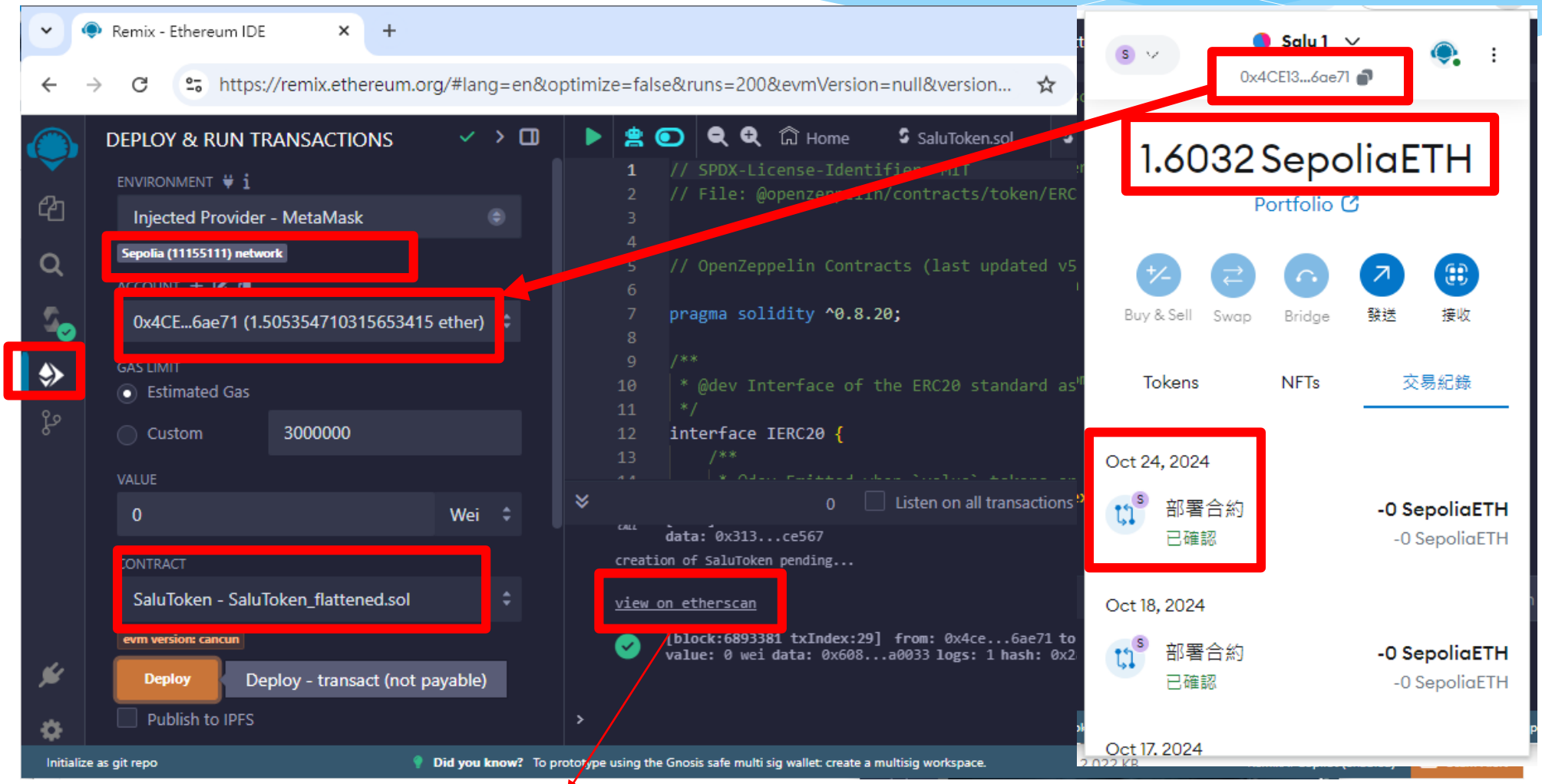
Compile Smart Contract



Deploy an ERC-20 Token

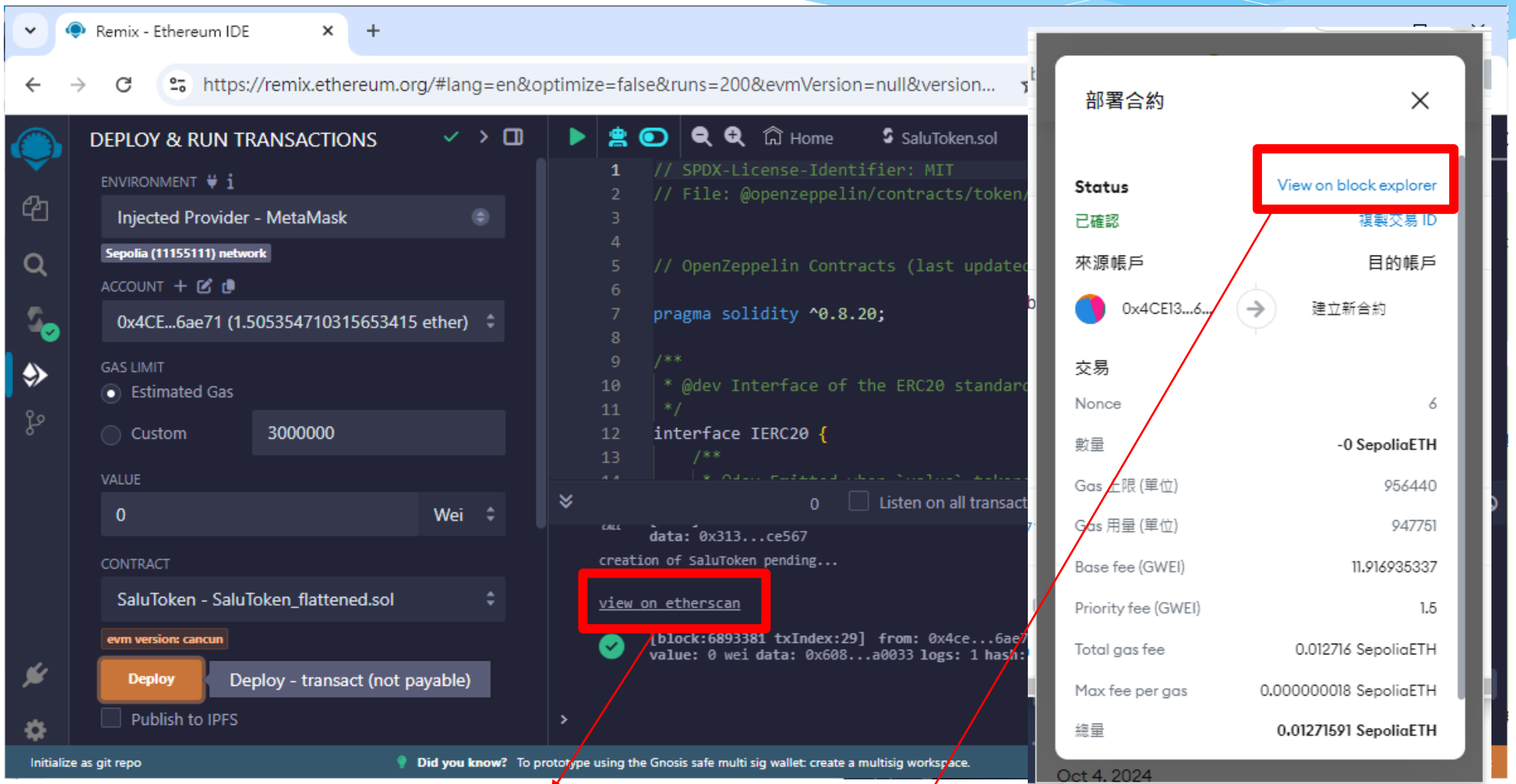


Deploy and Run Transaction



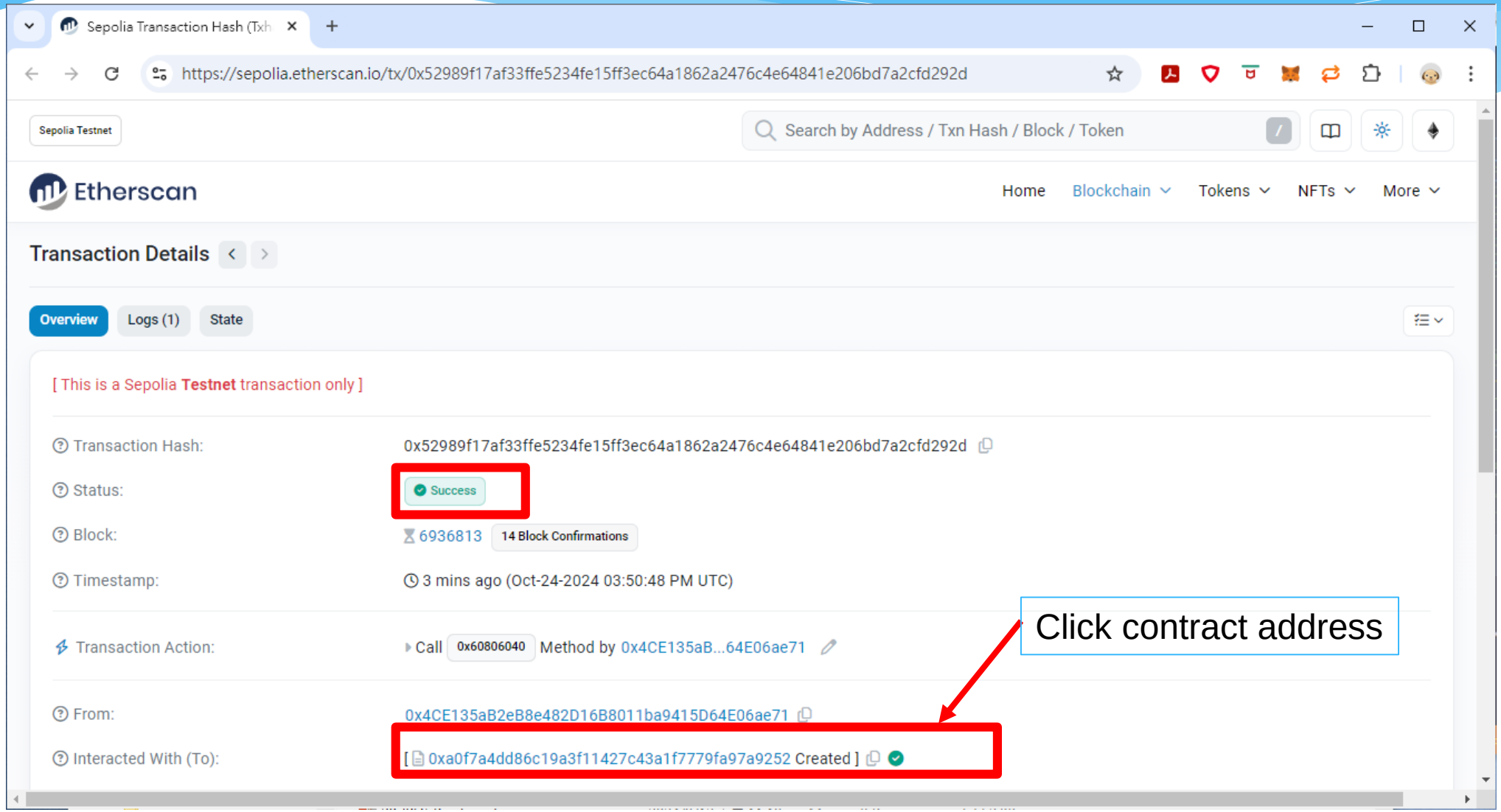
https://sepolia.etherscan.io/tx/0x52989f17af33ffe5234fe15ff3ec64a1862a2476c4e64841e206bd7a2cfd292d

Deploy and Run Transaction



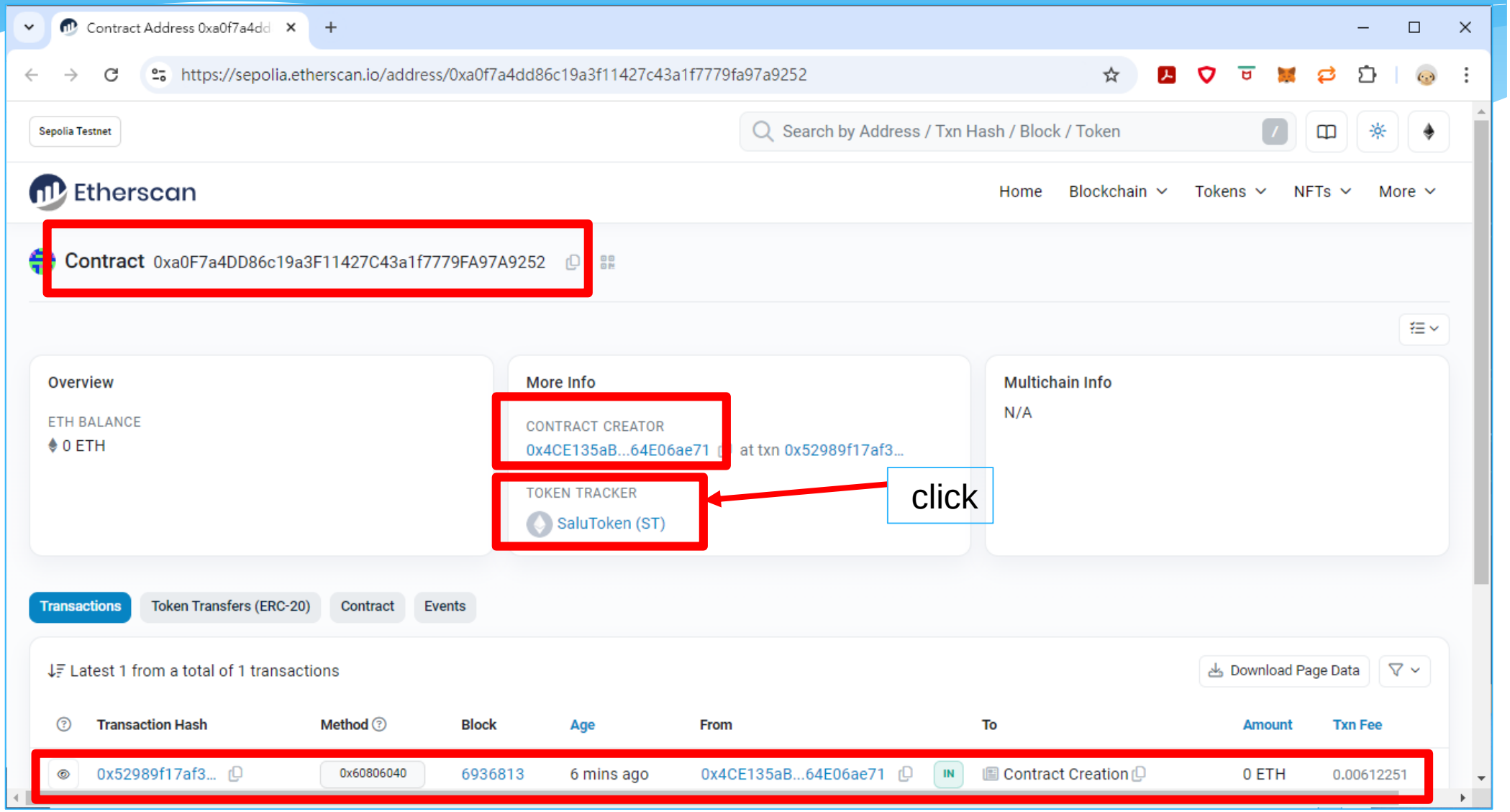
https://sepolia.etherscan.io/tx/0x52989f17af33ffe5234fe15ff3ec64a1862a2476c4e64841e206bd7a2cfd292d

View Transaction on Sepolia



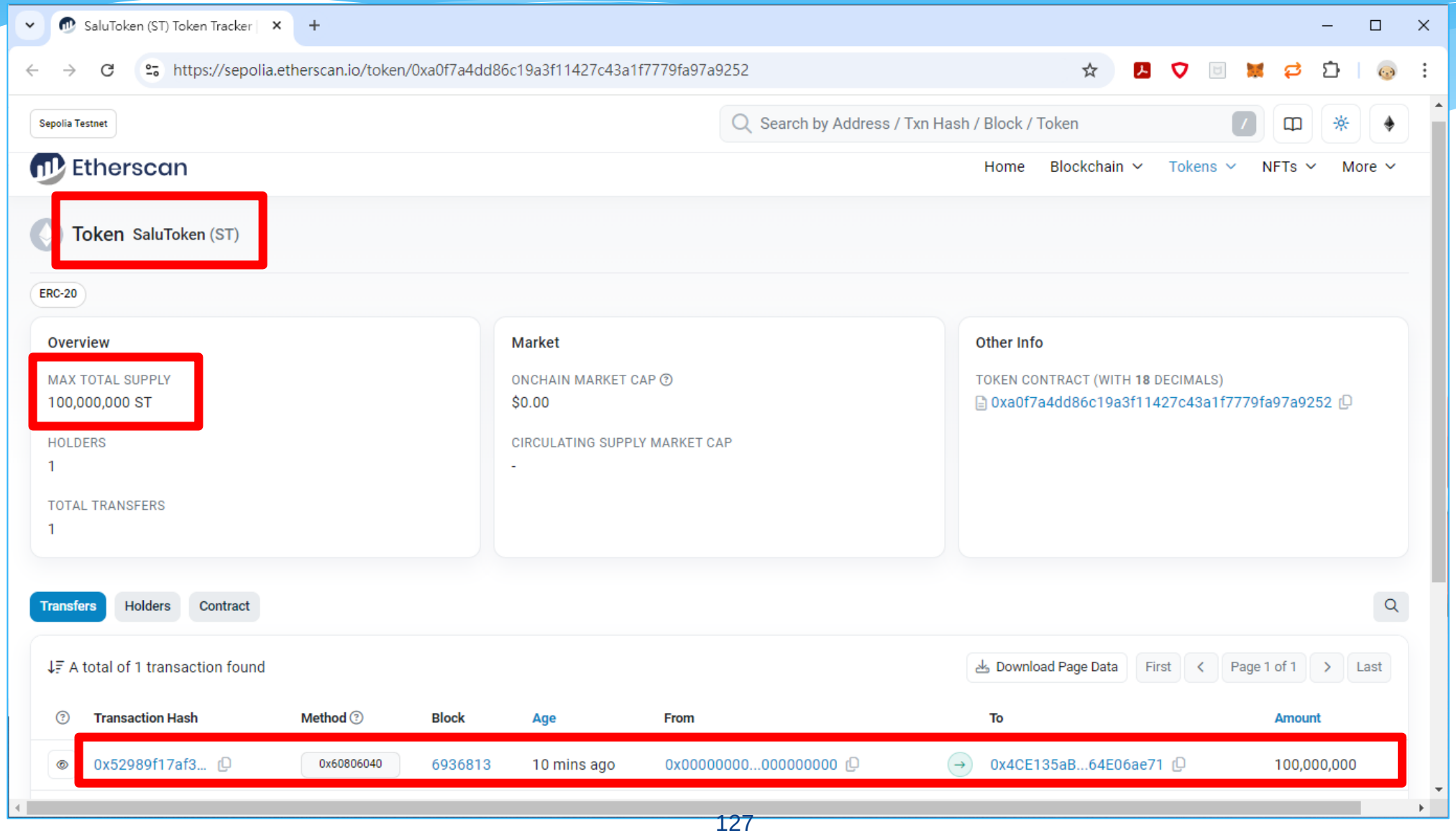
https://sepolia.etherscan.io/tx/0x52989f17af33ffe5234fe15ff3ec64a1862a2476c4e64841e206bd7a2cfd292d

Smart Contract Information



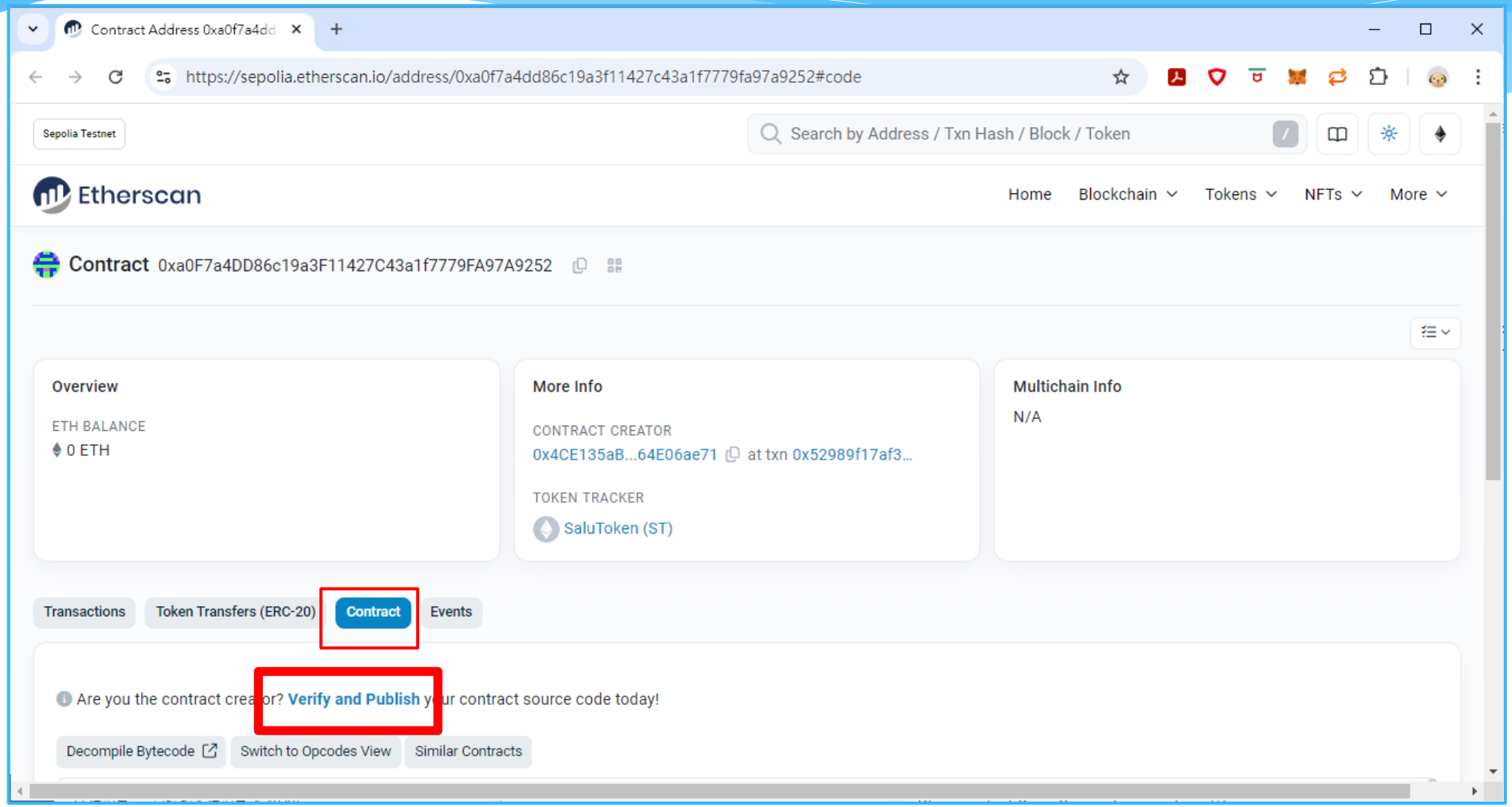
https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252

Smart Contract Information



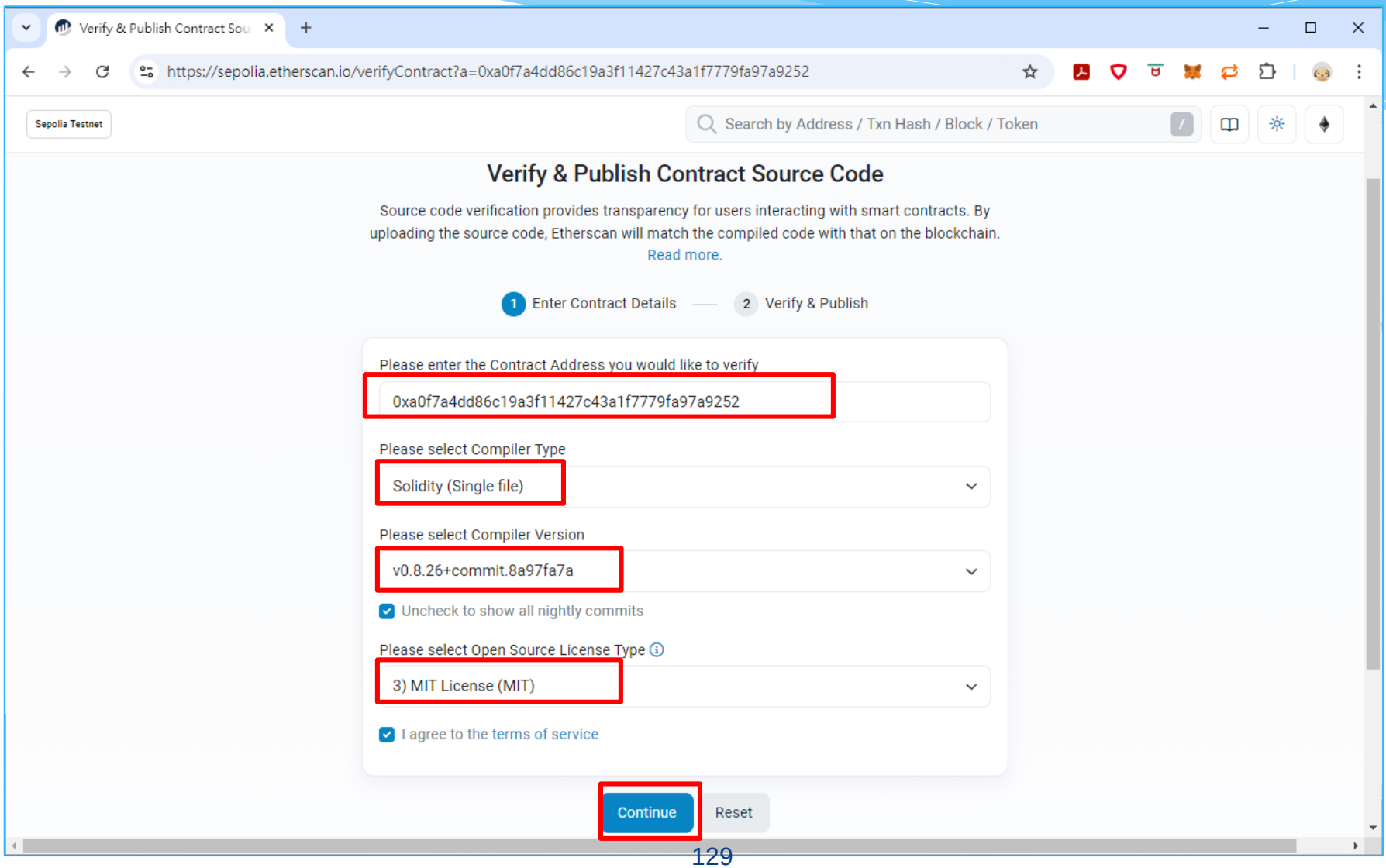
https://sepolia.etherscan.io/token/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252

View & Publish



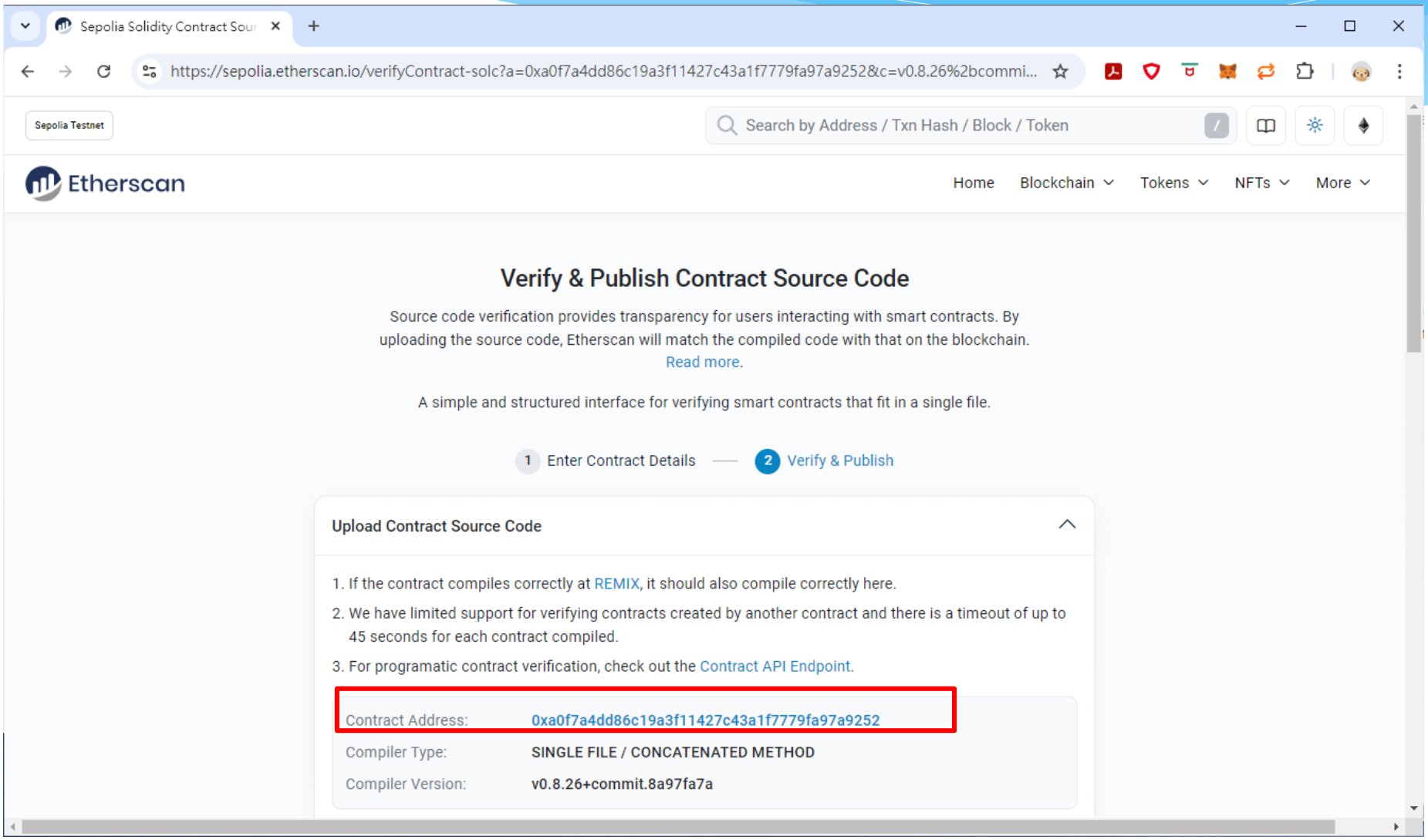
https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#code

View & Publish Contract Source Code

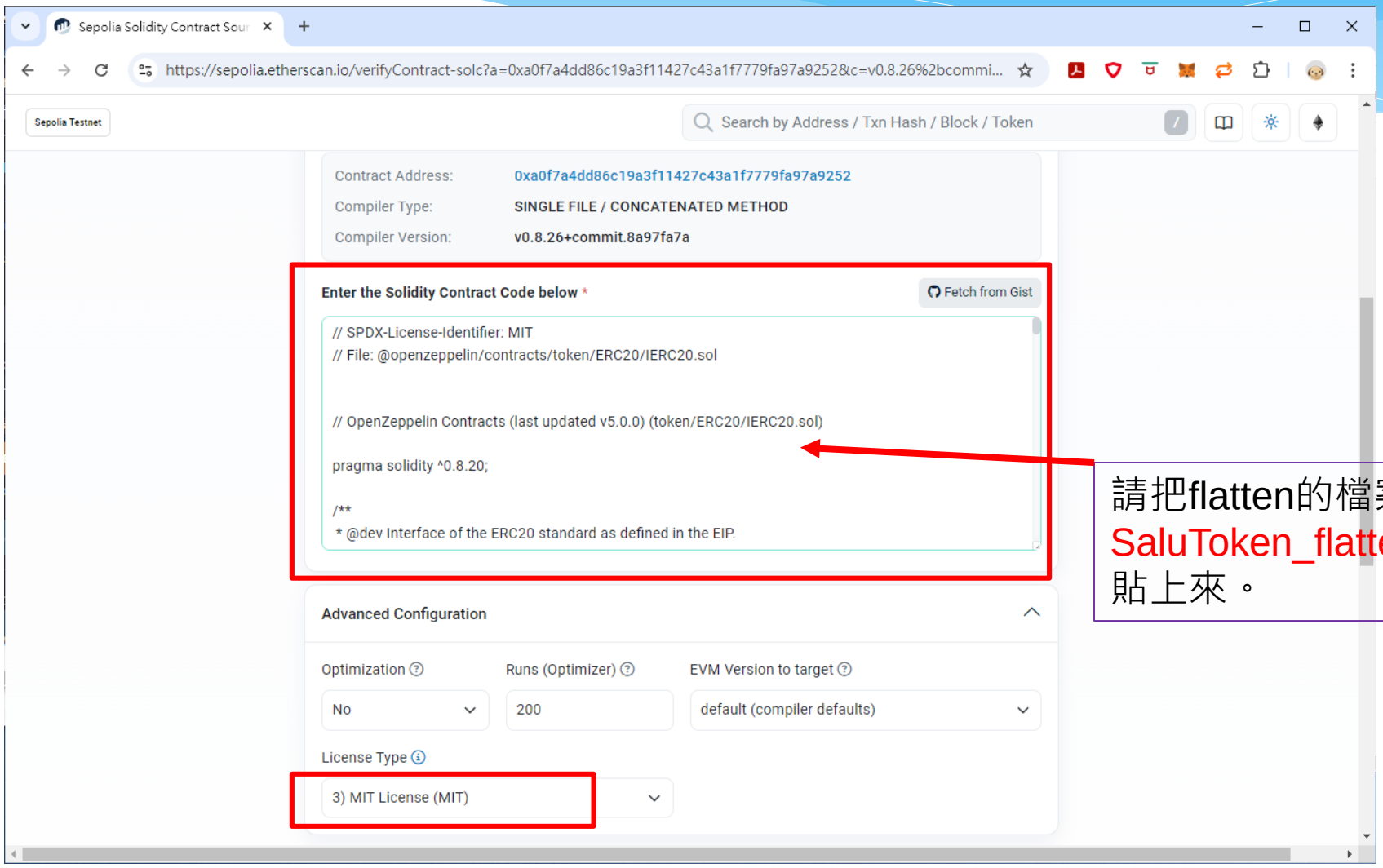


https://sepolia.etherscan.io/verifyContract?a=0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252

View & Publish Contract Source Code

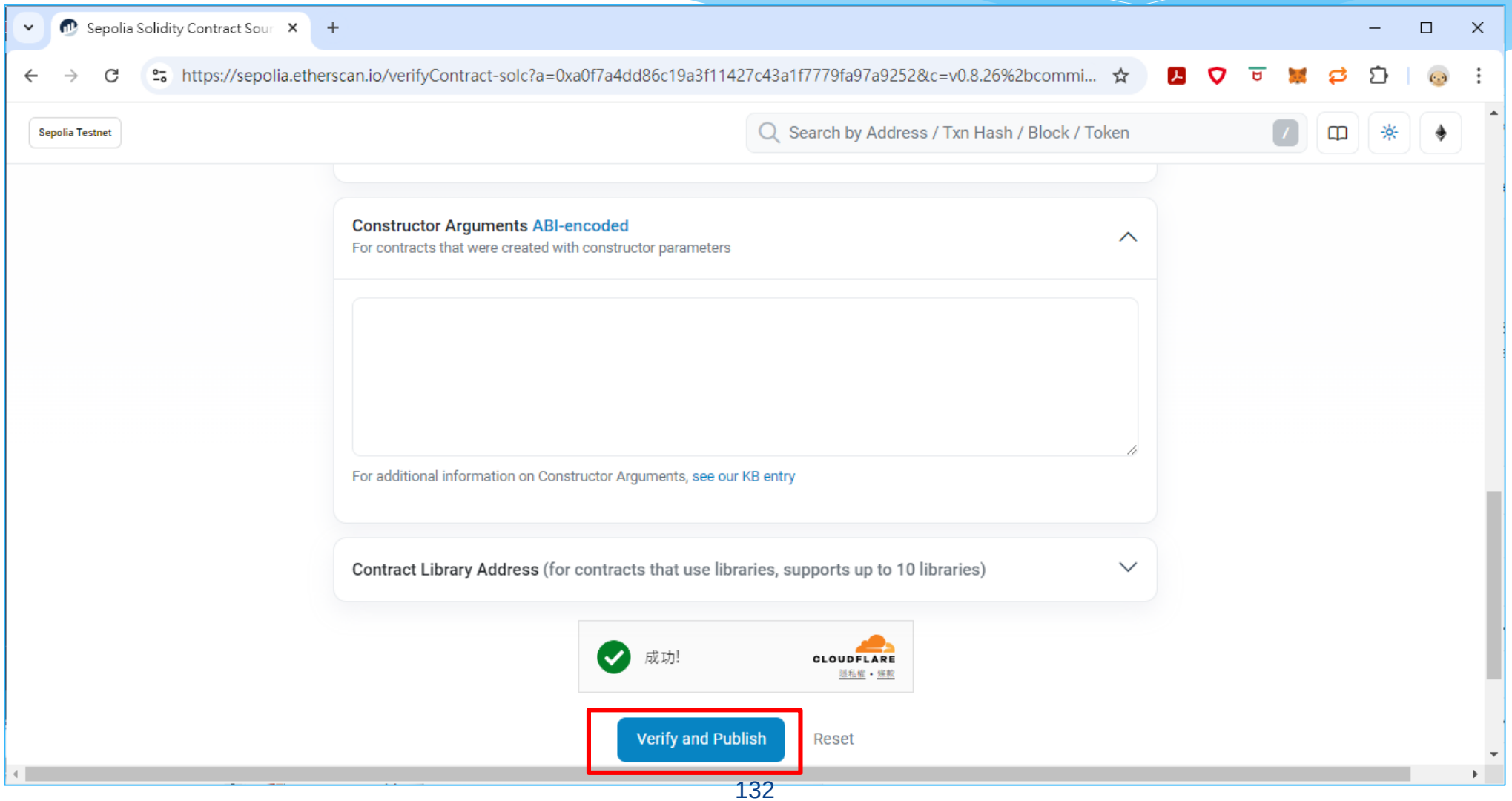


View & Publish Contract Source Code

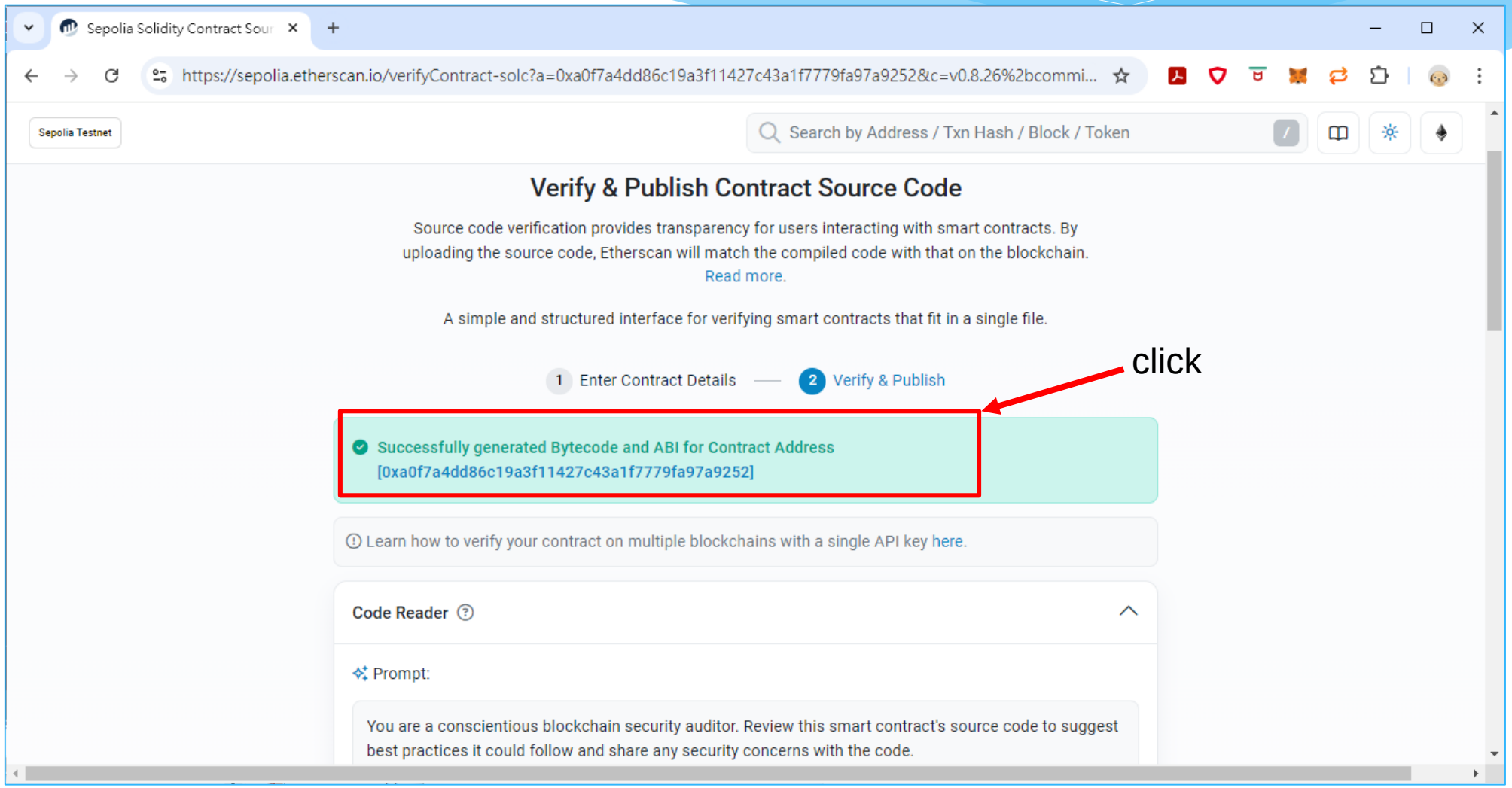


請把flatten的檔案 SaluToken_flattened.sol 貼上來。

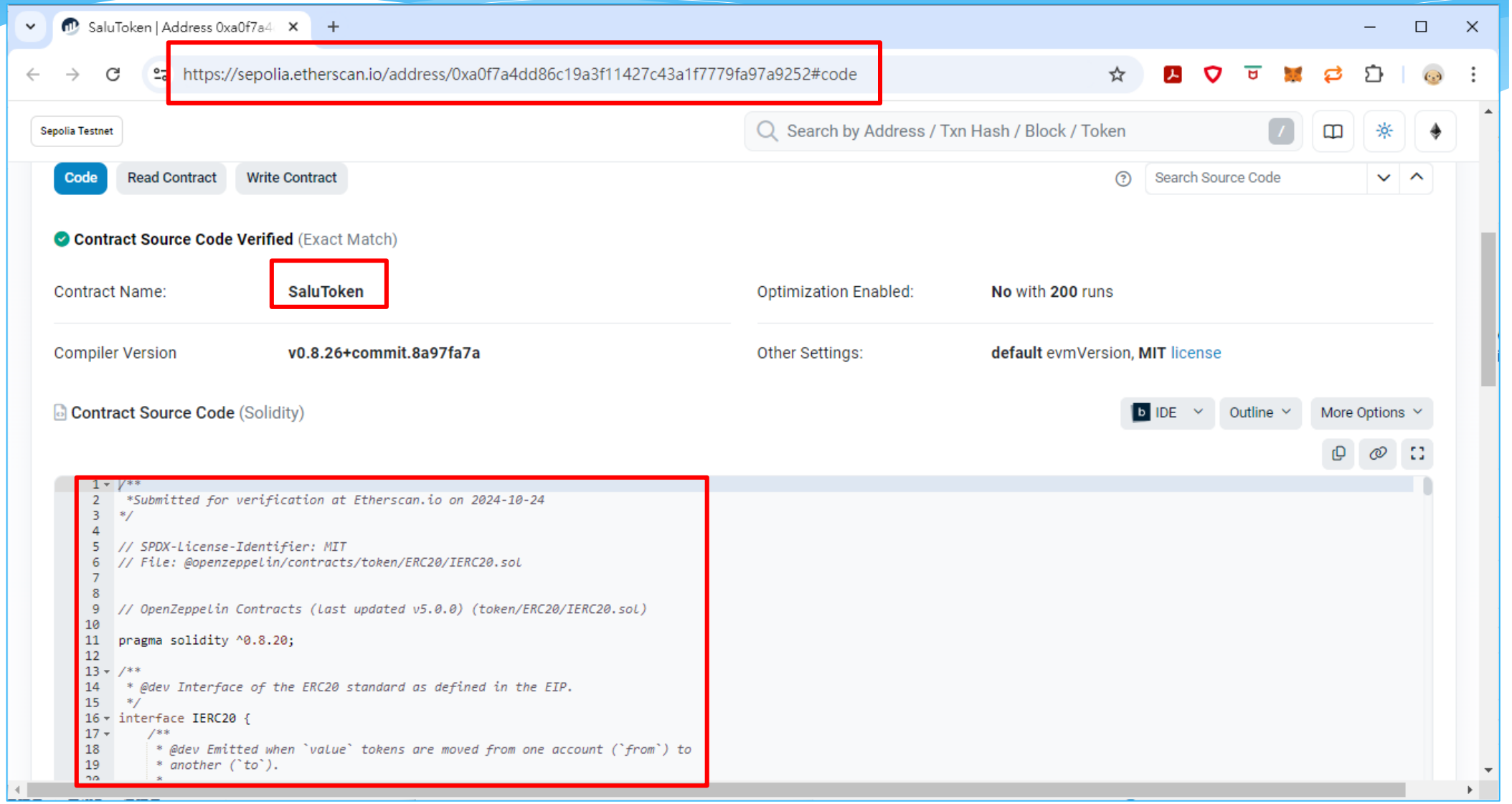
View & Publish Contract Source Code



View & Publish Contract Source Code

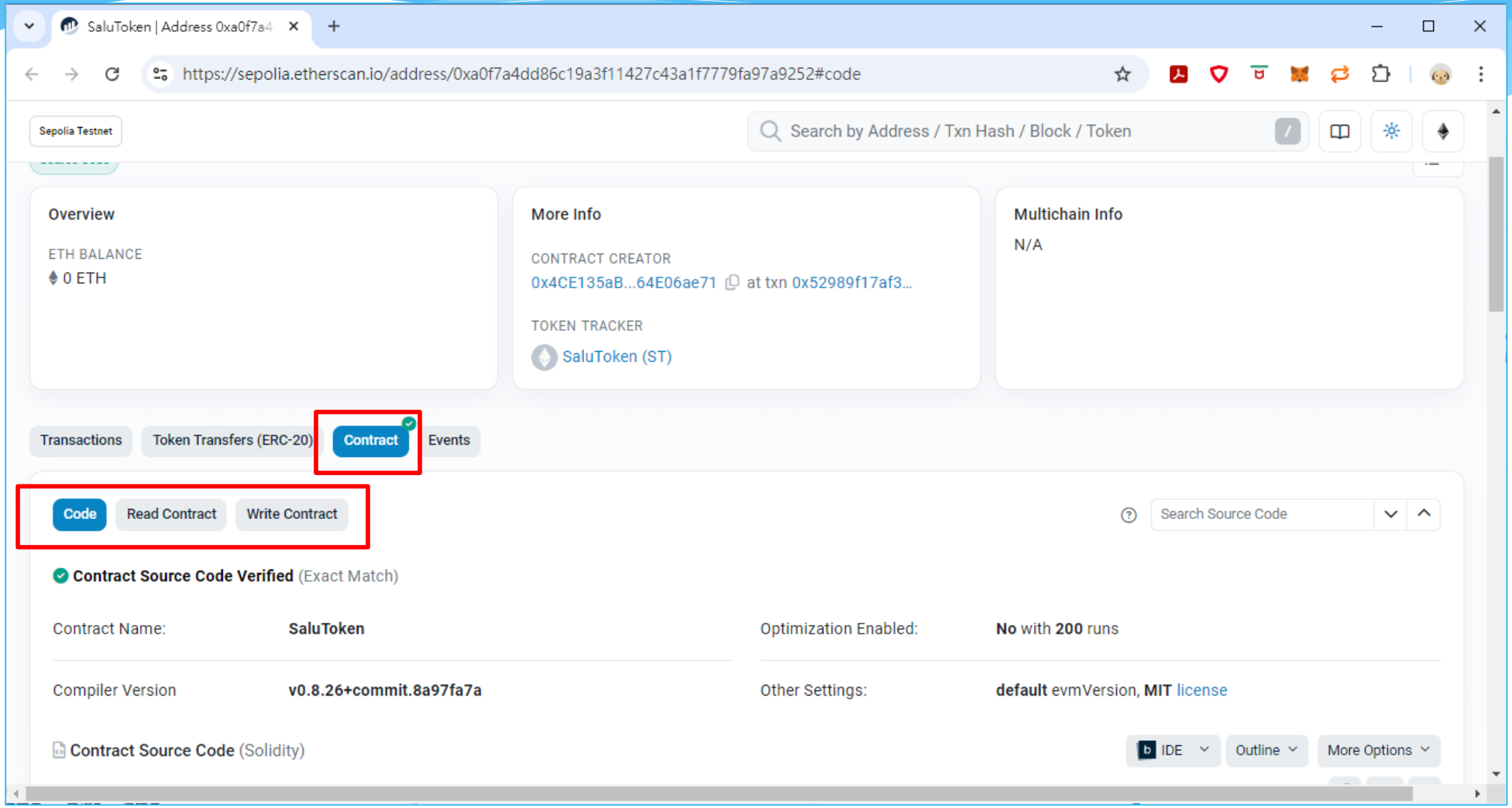


View Contract Source Code



https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#code

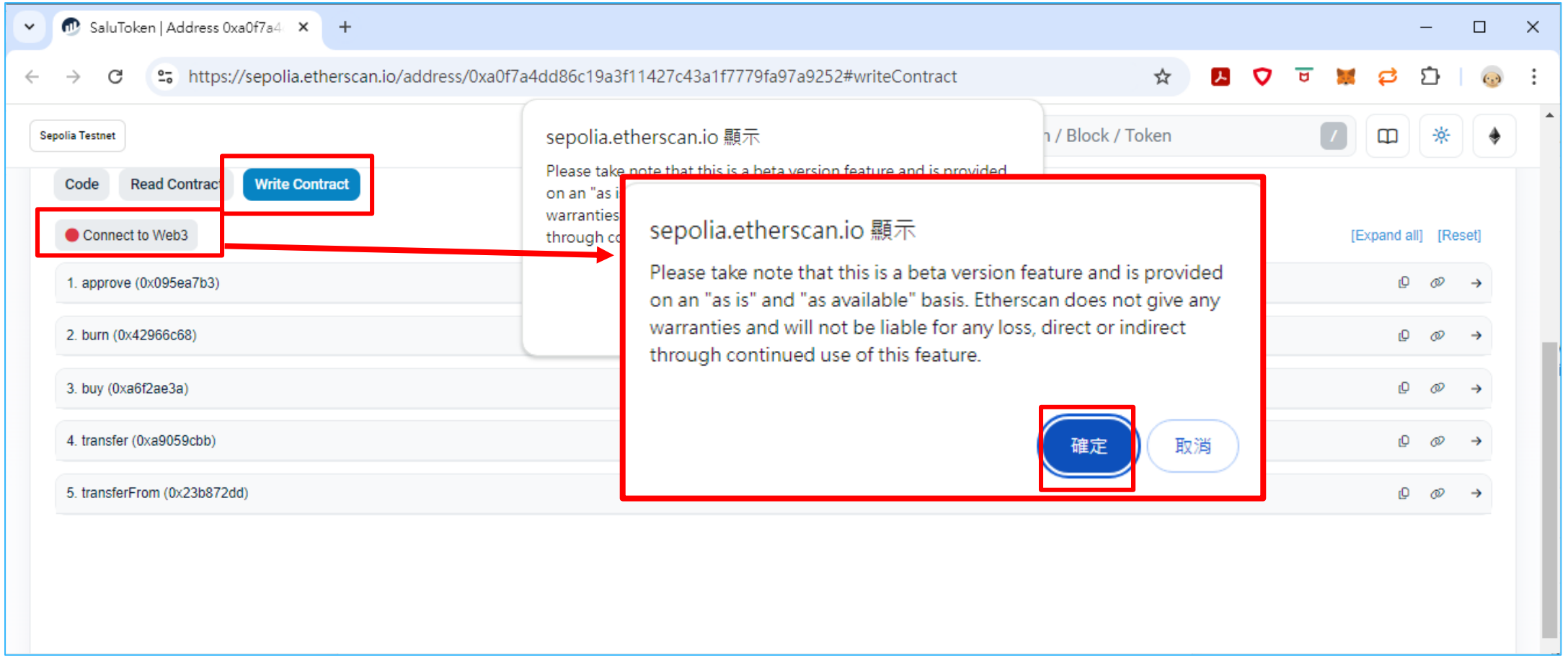
View & Publish Contract Source Code



https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#code

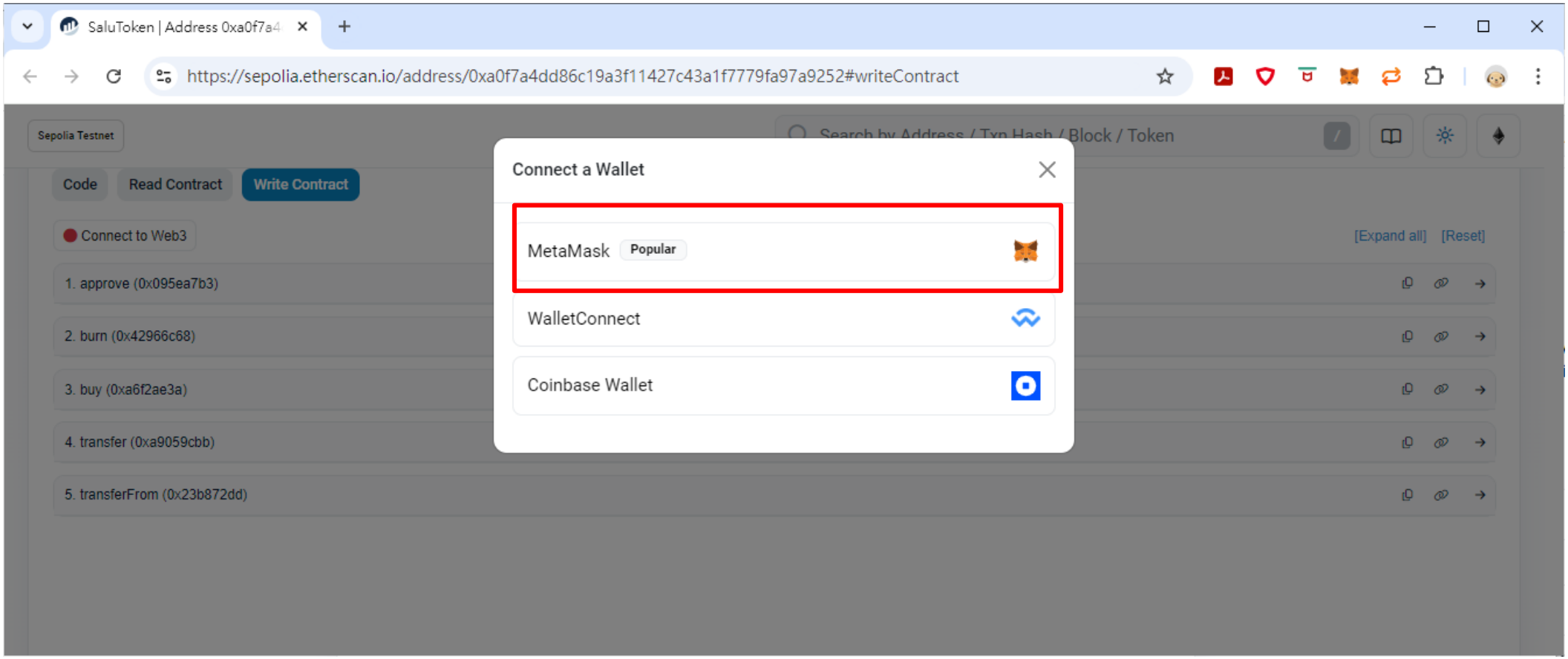
Buy Token

Connect to Web3



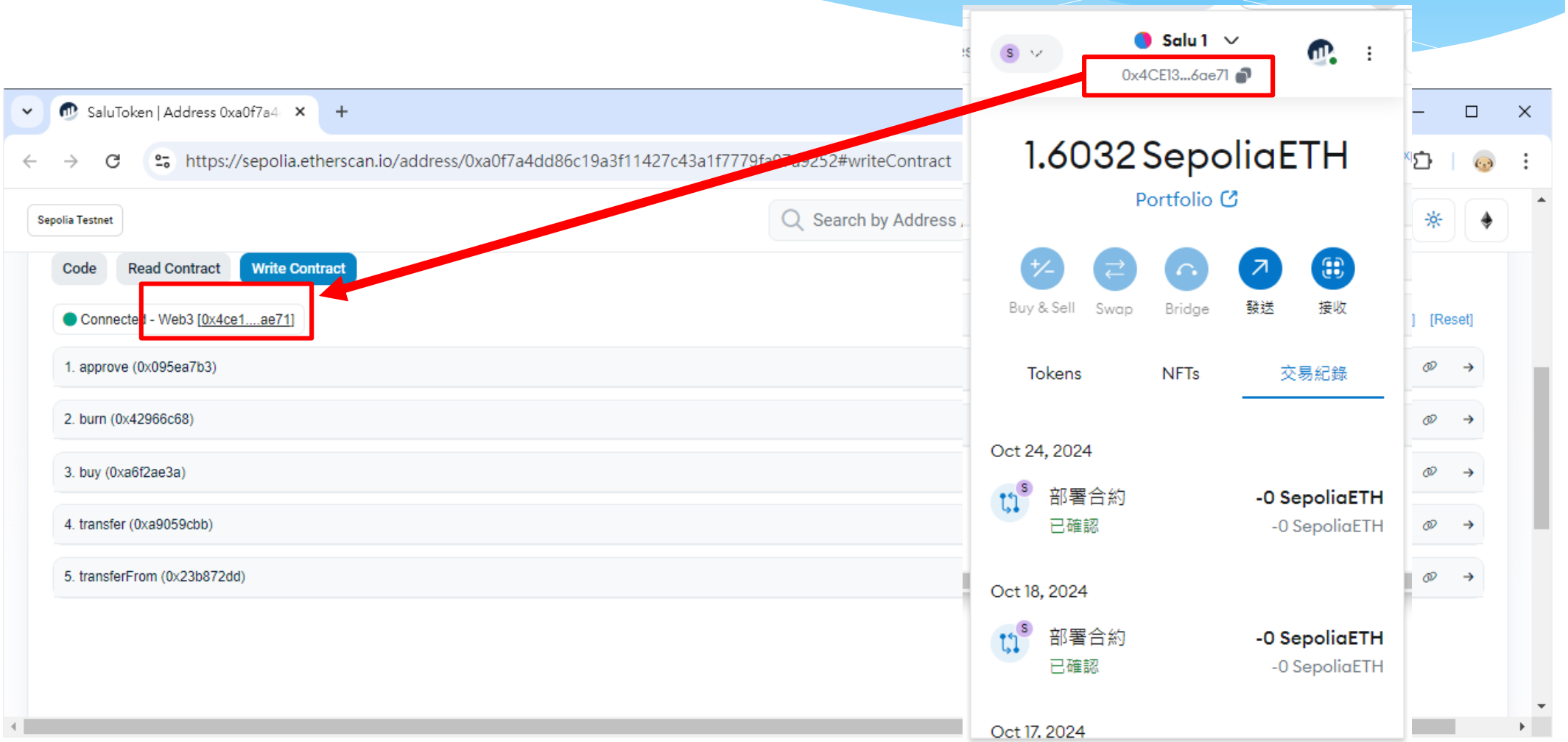
https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#writeContract

Connect a Wallet-MetaMask



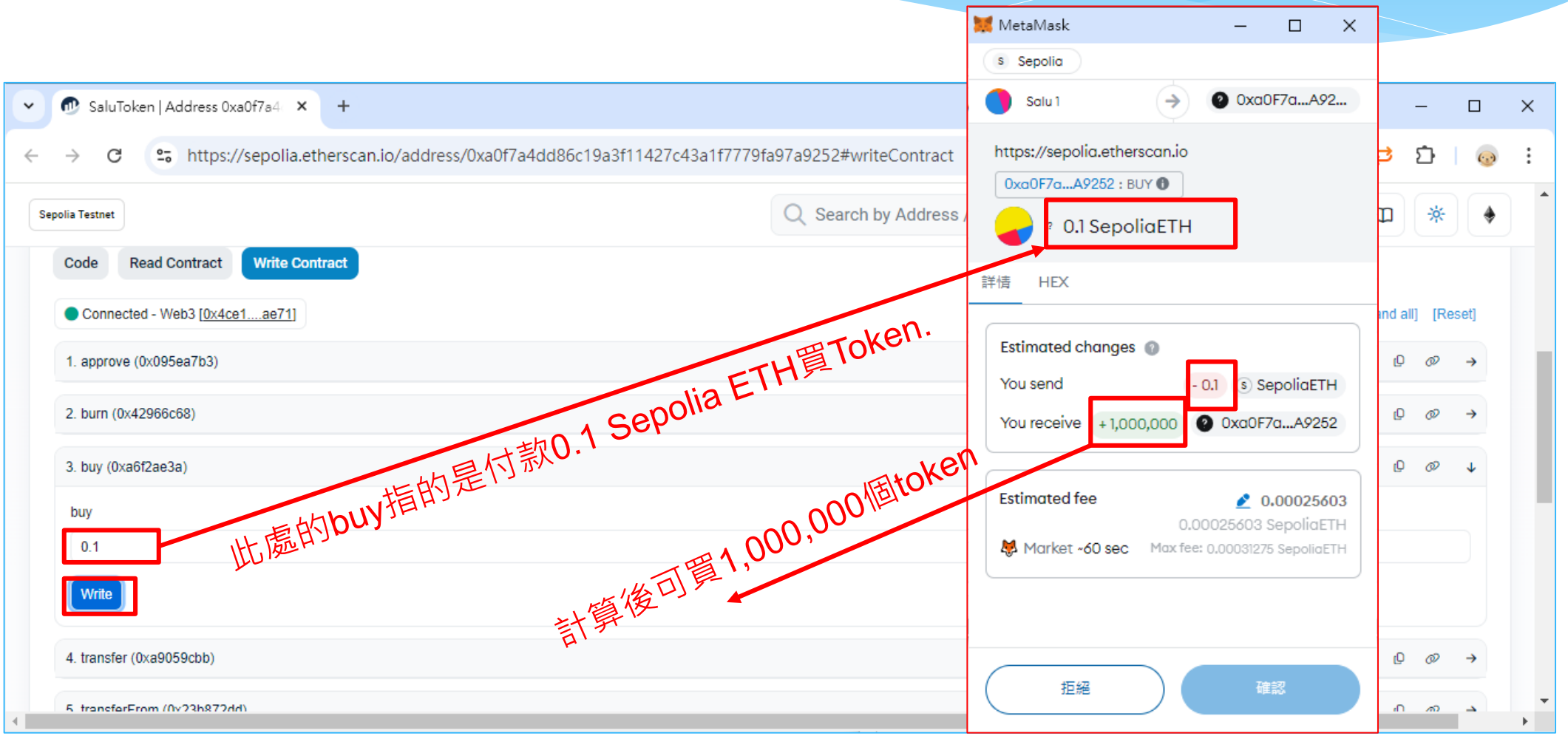
https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#writeContract

Connected Web3 using account



https://sepolia.etherscan.io/address/0xe7ffc96d36d74d6e81f2513b21669893a913205b#readContract

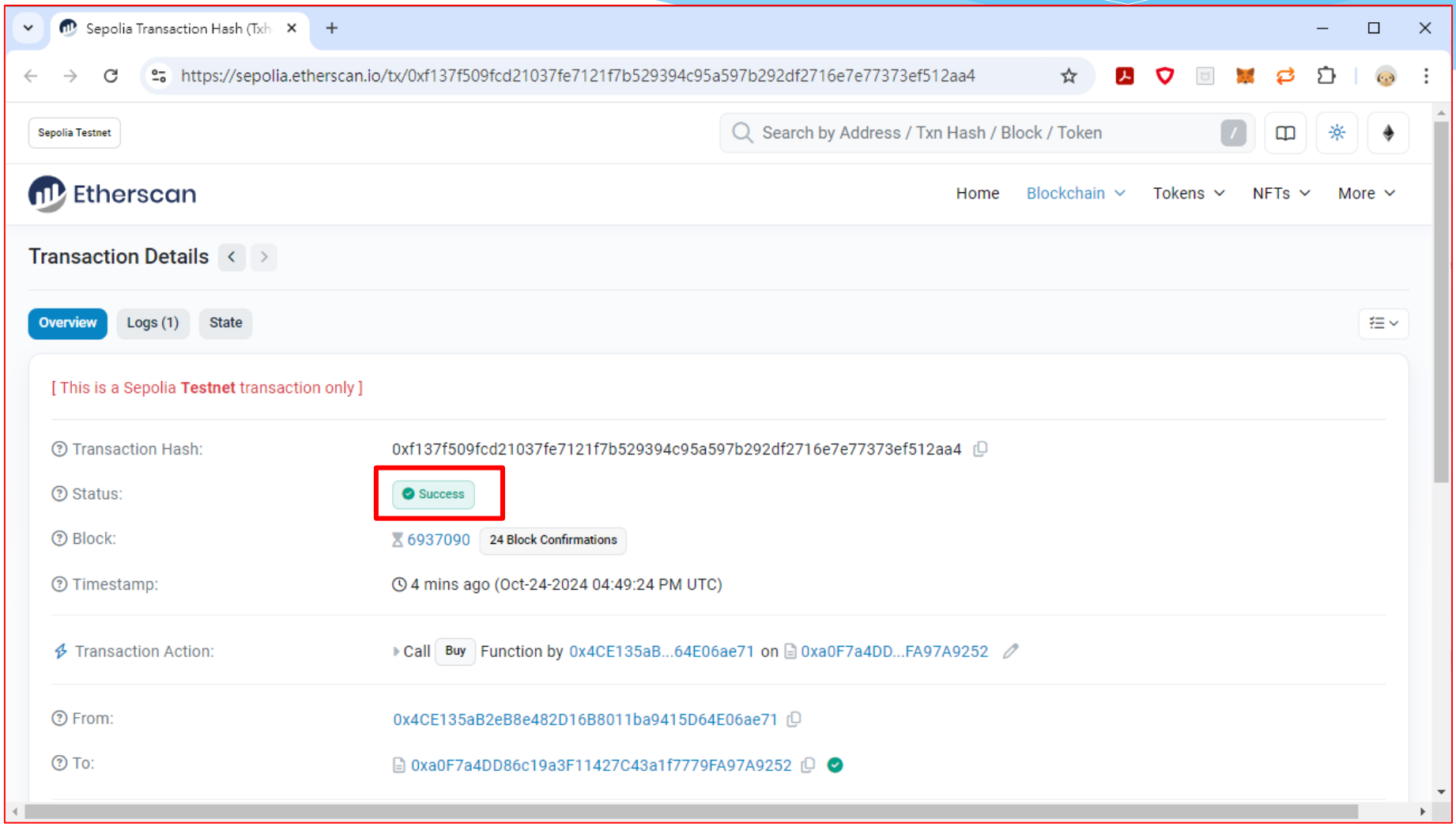
Write Contract - Buy



https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#writeContract

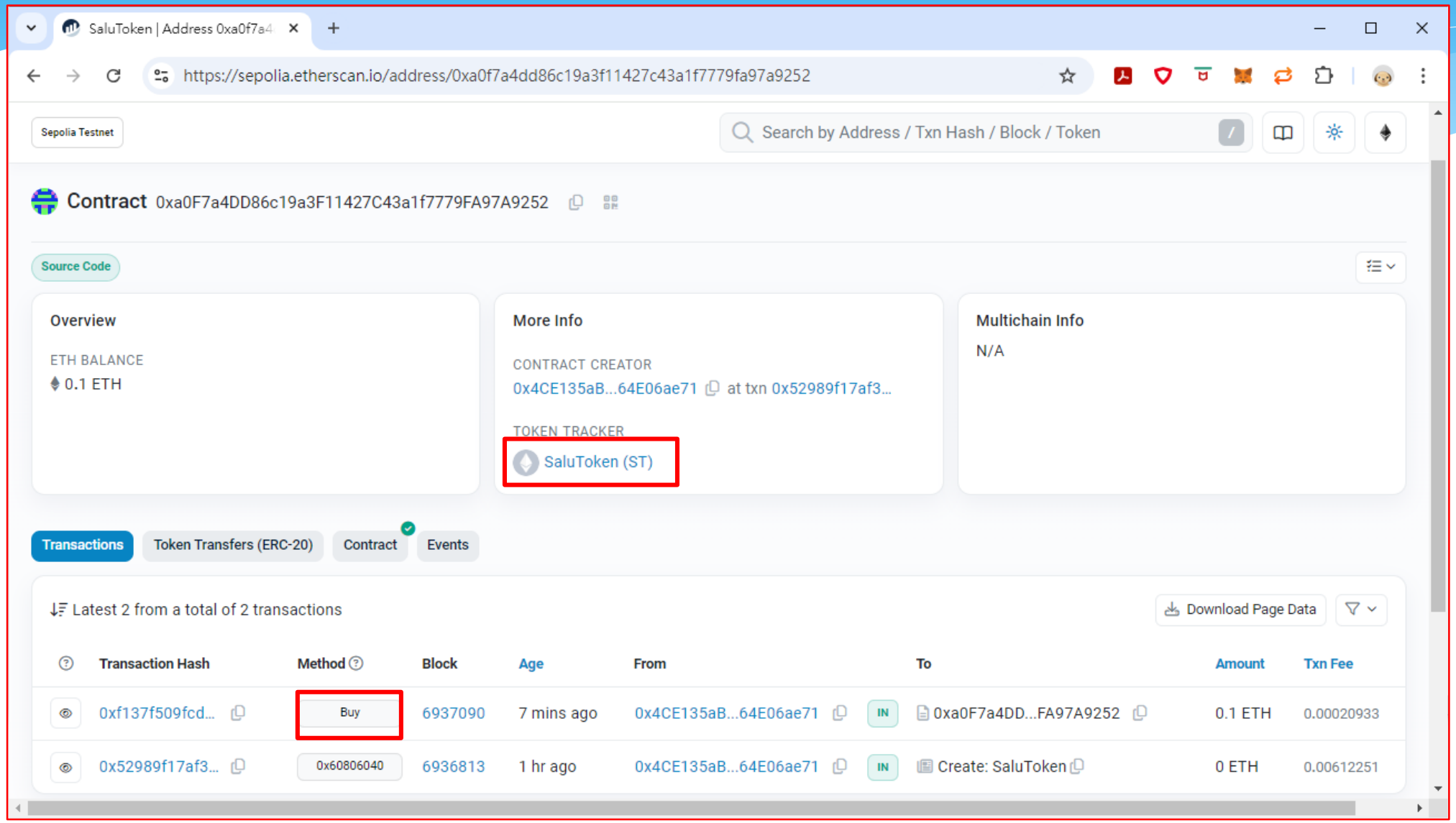
Write Contract - Buy

Write Contract - Buy



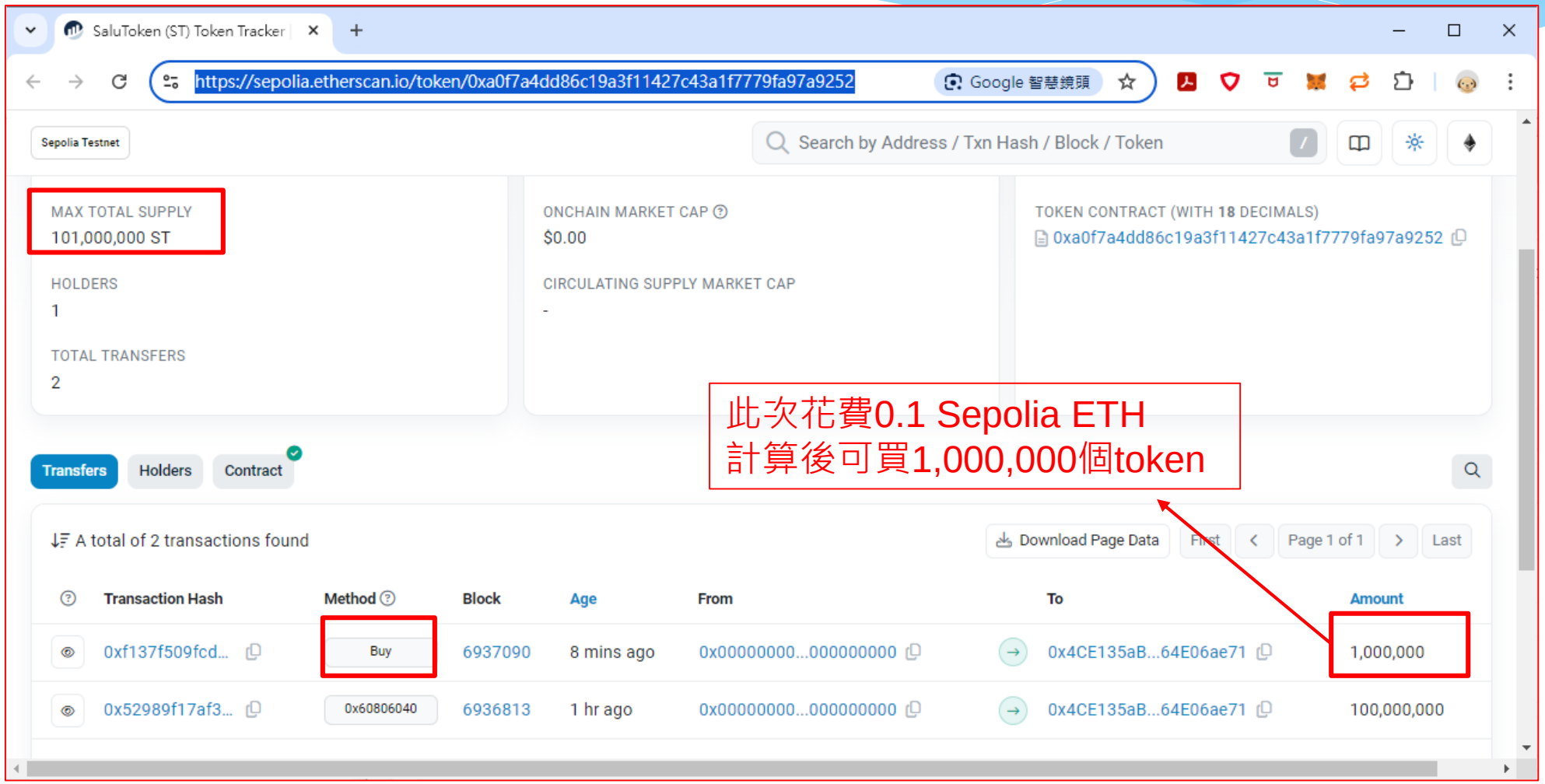
https://sepolia.etherscan.io/tx/0xf137f509fcd21037fe7121f7b529394c95a597b292df2716e7e77373ef512aa4

Write Contract - Buy



https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252

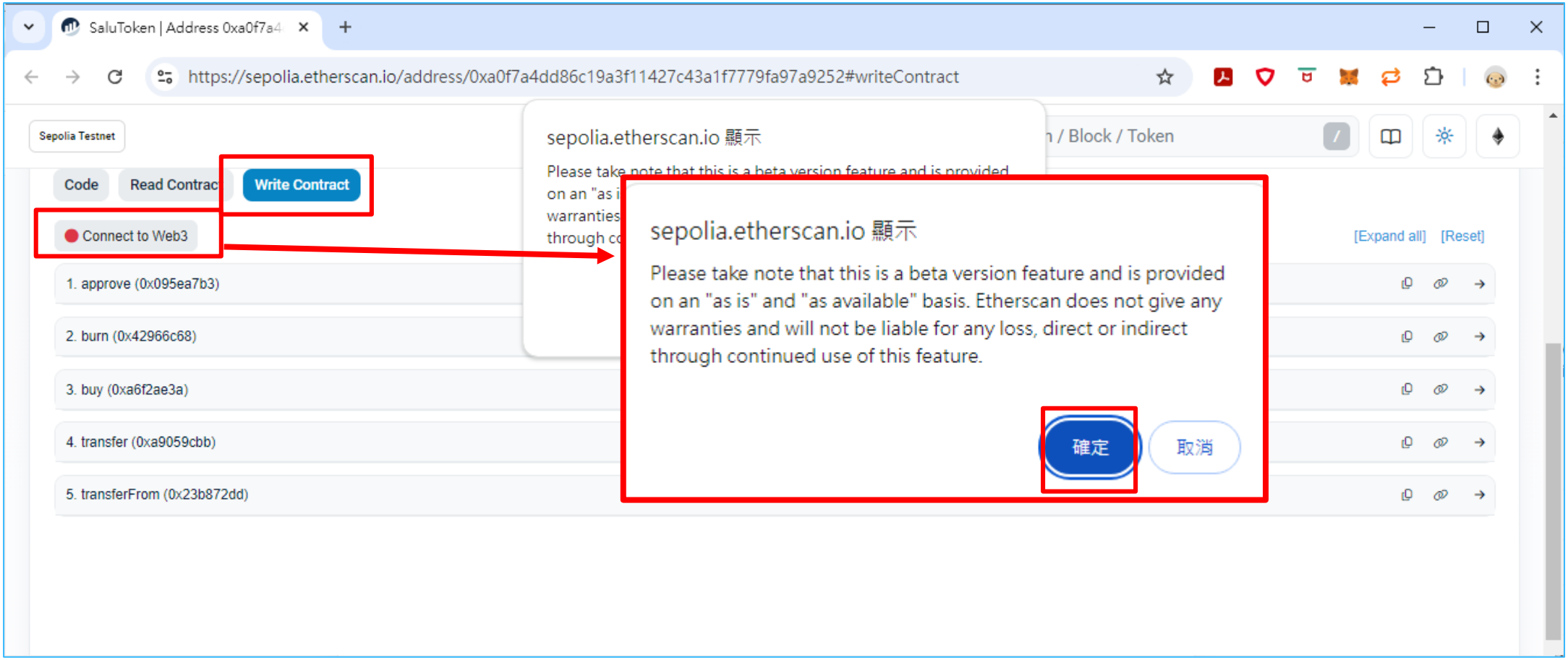
Write Contract - Buy



https://sepolia.etherscan.io/token/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252

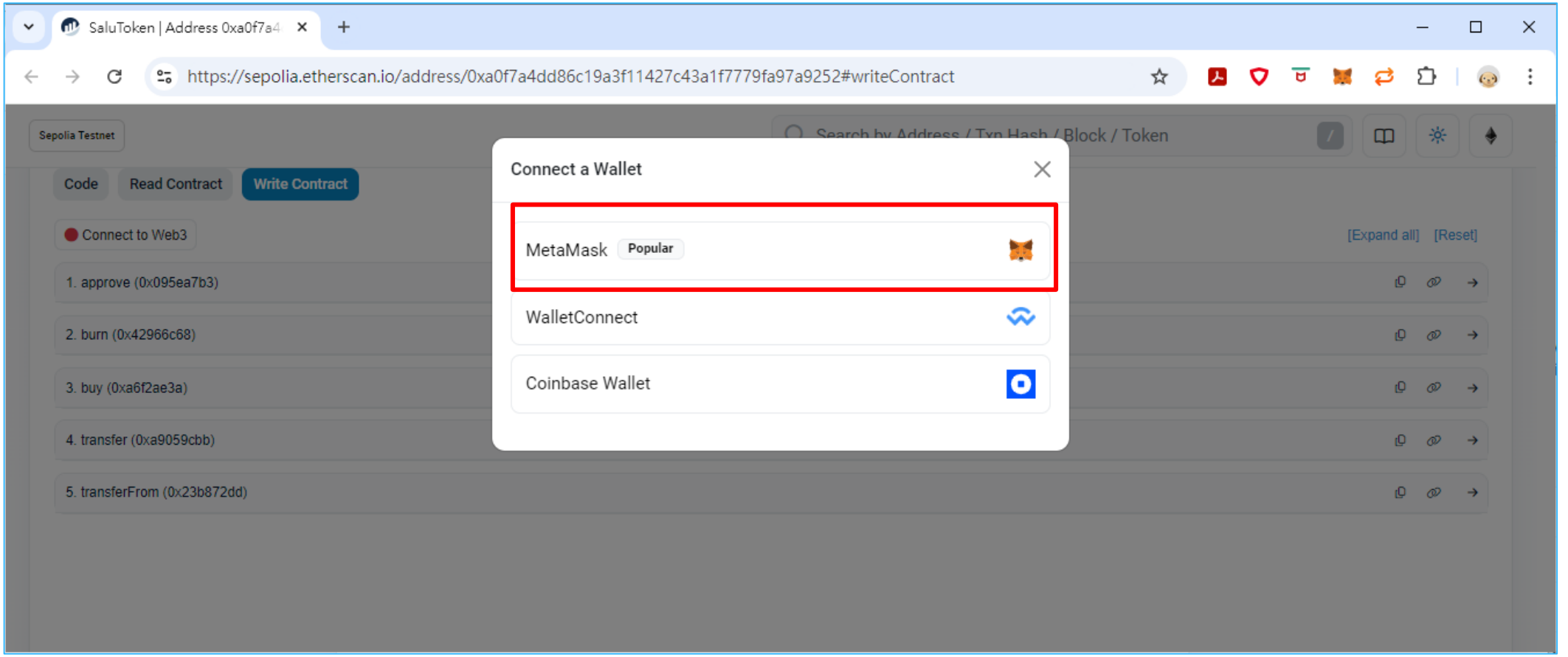
Burn Token

Connect to Web3



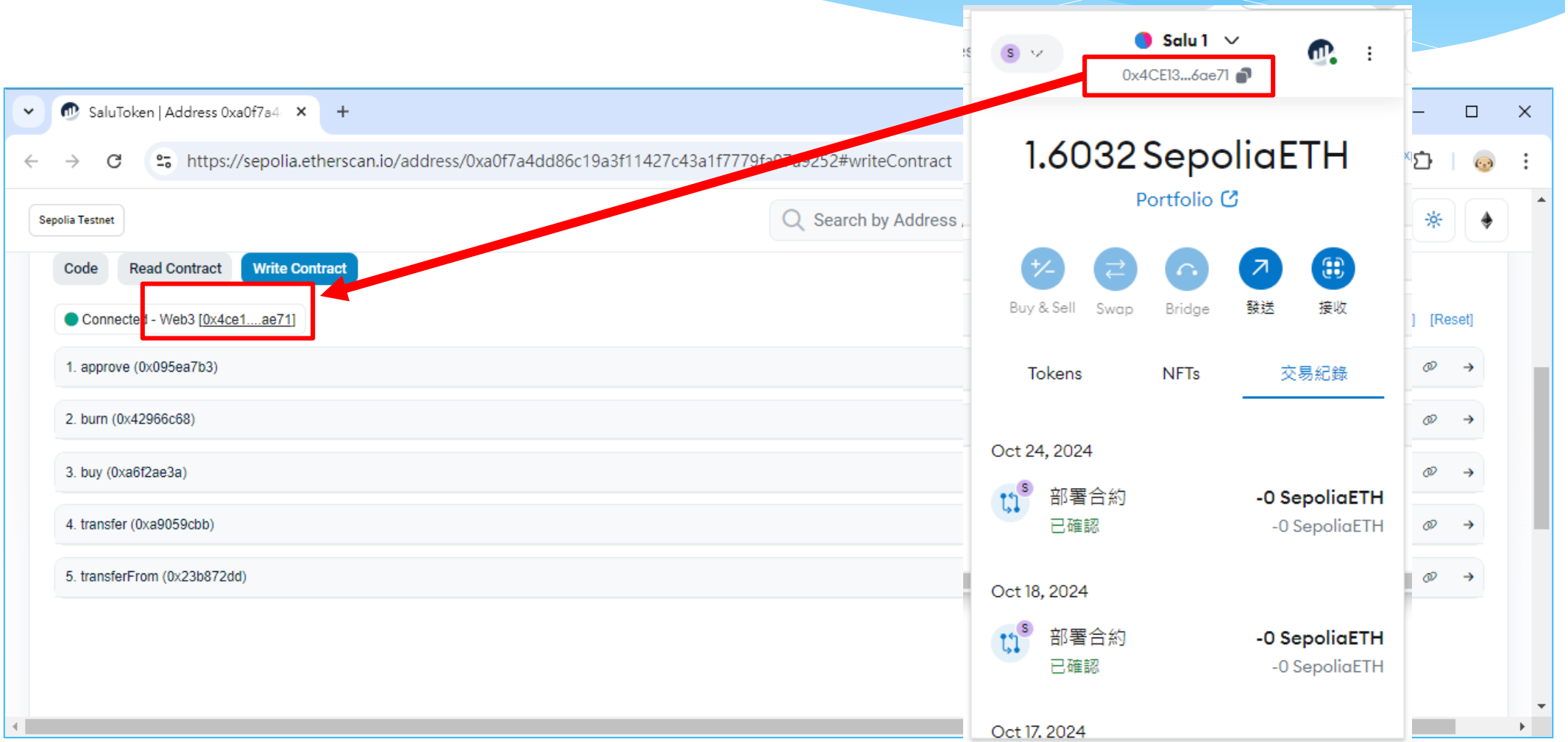
https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#writeContract

Connect a Wallet-MetaMask



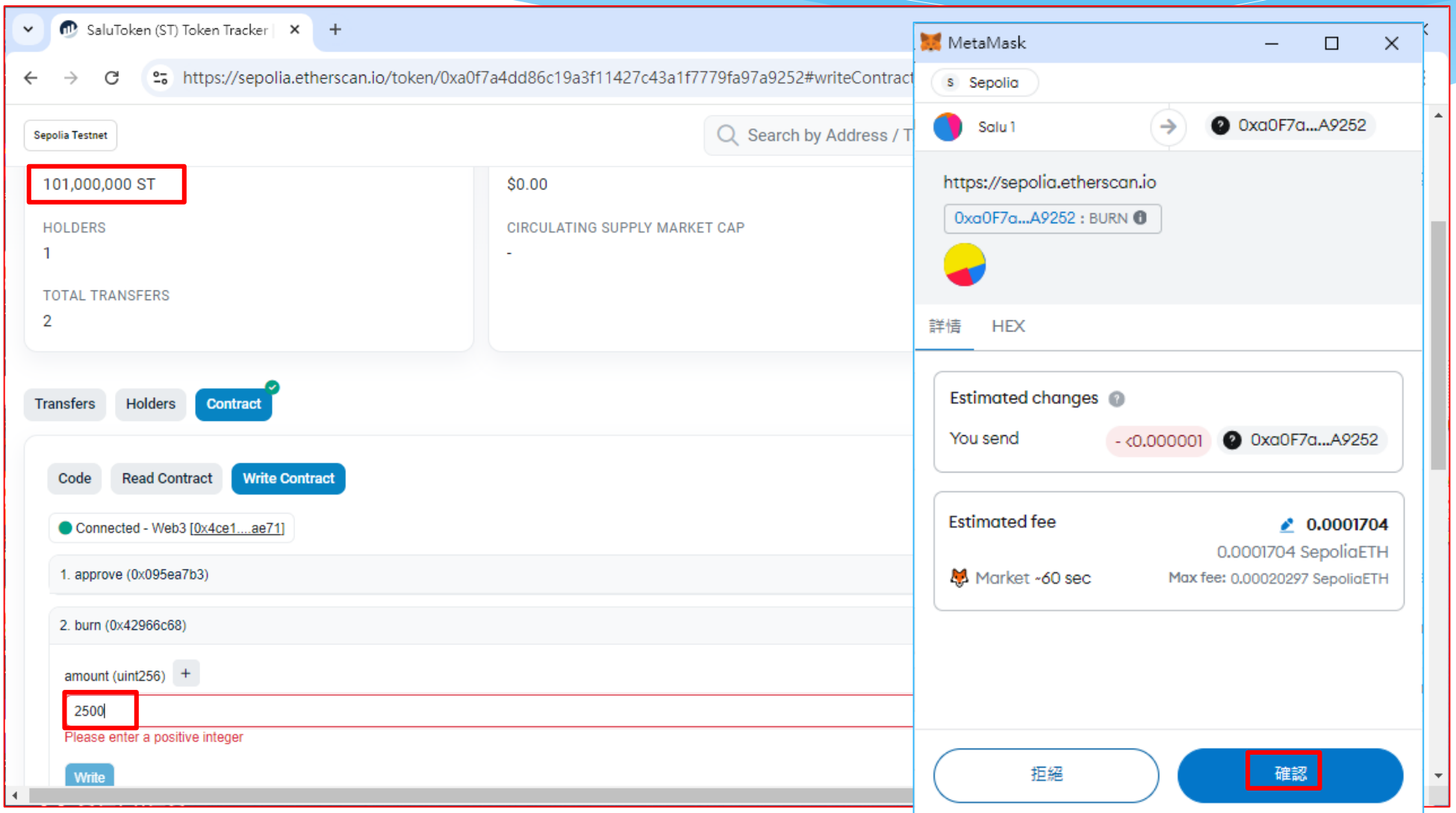
https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#writeContract

Connected Web3 using account



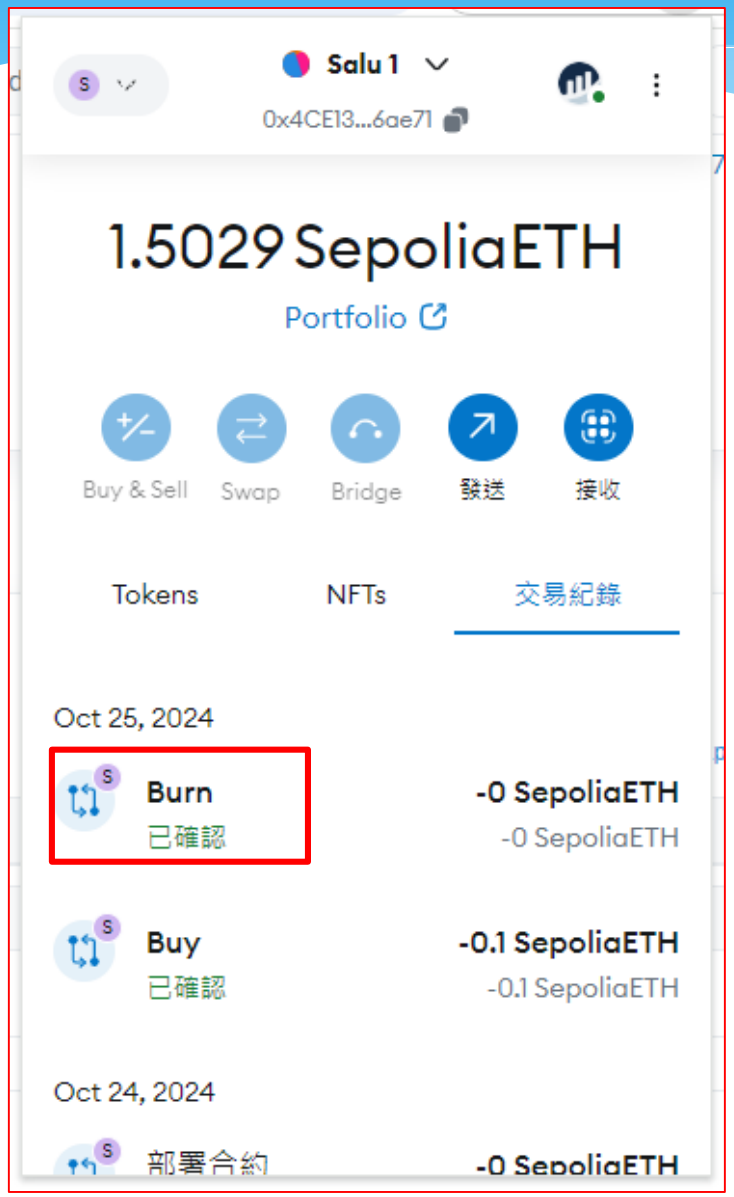
https://sepolia.etherscan.io/token/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#writeContract

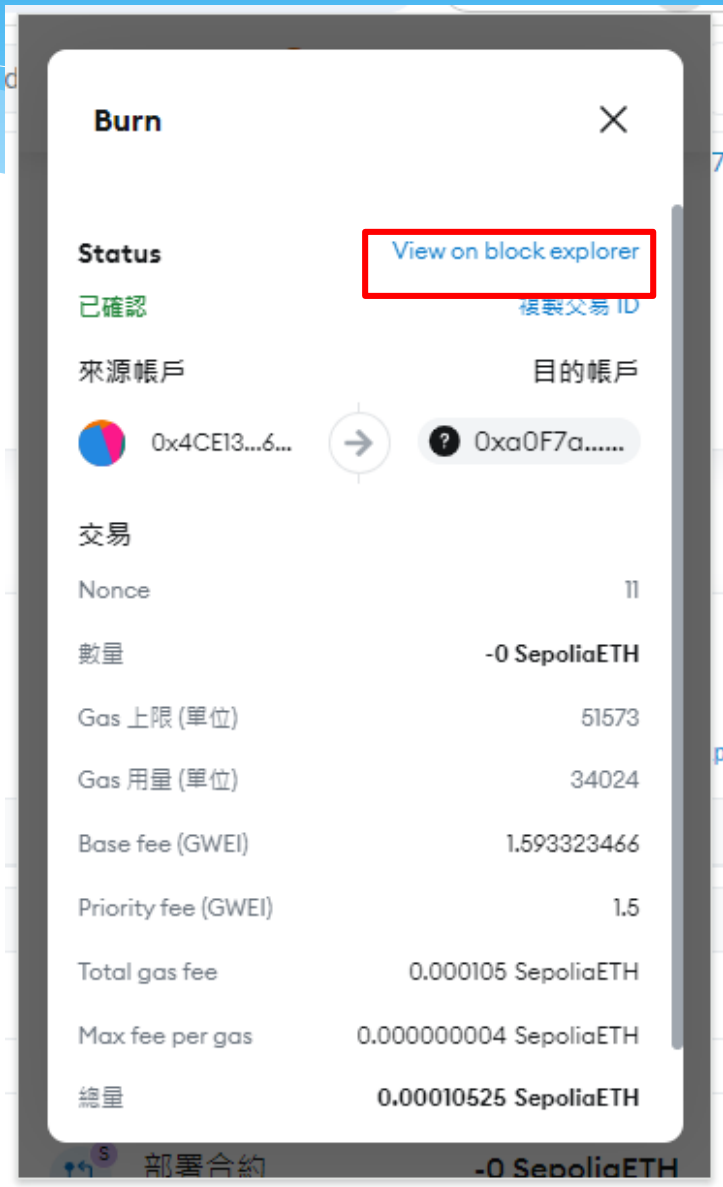
Write Contract - burn



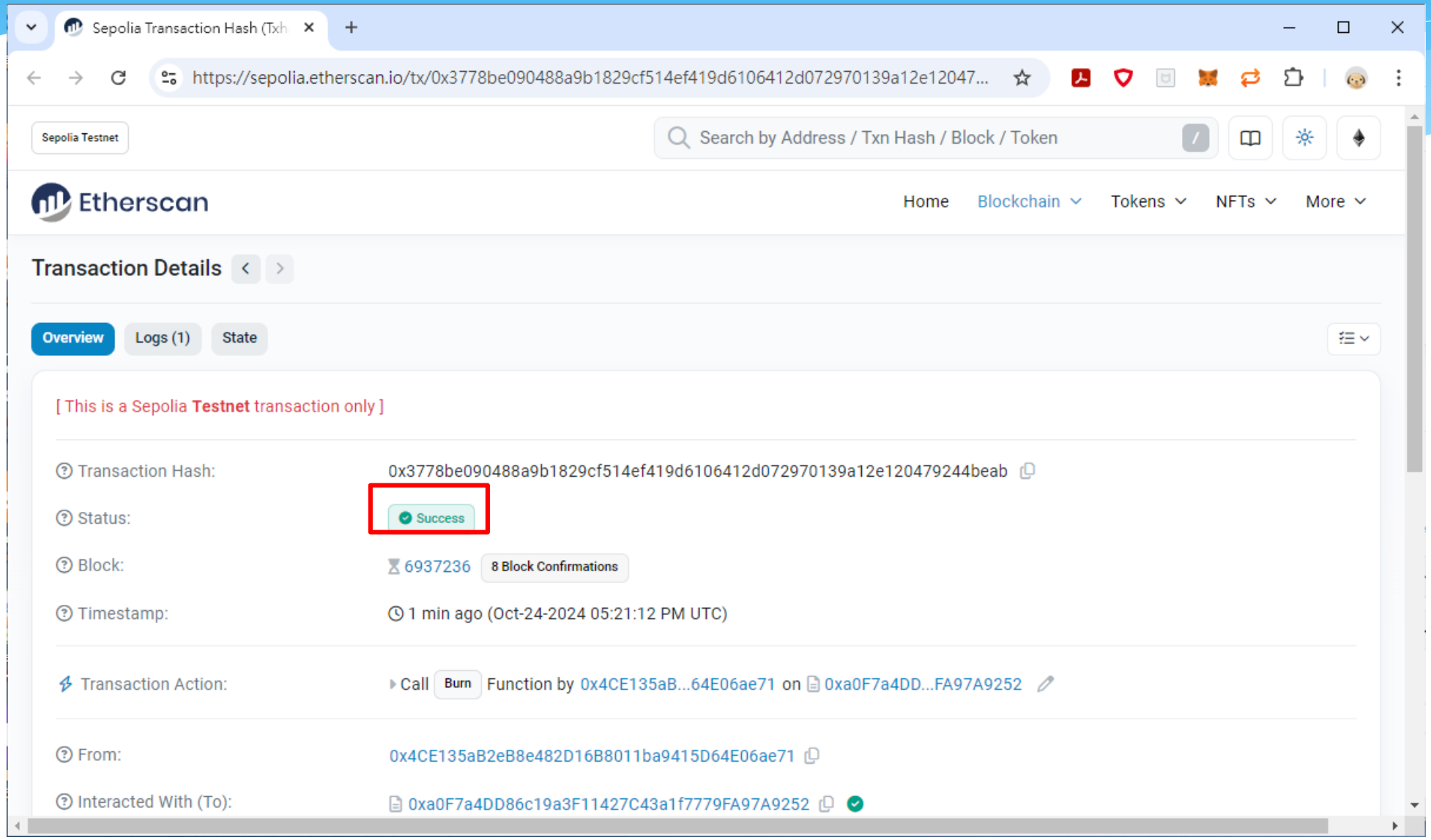
https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252#writeContract

Write Contract - Burn



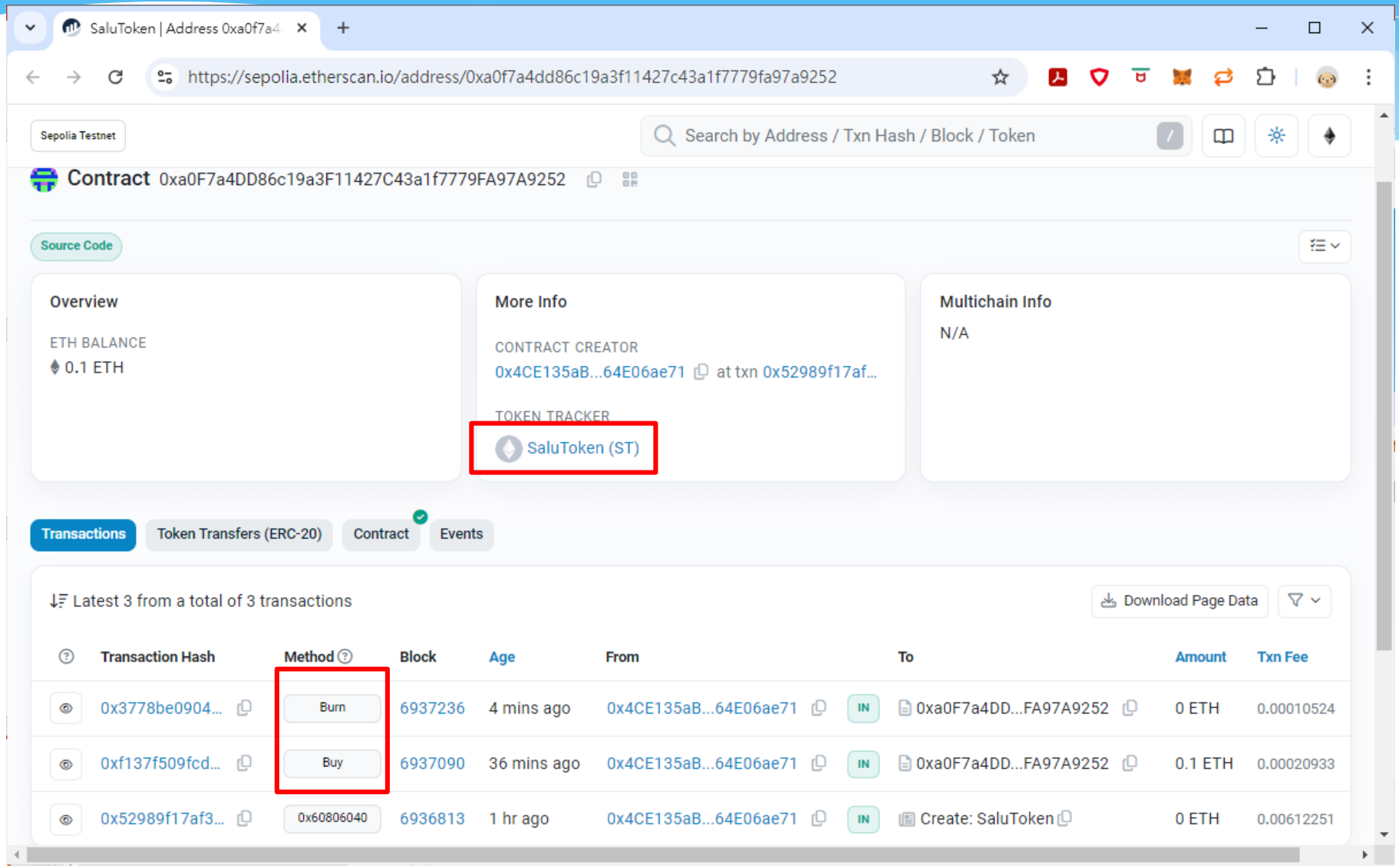


Write Contract - Burn



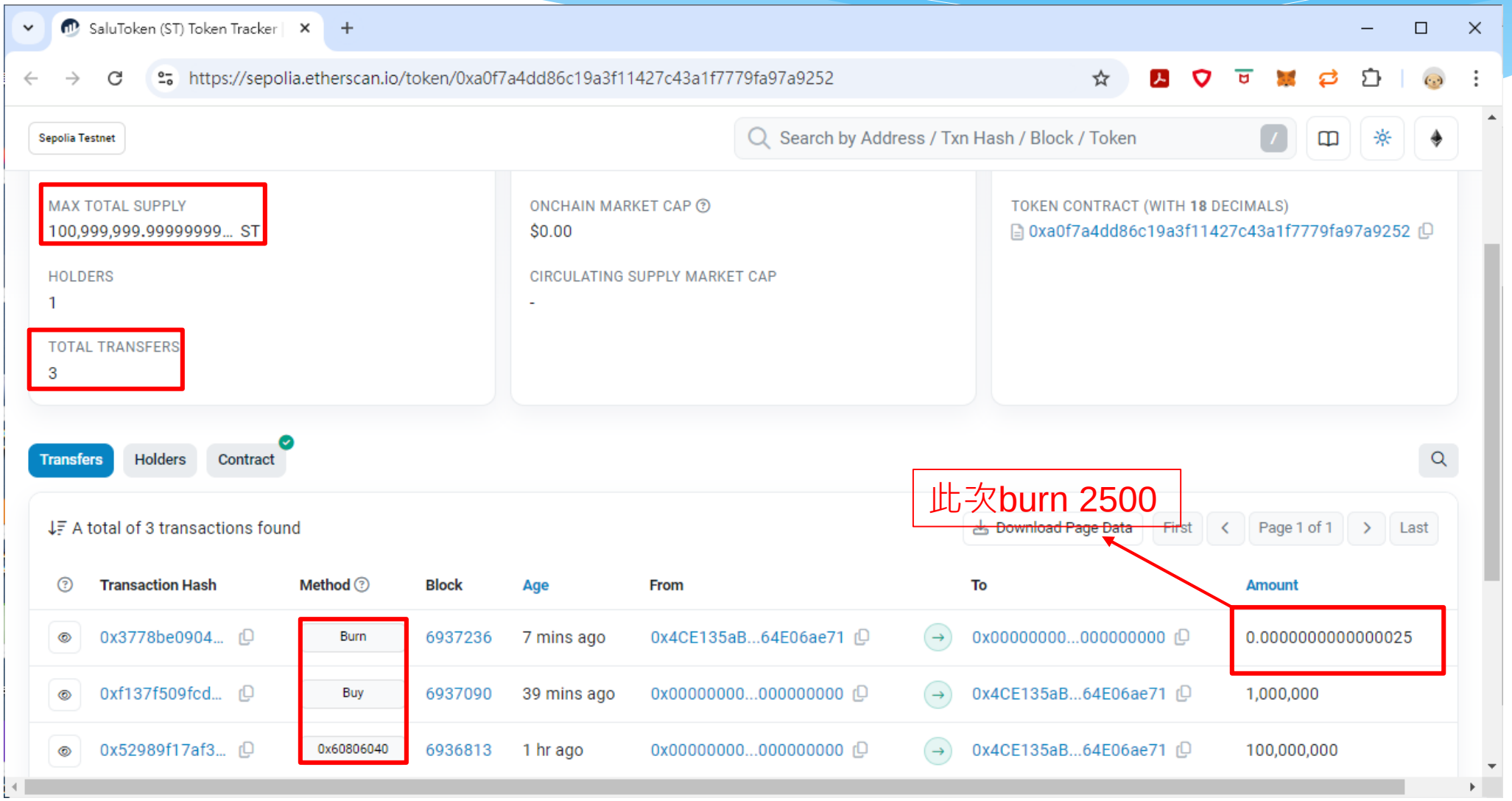
https://sepolia.etherscan.io/tx/0x3778be090488a9b1829cf514ef419d6106412d072970139a12e120479244beab

Write Contract - Burn



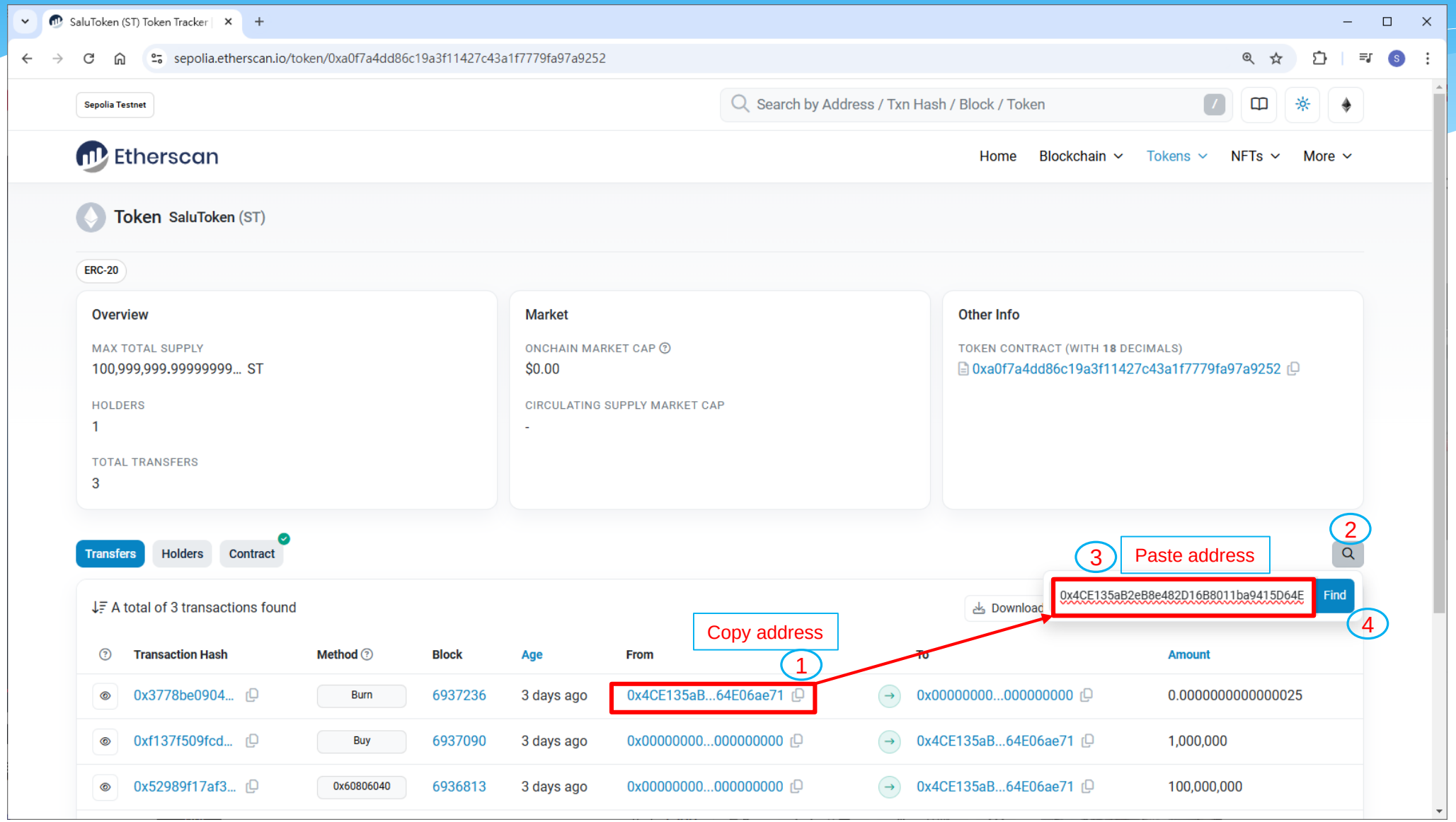
https://sepolia.etherscan.io/address/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252

Write Contract - Burn



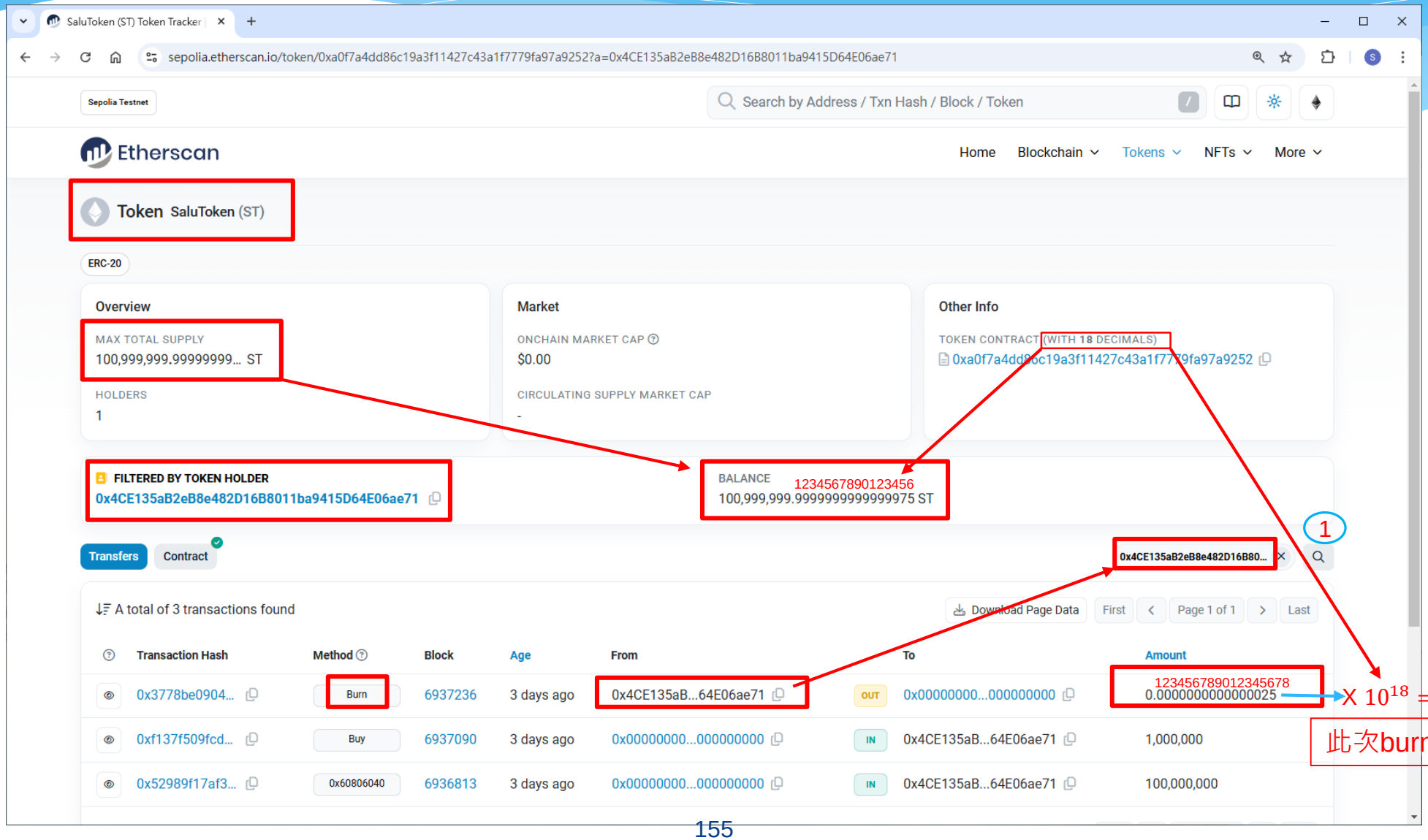
https://sepolia.etherscan.io/token/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252

Write Contract - Burn



https://sepolia.etherscan.io/token/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252

Write Contract - Burn



https://sepolia.etherscan.io/token/0xa0f7a4dd86c19a3f11427c43a1f7779fa97a9252?a=0x4CE135aB2eB8e482D16B8011ba9415D64E06ae71

Security and Access Control

Security and Access Control

- Access control, or “who is allowed to do this thing”, is incredibly important when dealing with smart contracts. The access control of your contract may **govern who can mint tokens, vote on proposals, freeze transfers, and many other things.**
- It is therefore **critical** to understand how you implement it, lest someone else **steals your whole system**.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts@4.9.0/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts@4.9.0/access/Ownable.sol";

contract SaluToken is ERC20, Ownable{
    constructor() ERC20("SaluToken", "ST") {}

    function issueToken() public onlyOwner{
        _mint(msg.sender, 1000 * (10 ** 18));
    }
}
```

使用v4.9.0版本

1. The most common and basic form of access control is the concept of *ownership* where there’s only an account allowed to perform sensitive tasks on a contract (the owner). This approach is perfectly reasonable for contracts that have a single administrative user.
2. OpenZeppelin provides Ownable for implementing ownership in contracts.

By default, the owner of an **Ownable contract** is the account that deployed it, which is usually exactly what you want.

onlyOwner will run every time issueToken() gets called, verifying if the caller is the owner of the contract.

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v4.9/contracts/access/Ownable.sol

Security and Access Control

- Access control, or “who is allowed to do this thing”, is incredibly important when dealing with smart contracts. The access control of your contract may **govern who can mint tokens, vote on proposals, freeze transfers, and many other things.**
- It is therefore **critical** to understand how you implement it, lest someone else **steals your whole system**.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";


contract SaluToken is ERC20,   Ownable(address(msg.sender)) {
    constructor() ERC20("SaluToken", "ST") {}

    function issueToken() public onlyOwner{
        _mint(msg.sender, 1000 * (10 ** 18));
    }
}
```

預設使用目前最新版本v5.1.0

1. The most common and basic form of access control is the concept of *ownership* where there’s only an account allowed to perform sensitive tasks on a contract (the owner). This approach is perfectly reasonable for contracts that have a single administrative user.
2. OpenZeppelin provides Ownable for implementing ownership in contracts.

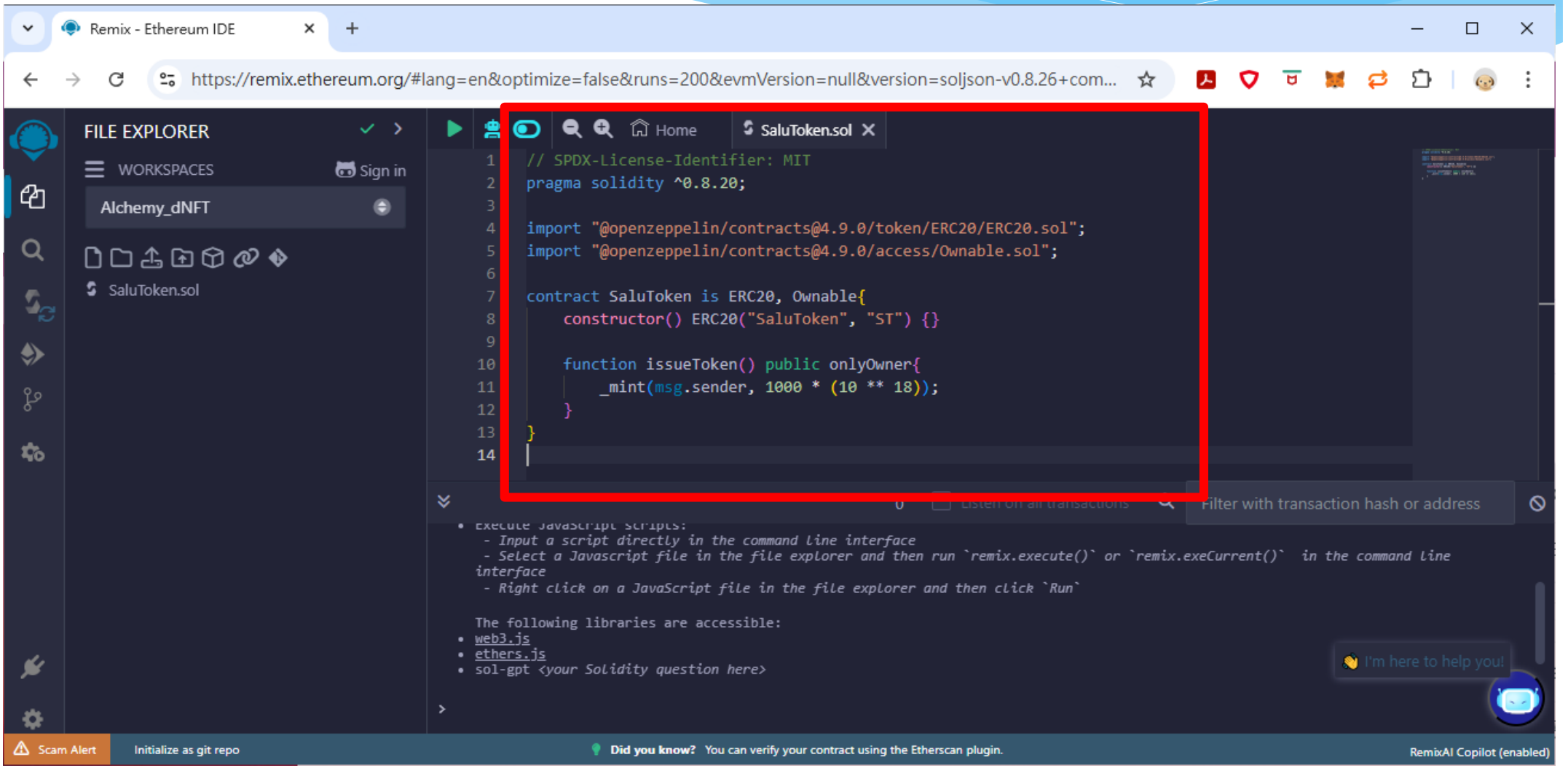
By default, the owner of an **Ownable contract** is the account that deployed it, which is usually exactly what you want.

onlyOwner will run every time issueToken() gets called, verifying if the caller is the owner of the contract.

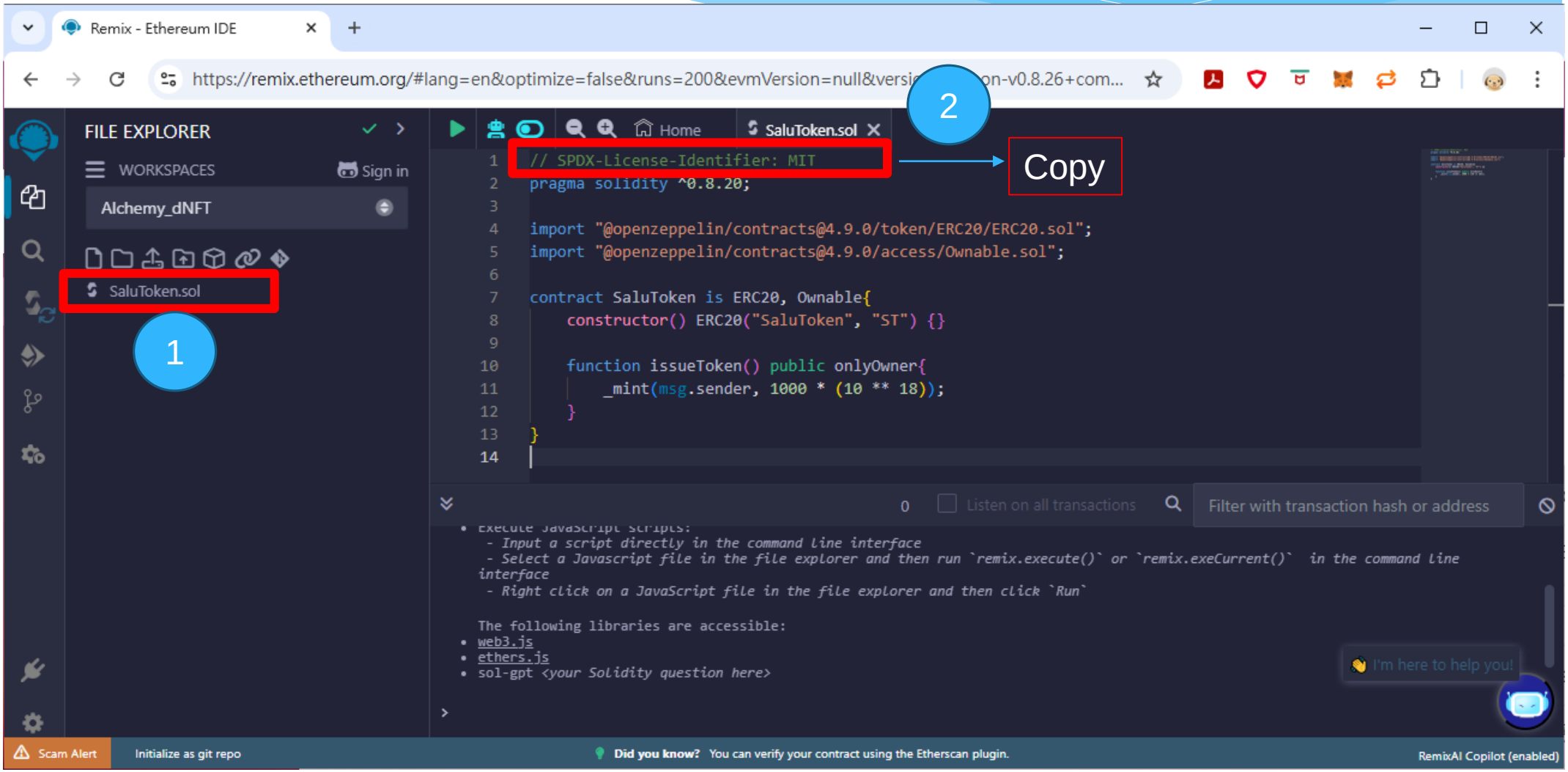
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v5.1/contracts/access/Ownable.sol

Openzeppelin v4.9.0

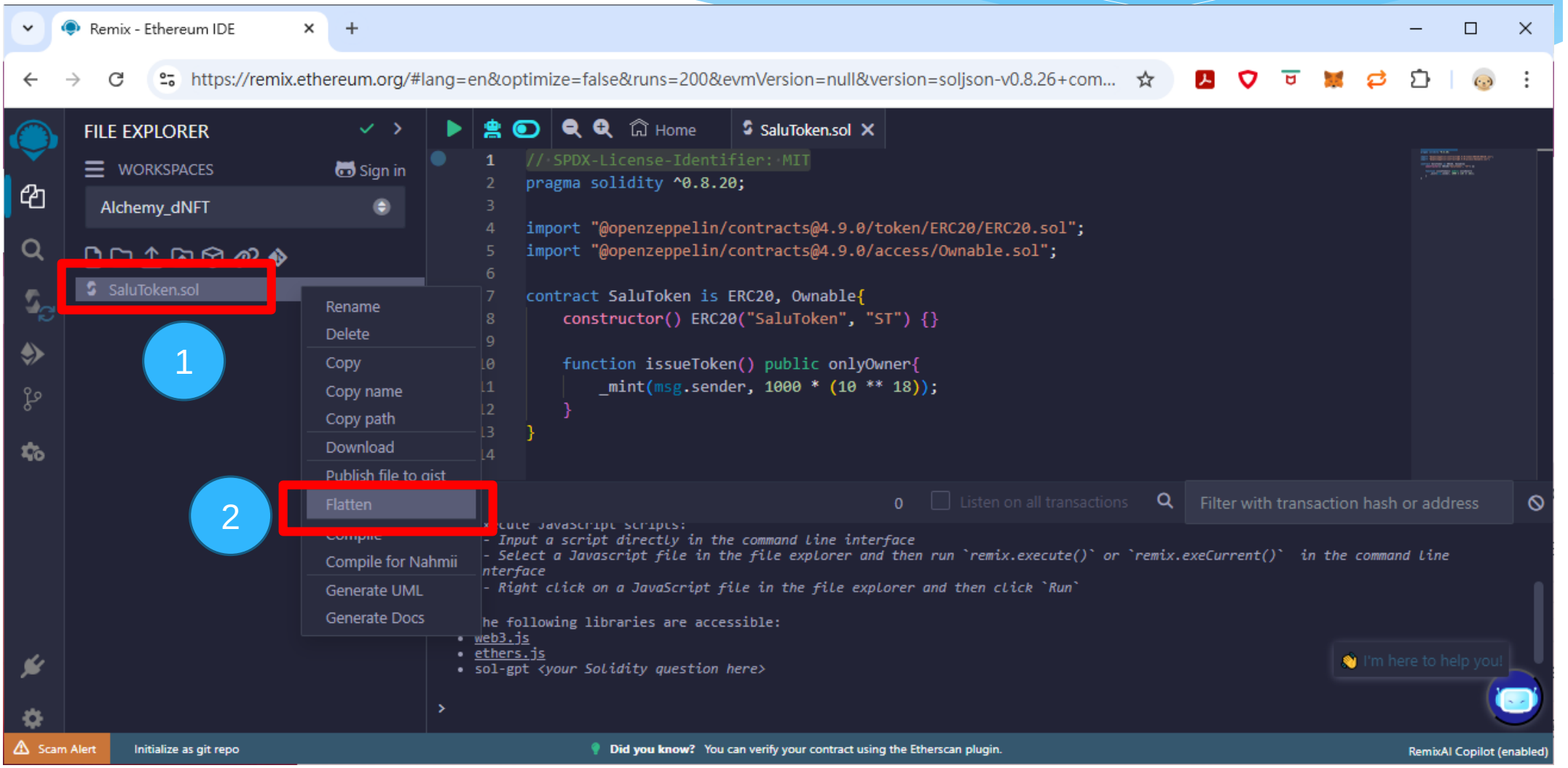
Modify SaluToken.sol



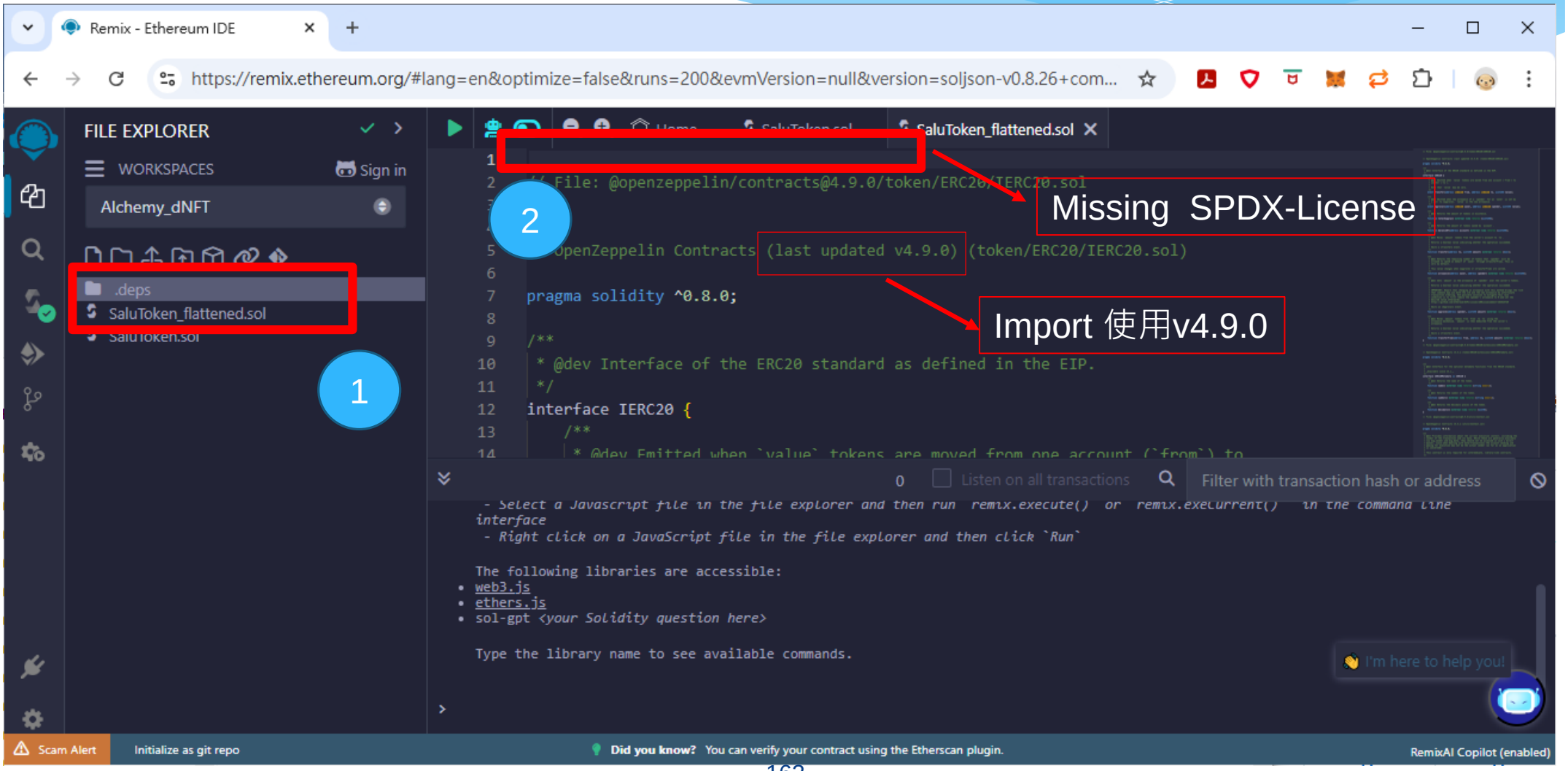
Copy SPDX-License



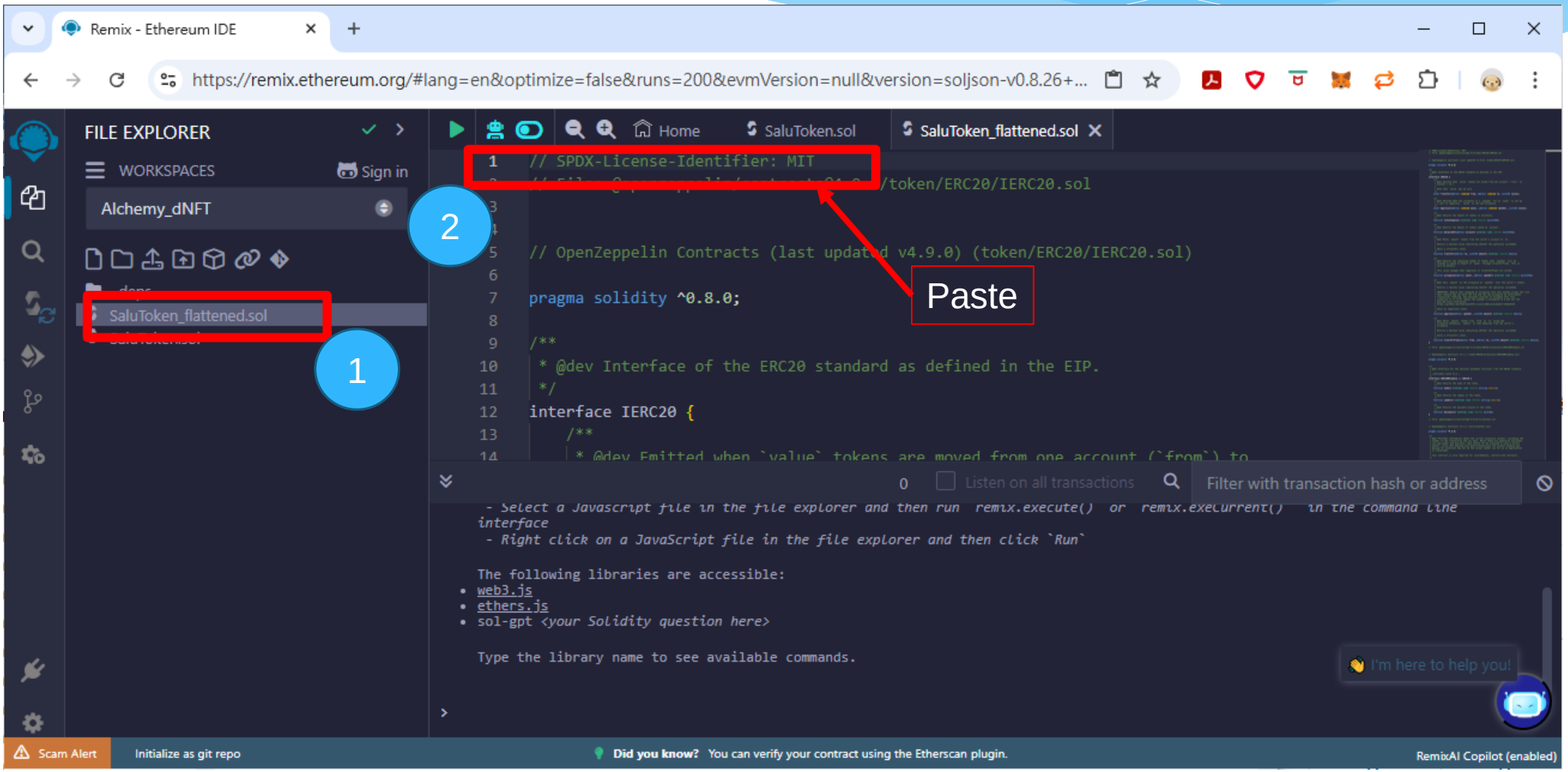
Flatten SaluToken.sol



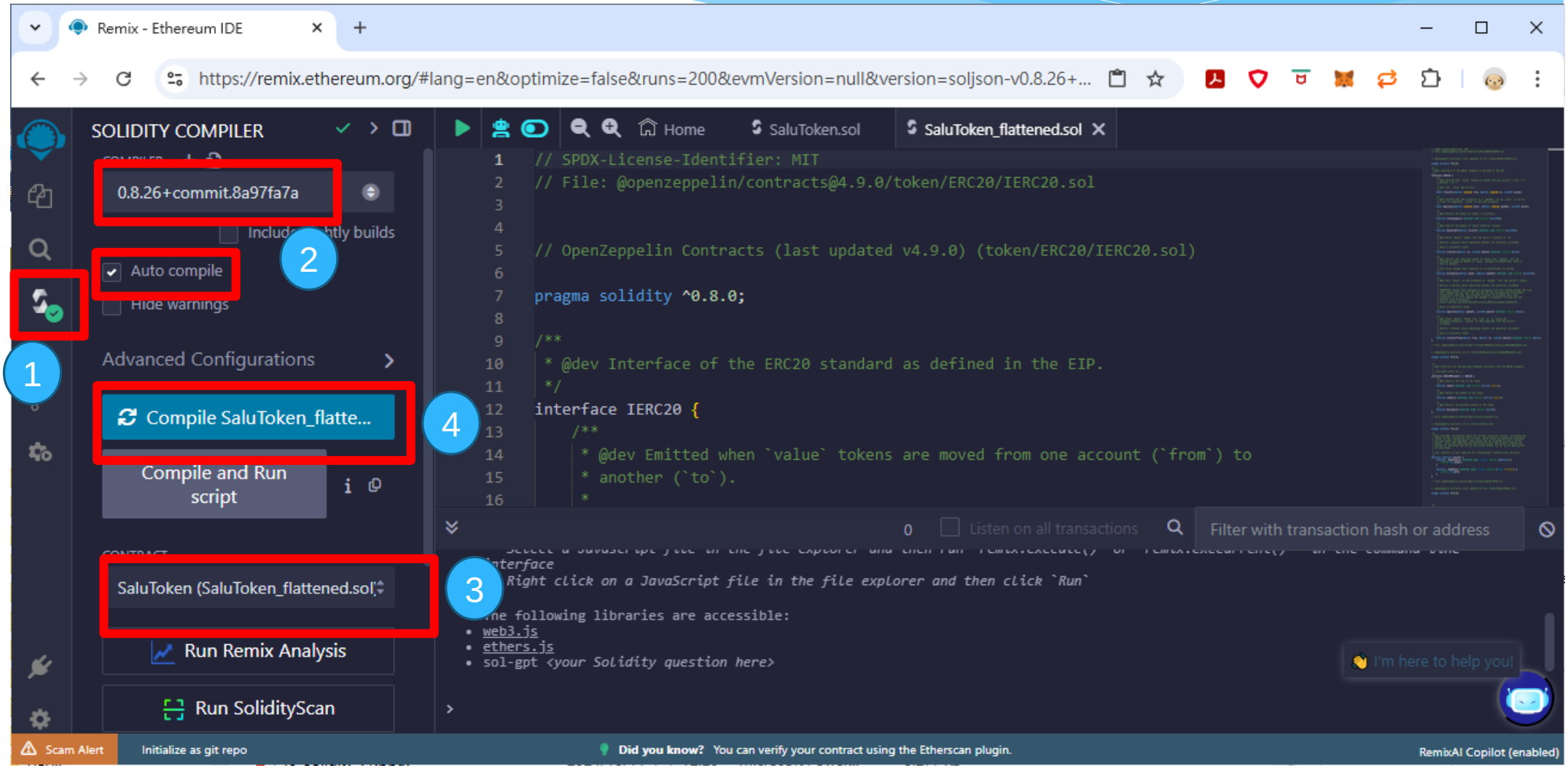
SaluToken_flattened.sol



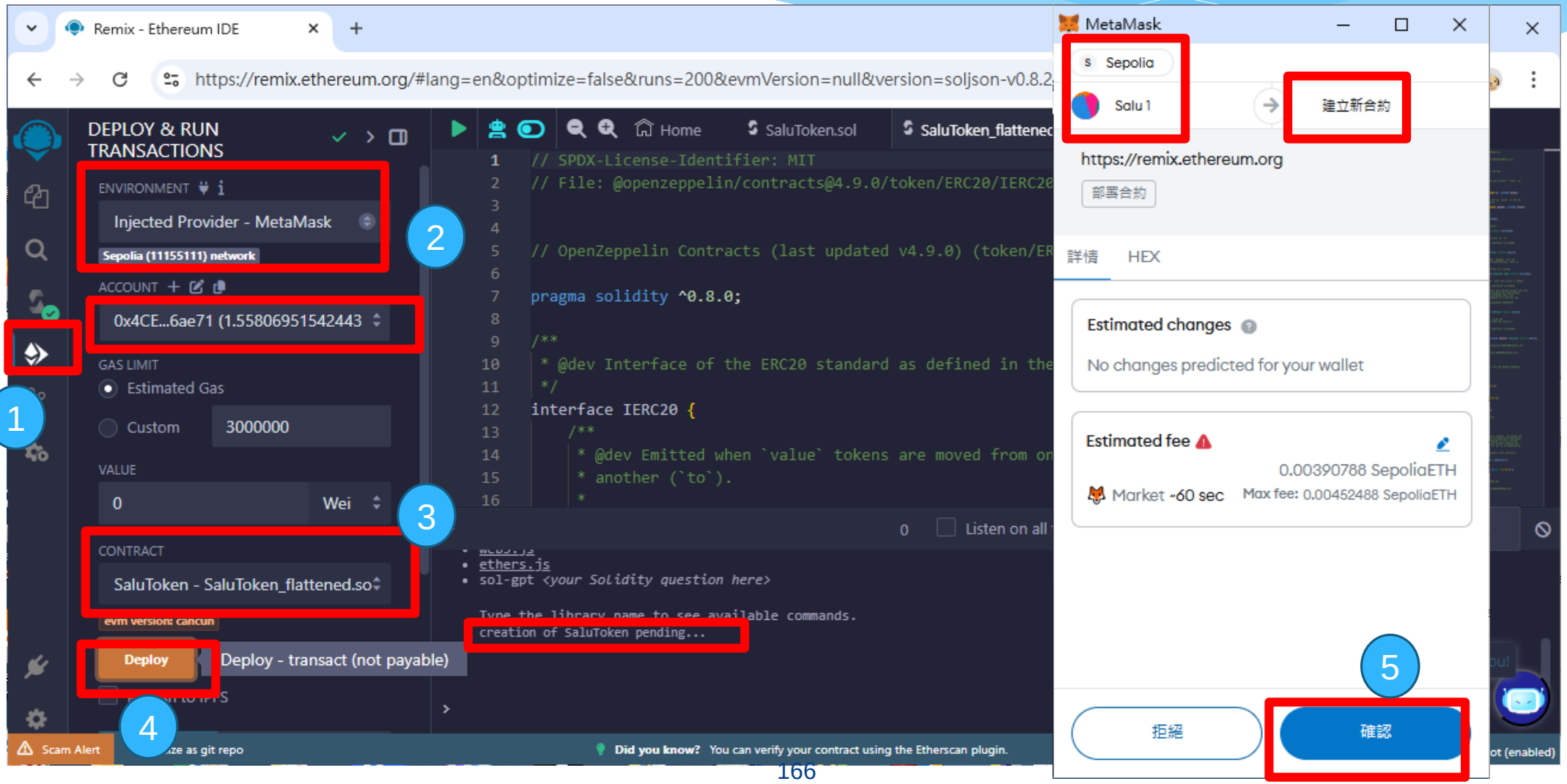
SaluToken_flattened.sol



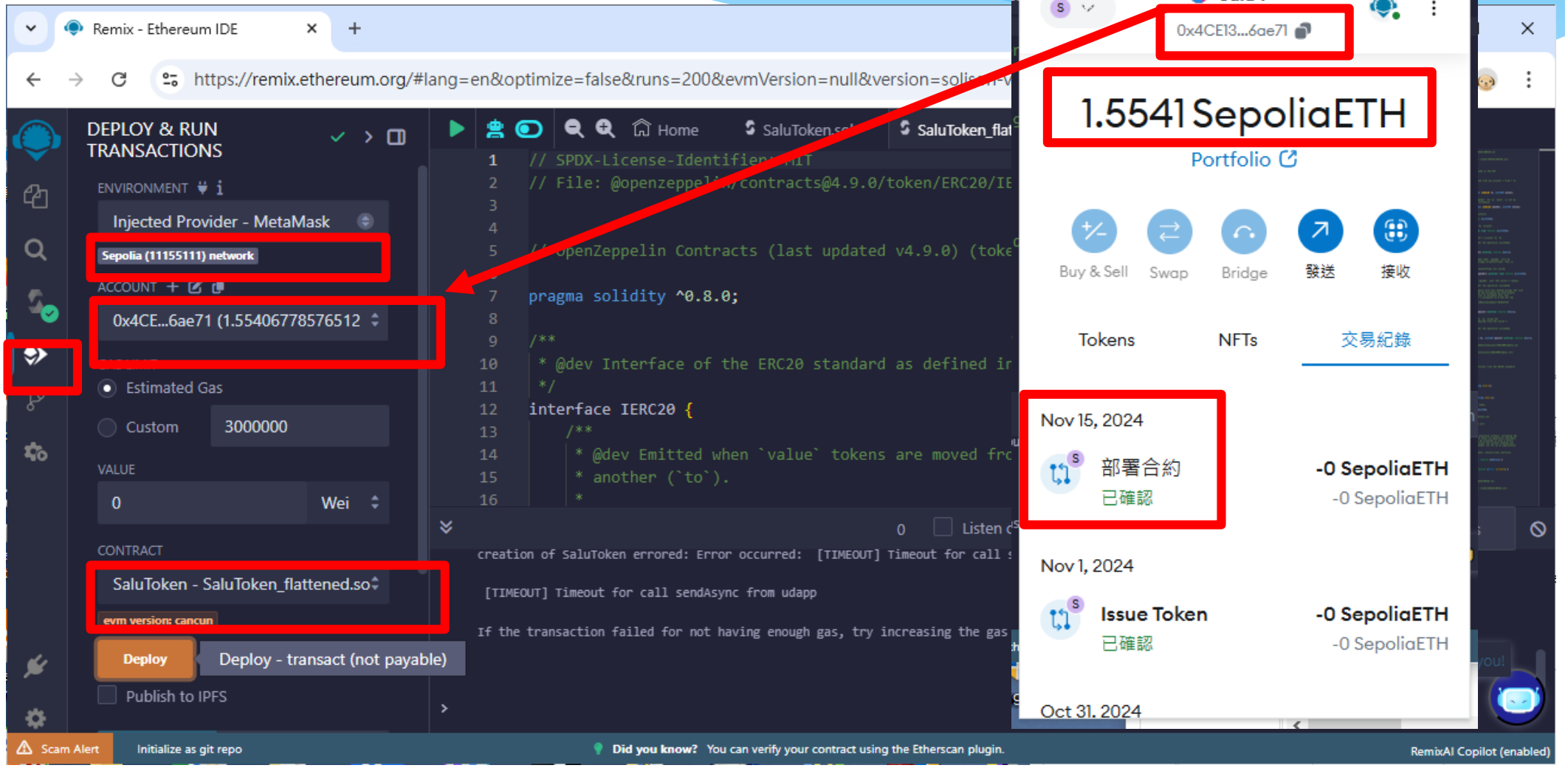
Compile Smart Contract



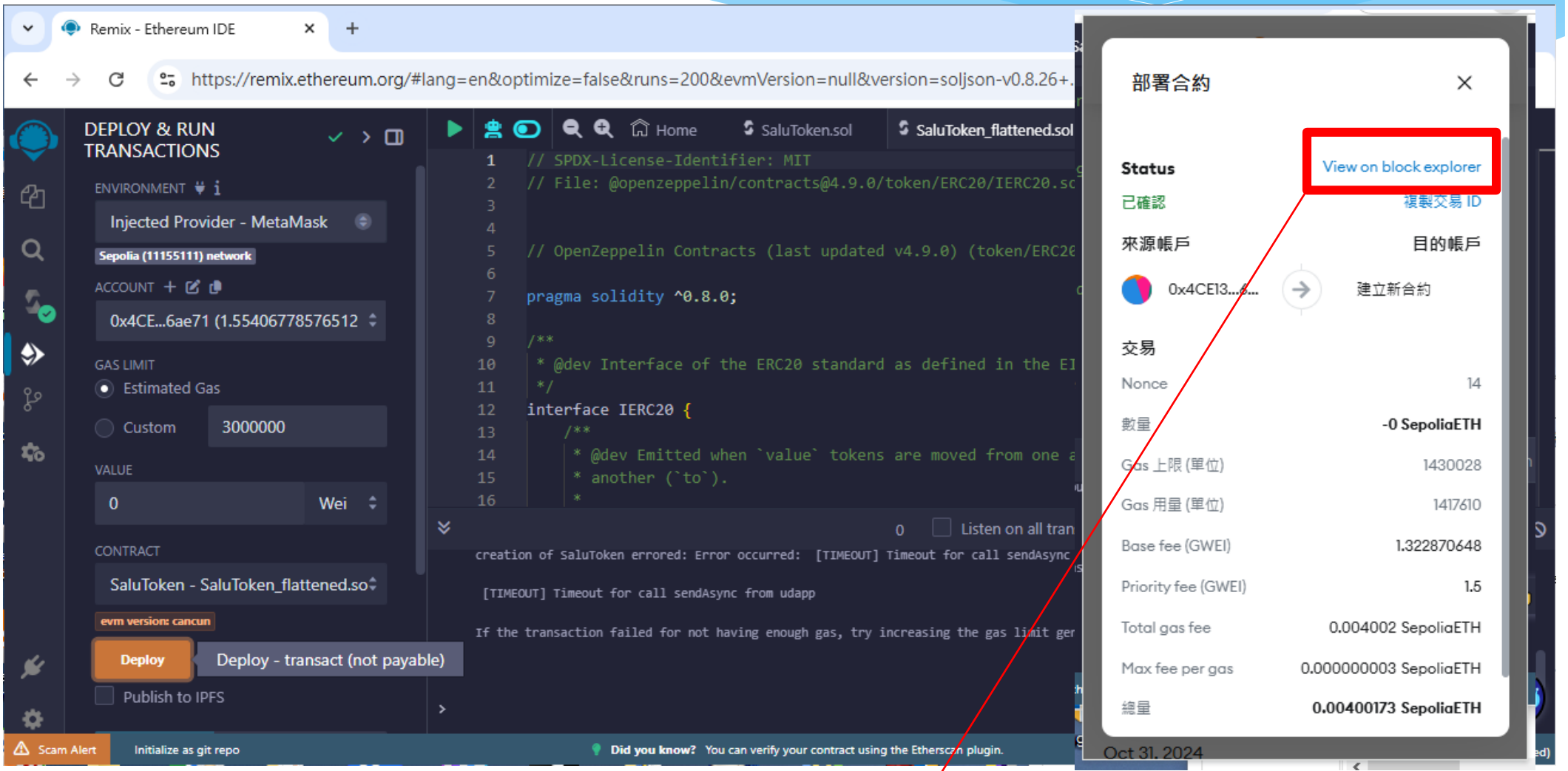
Deploy an ERC-20 Token



Deploy an ERC-20 Token

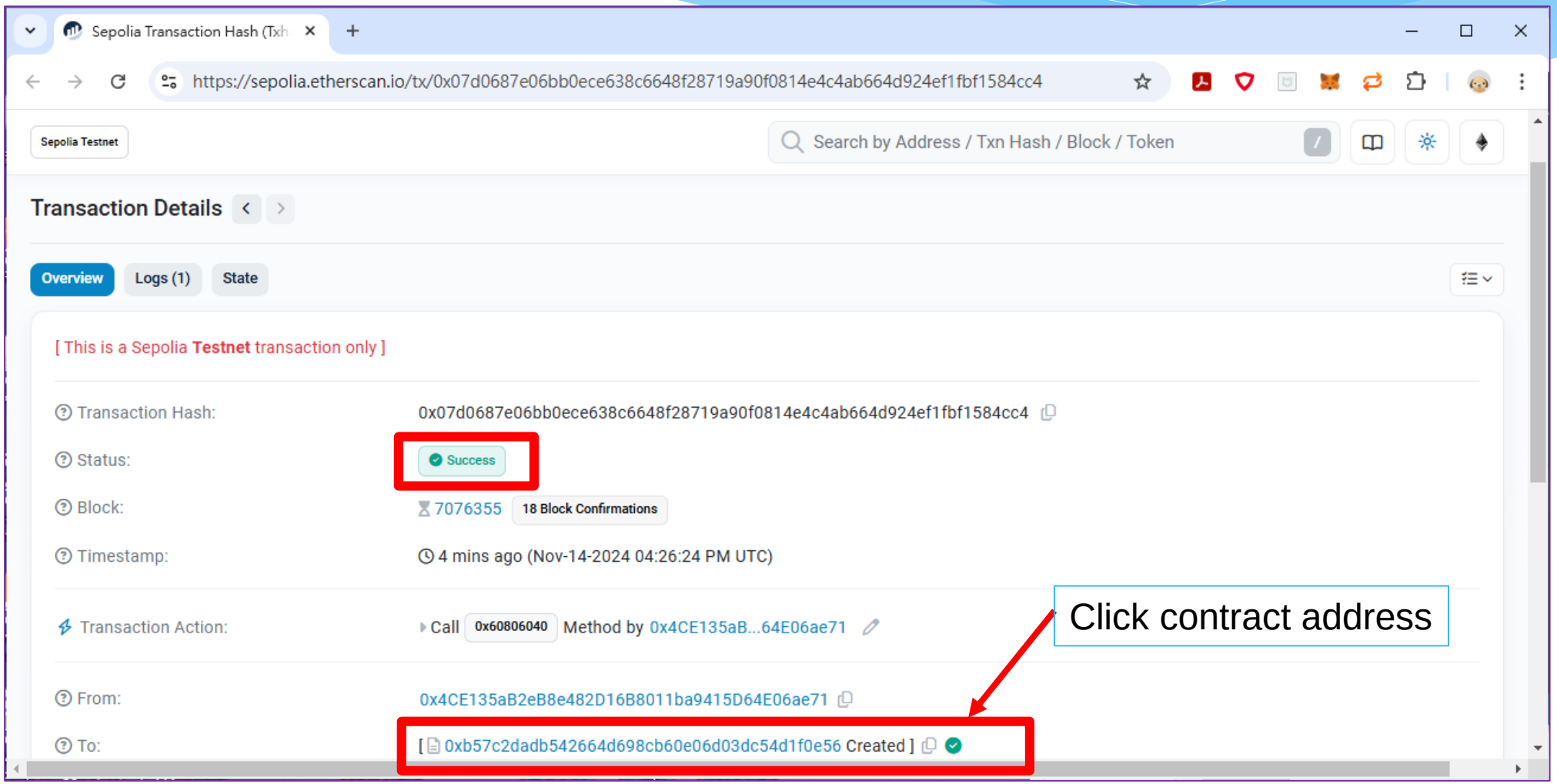


Deploy an ERC-20 Token



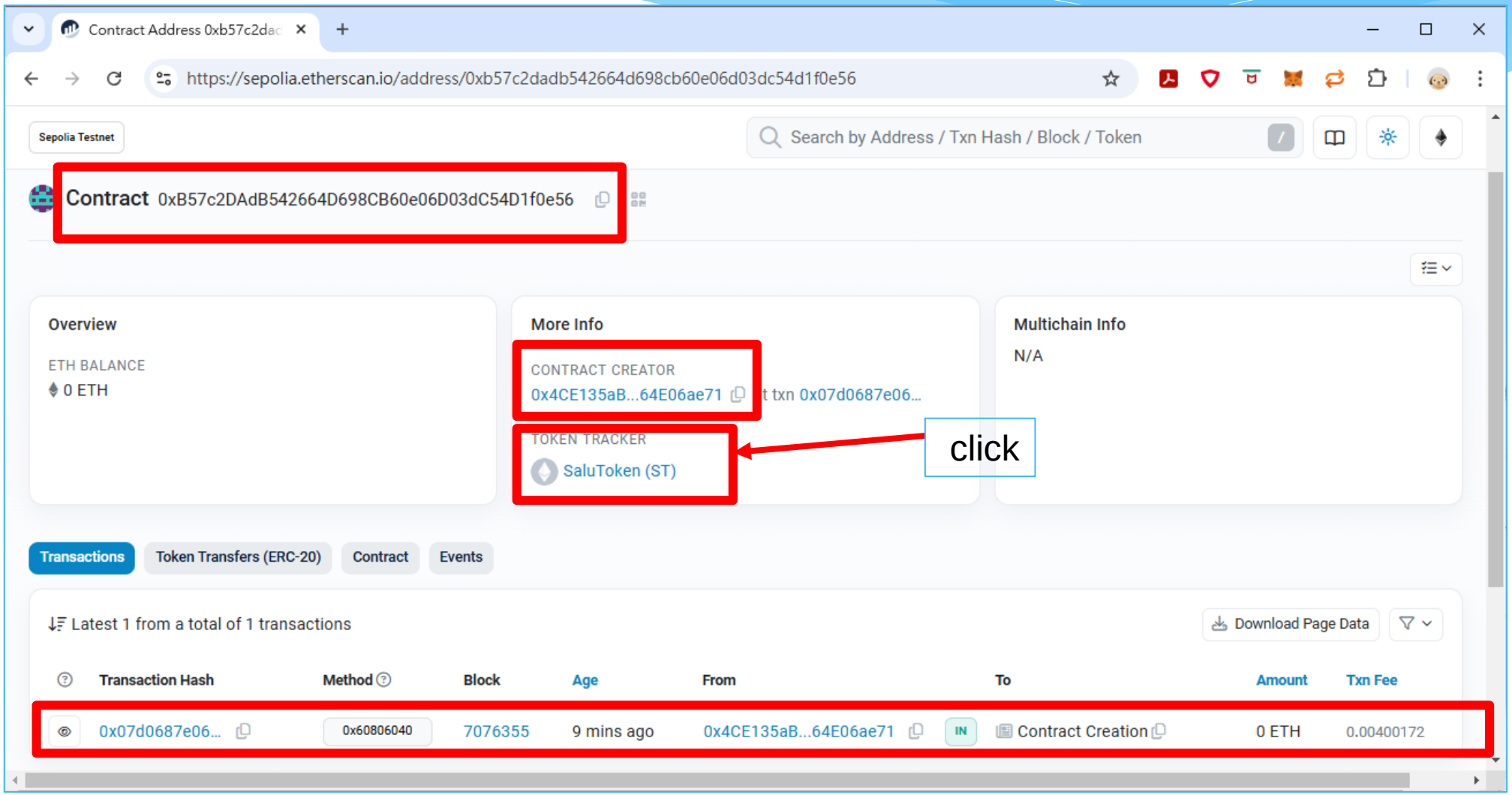
https://sepolia.etherscan.io/tx/0x07d0687e06bb0ece638c6648f28719a90f0814e4c4ab664d924ef1fbf1584cc4

View Transaction on Sepolia



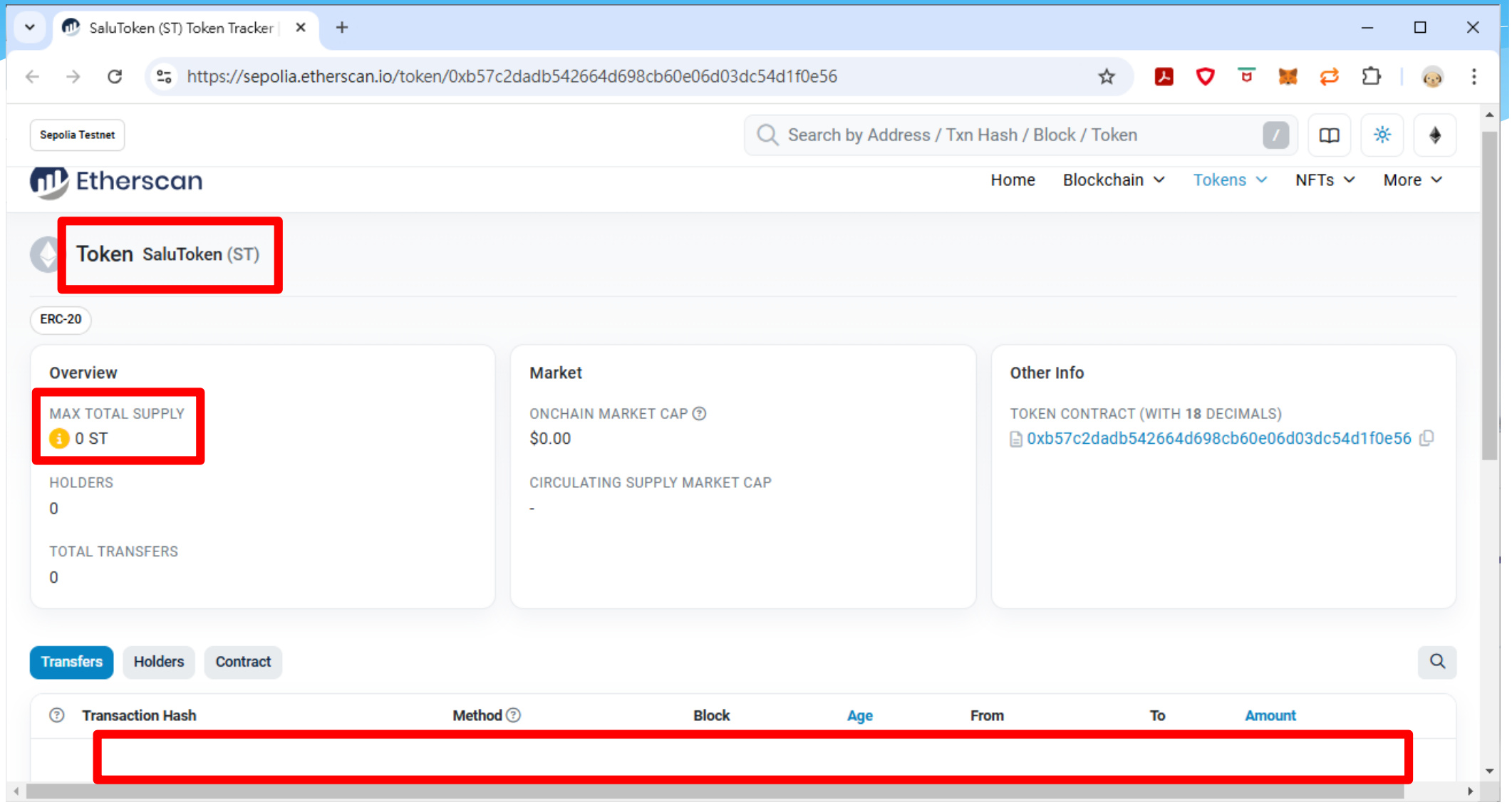
https://sepolia.etherscan.io/tx/0x07d0687e06bb0ece638c6648f28719a90f0814e4c4ab664d924ef1fbf1584cc4

Smart Contract Information



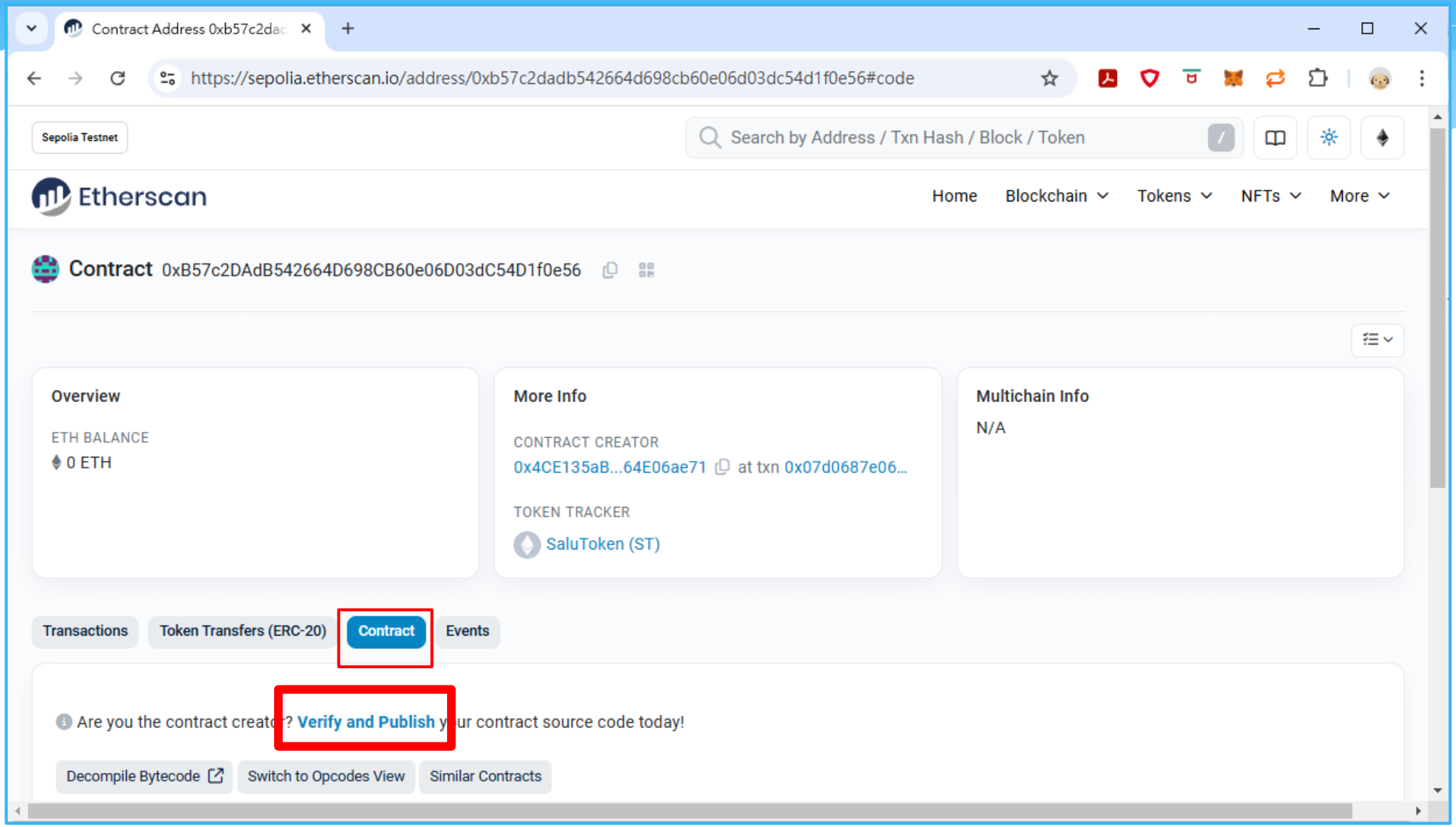
https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56

Smart Contract Information



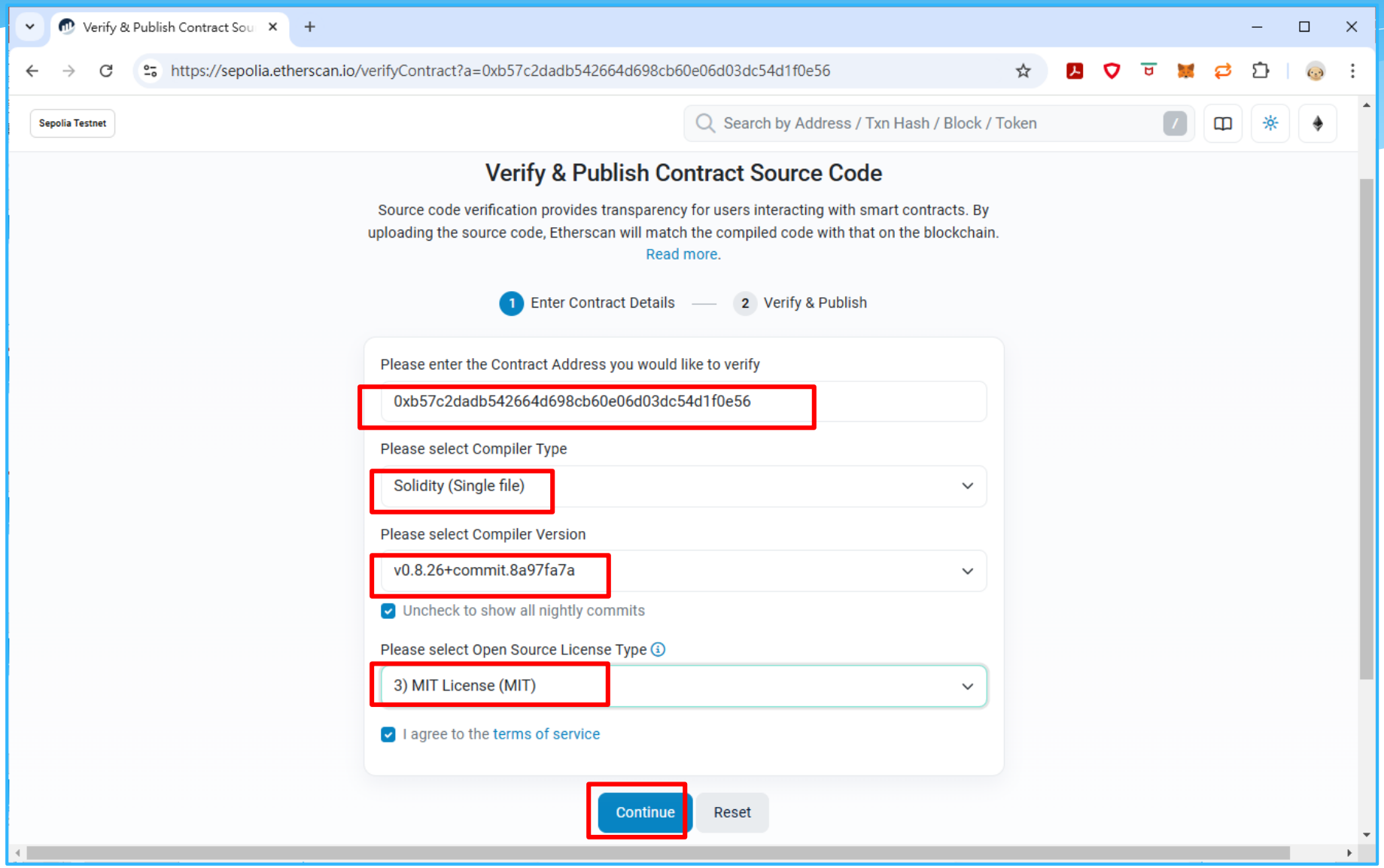
https://sepolia.etherscan.io/token/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56

View & Publish



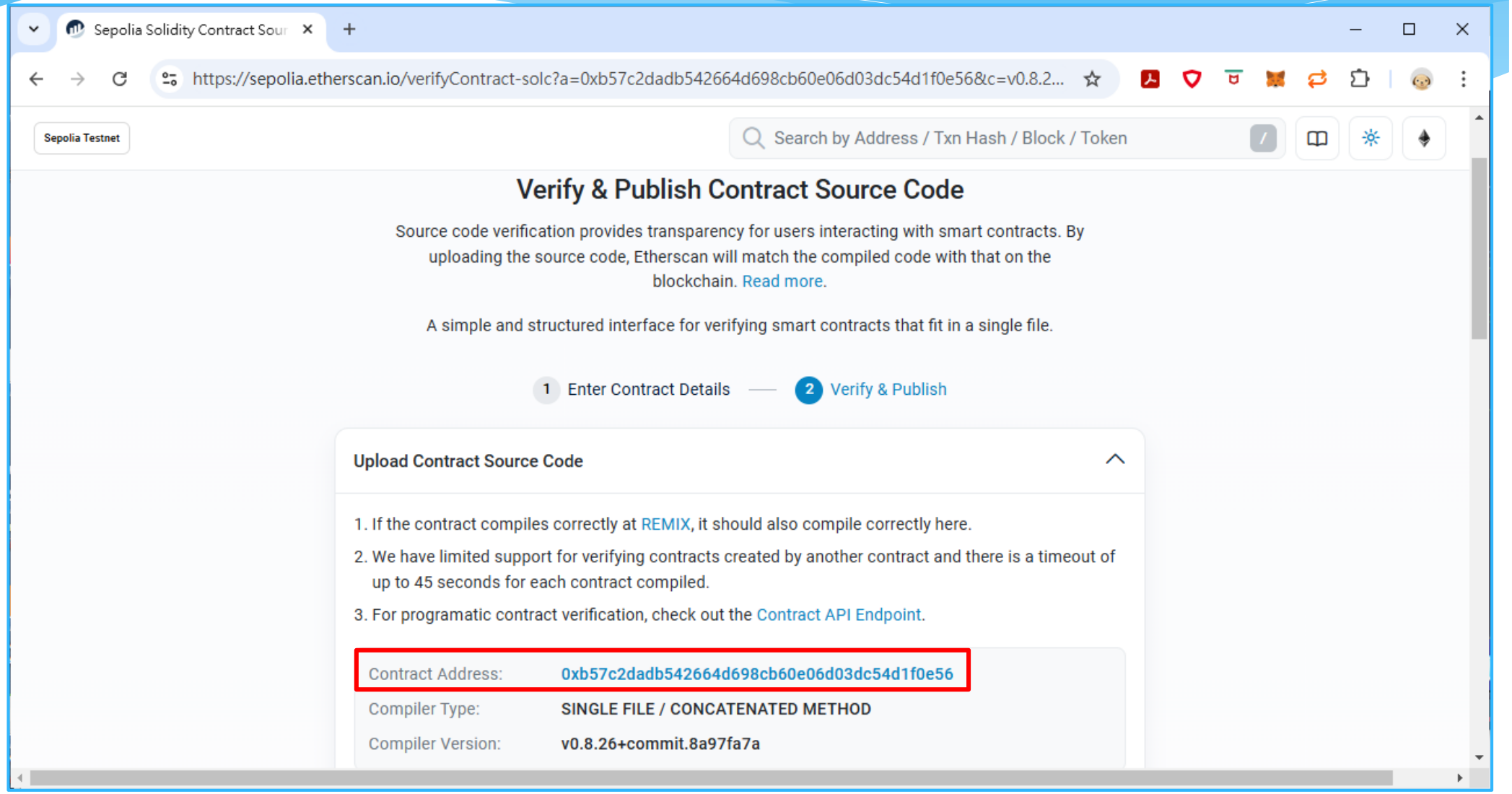
https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#code

View & Publish Contract Source Code

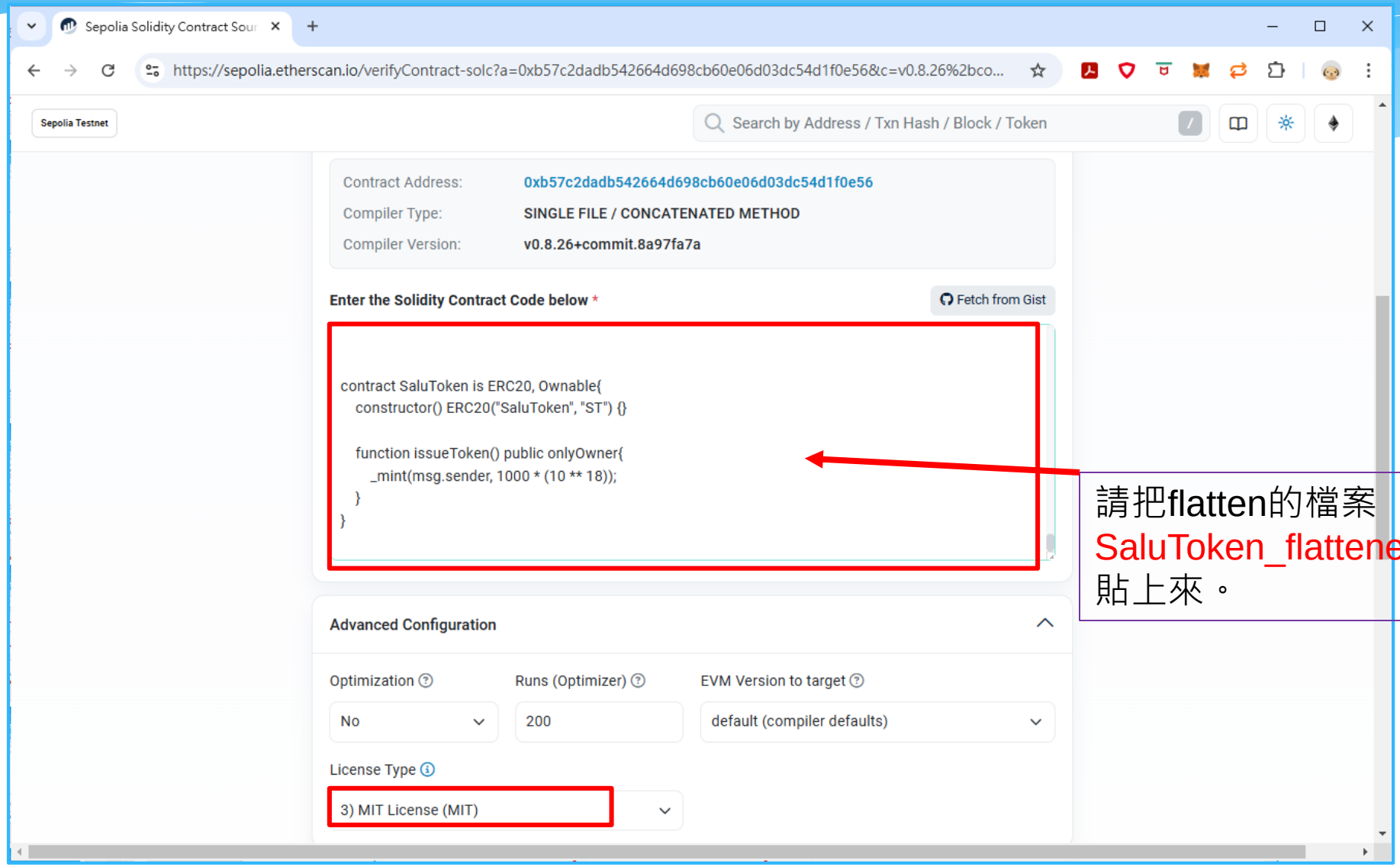


https://sepolia.etherscan.io/verifyContract?a=0xb57c2dadb542664d698cb60e06d03dc54d1f0e56

View & Publish Contract Source Code

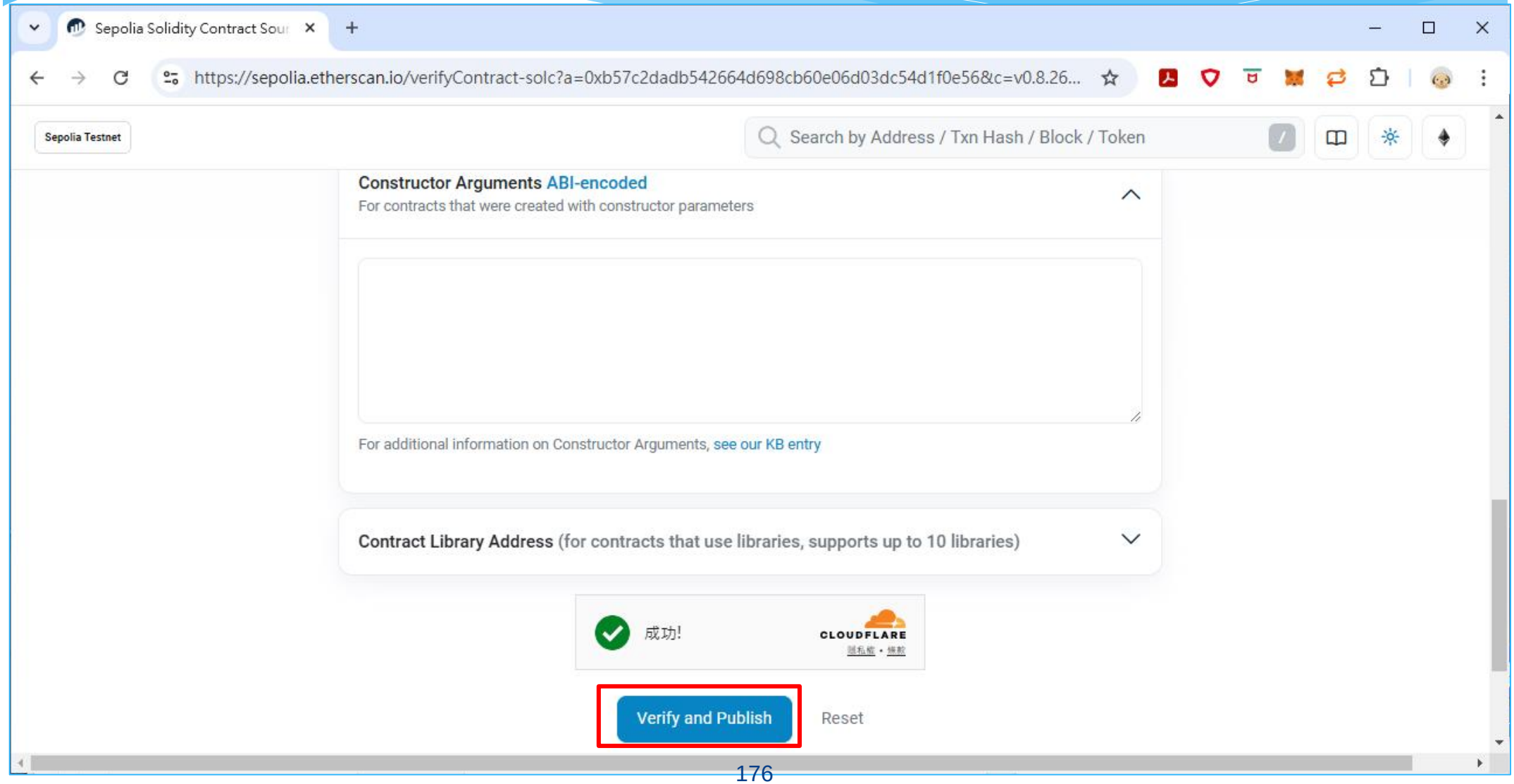


View & Publish Contract Source Code

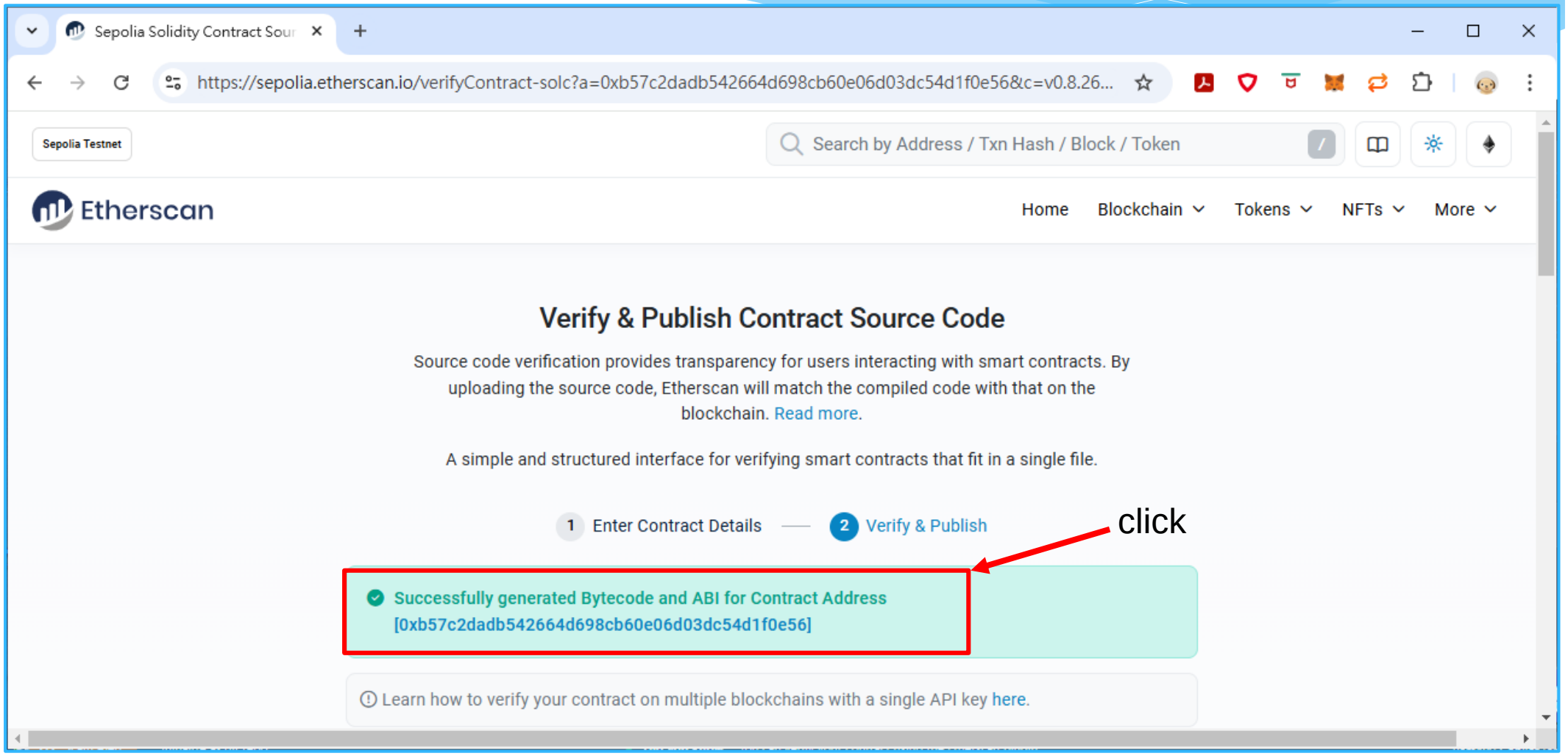


請把flatten的檔案SaluToken_flattened.sol貼上來。

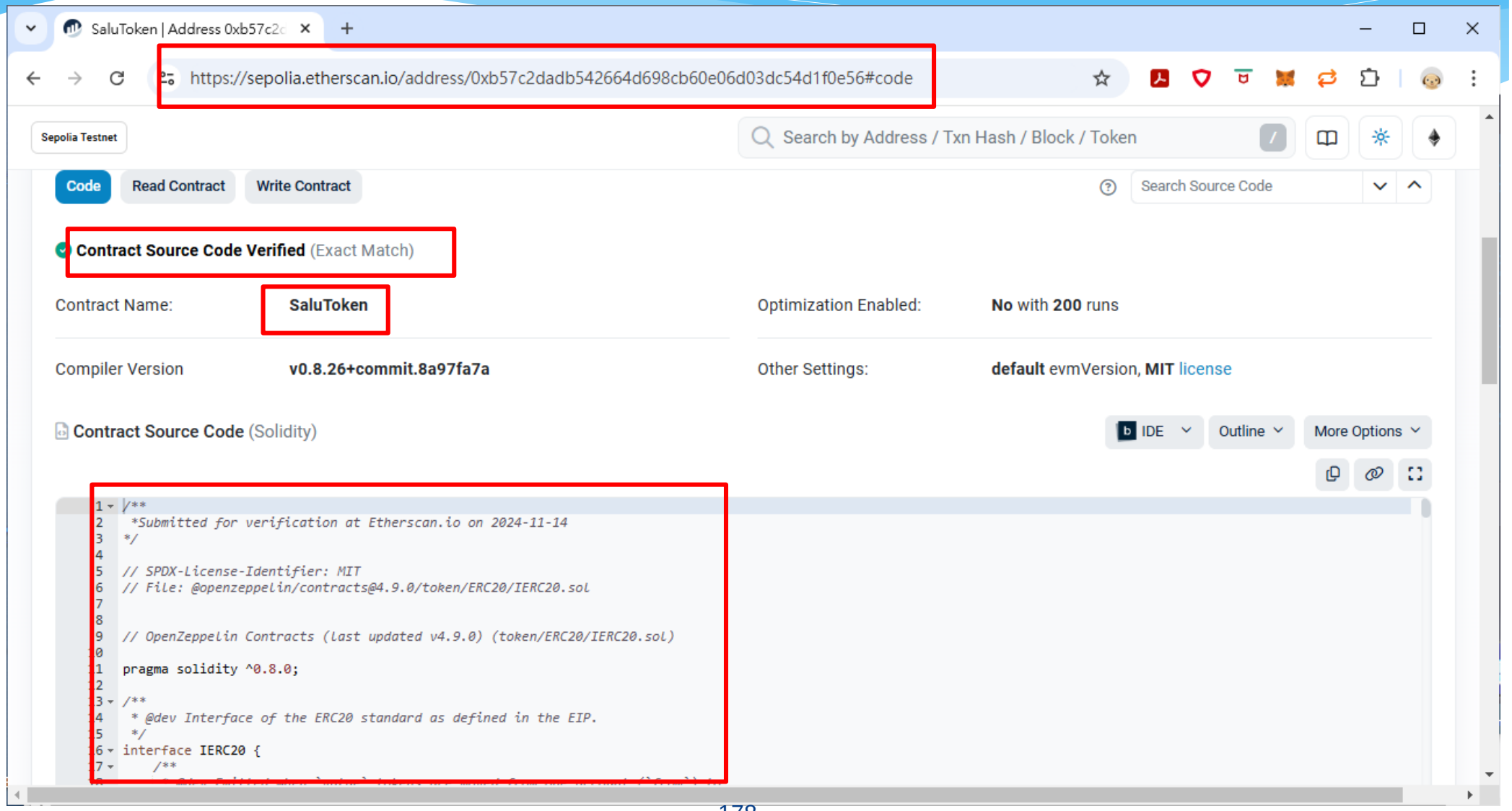
View & Publish Contract Source Code



View & Publish Contract Source Code

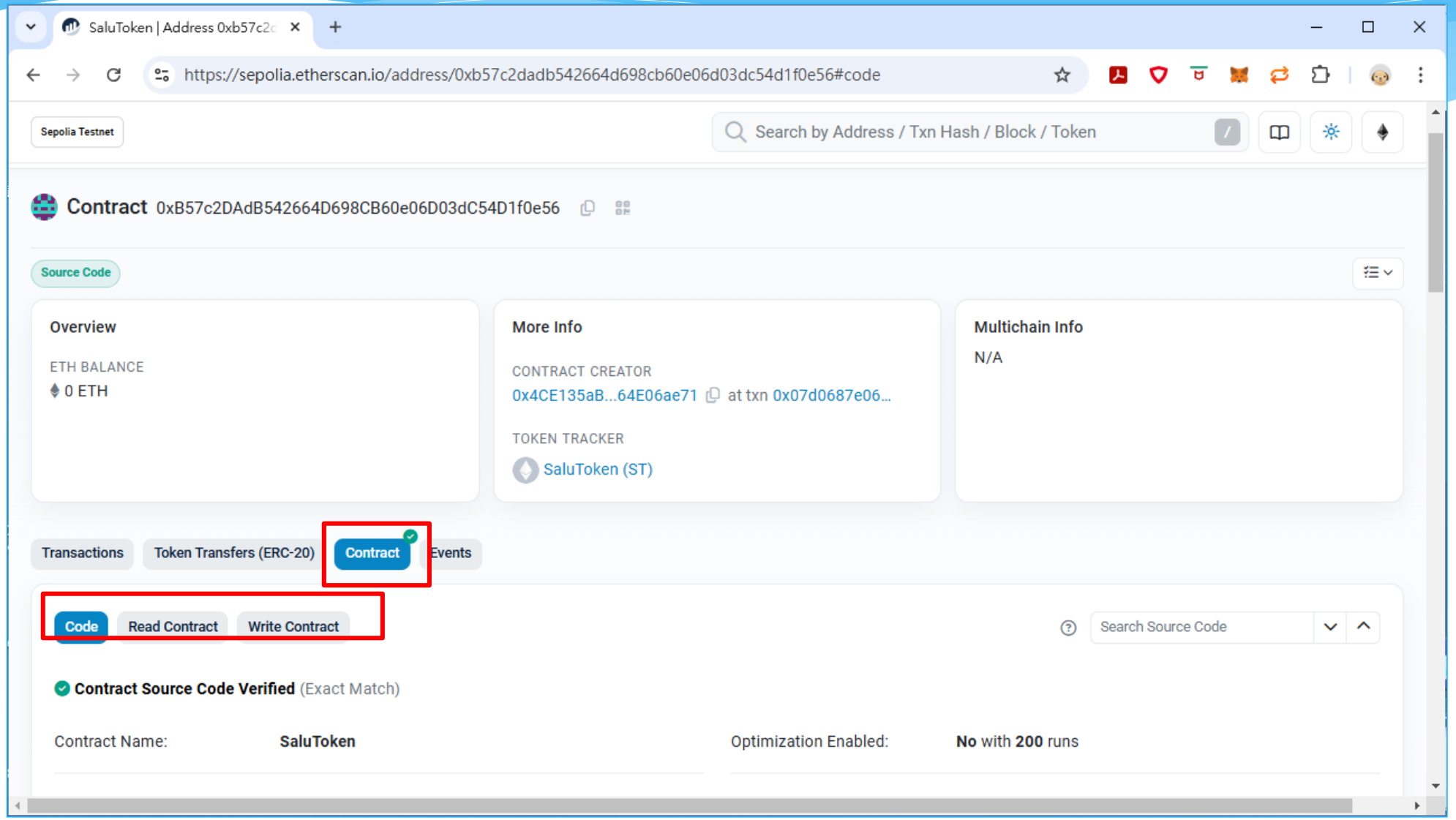


View Contract Source Code



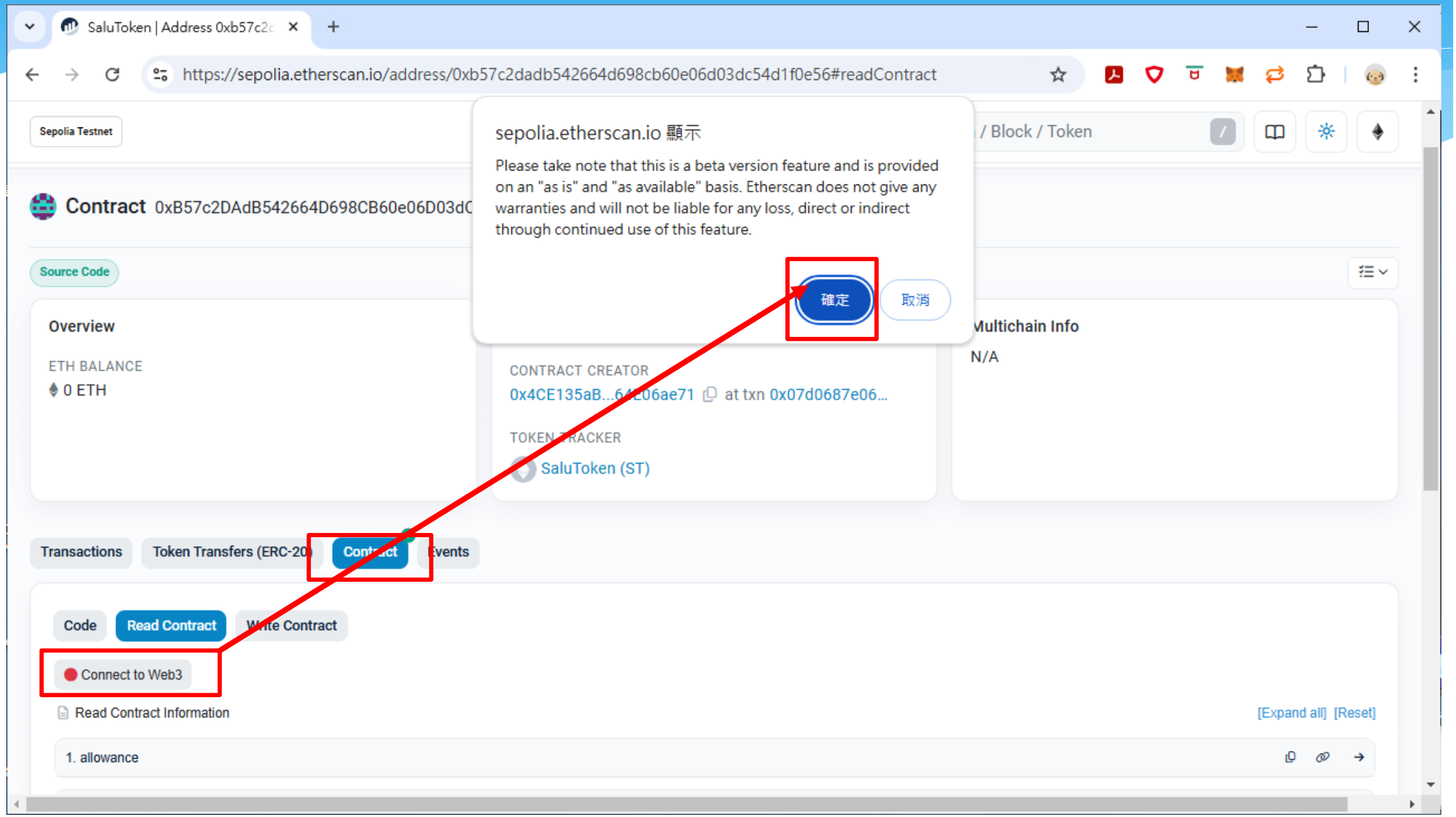
https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#code

View & Publish Contract Source Code



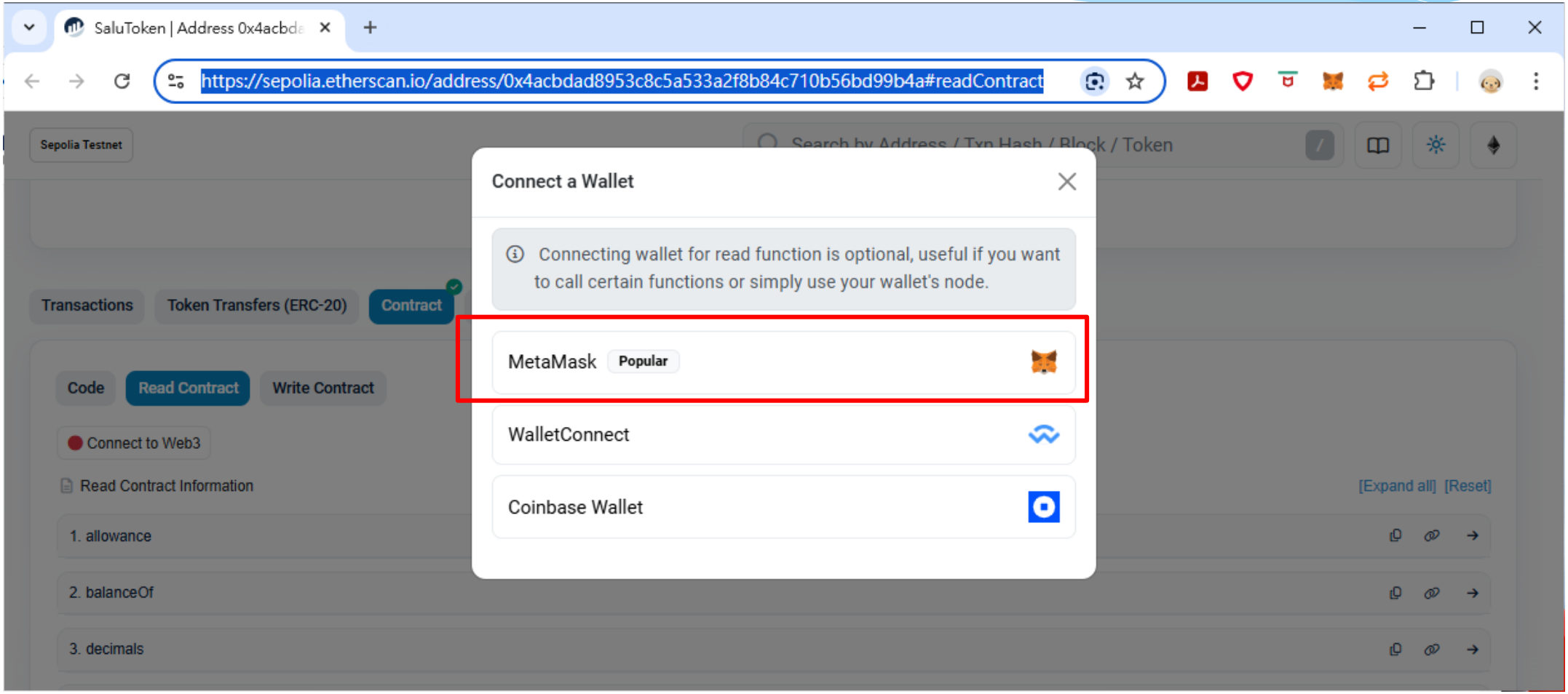
https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#code

Connect to Web3



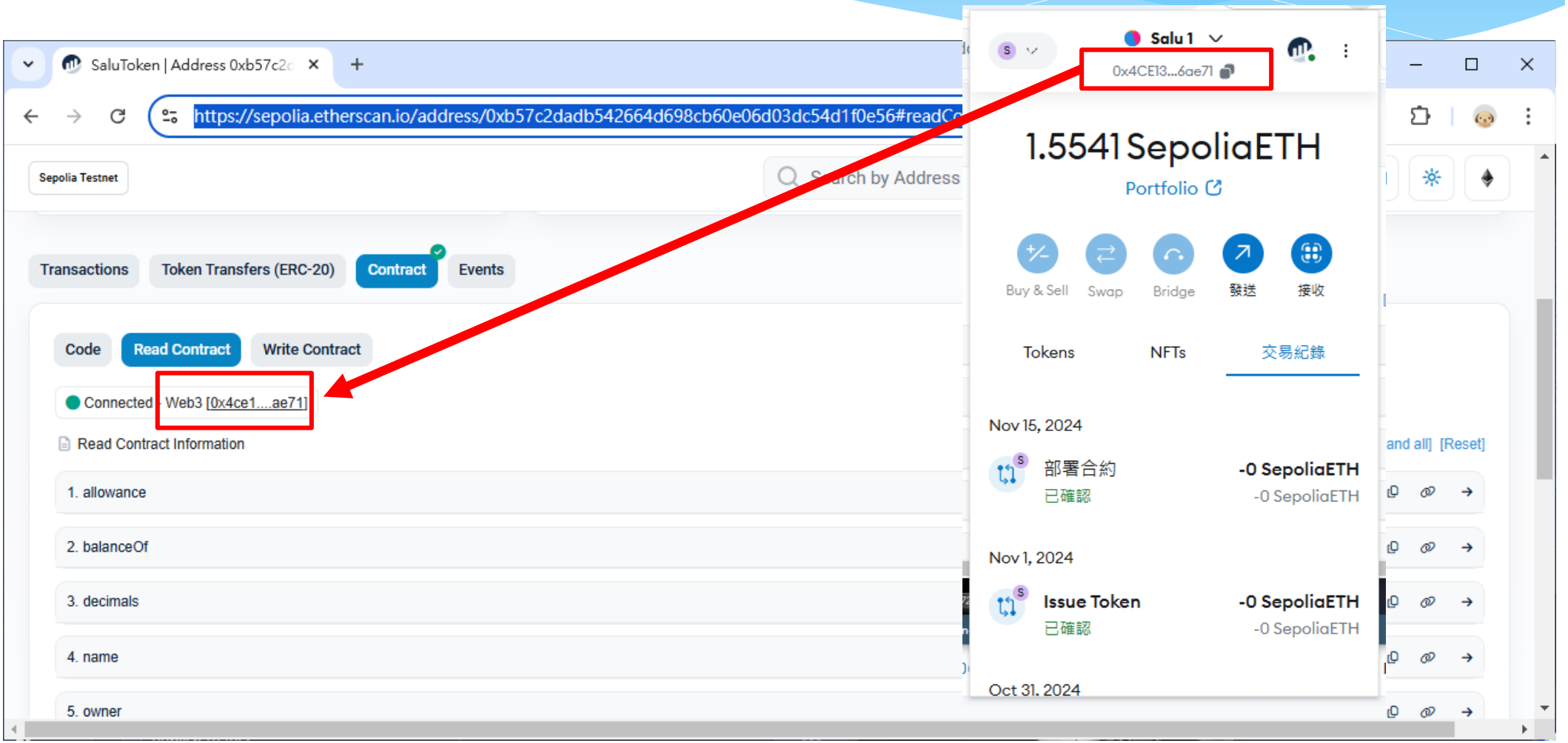
https://sepolia.etherscan.io/address/0x4acbdad8953c8c5a533a2f8b84c710b56bd99b4a#readContract

Connect a Wallet-MetaMask



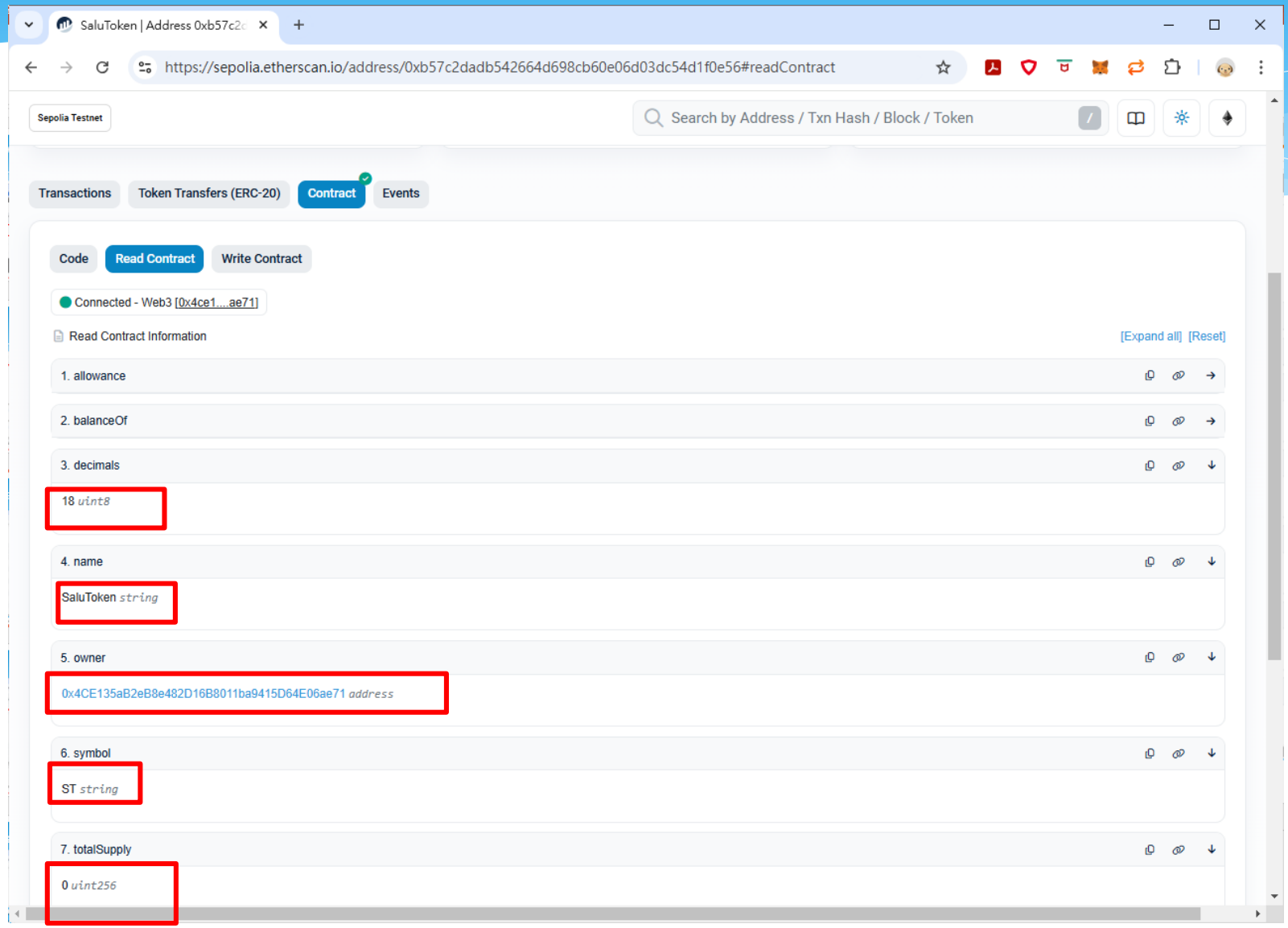
https://sepolia.etherscan.io/address/0x4acbdad8953c8c5a533a2f8b84c710b56bd99b4a#readContract

Connected Web3 using account



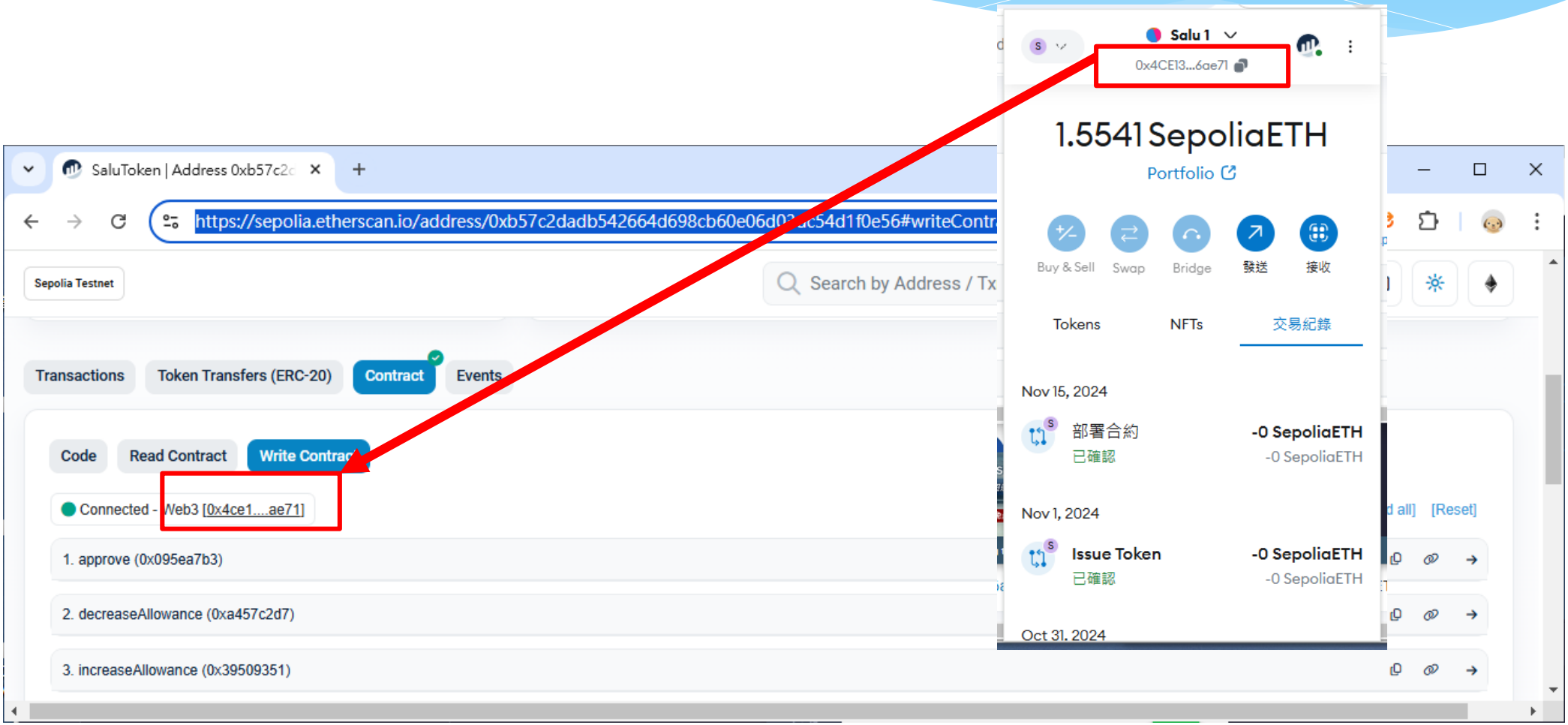
https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#readContract

Read Contract



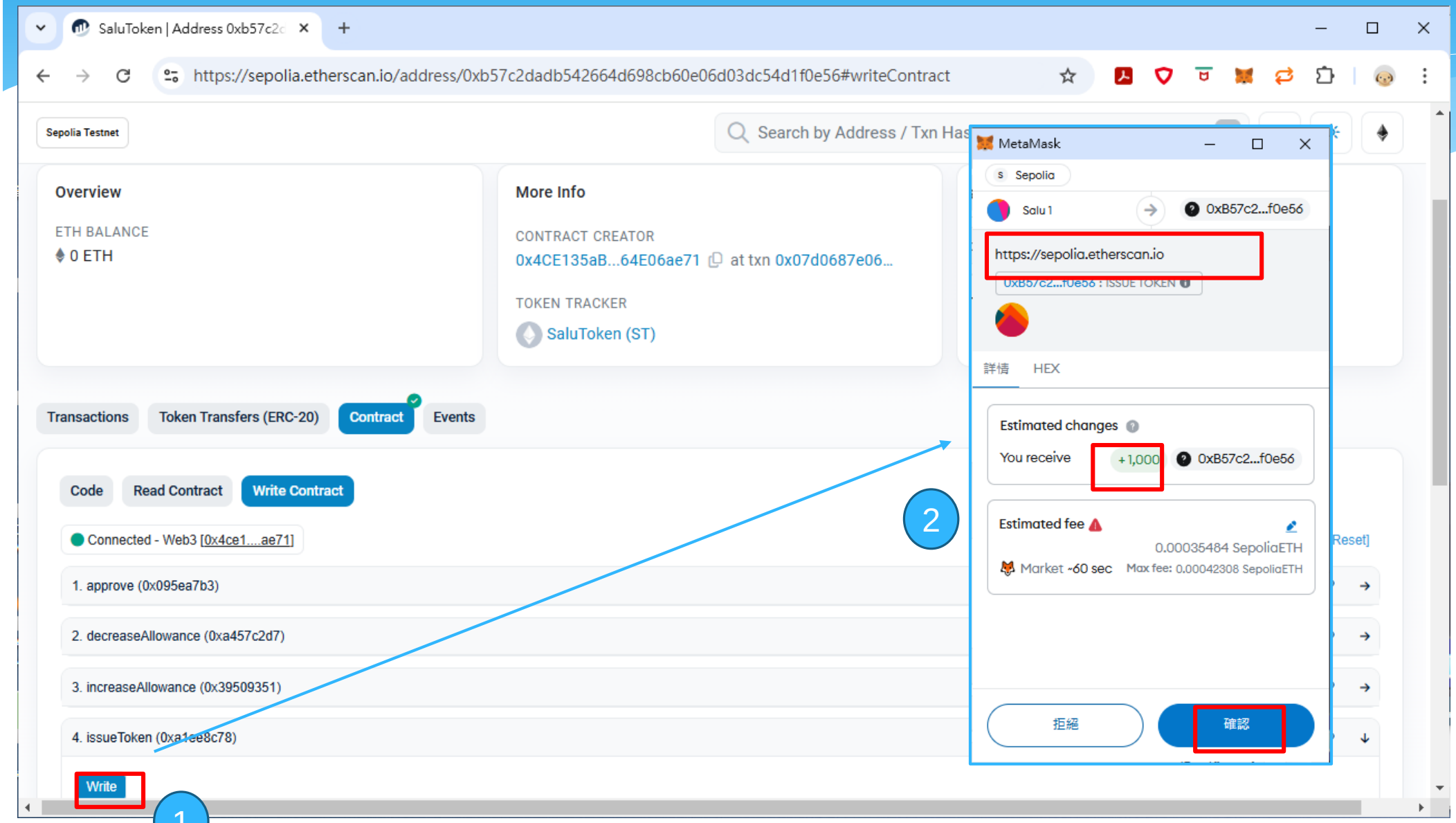
https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#readContract

Connected Web3 using account



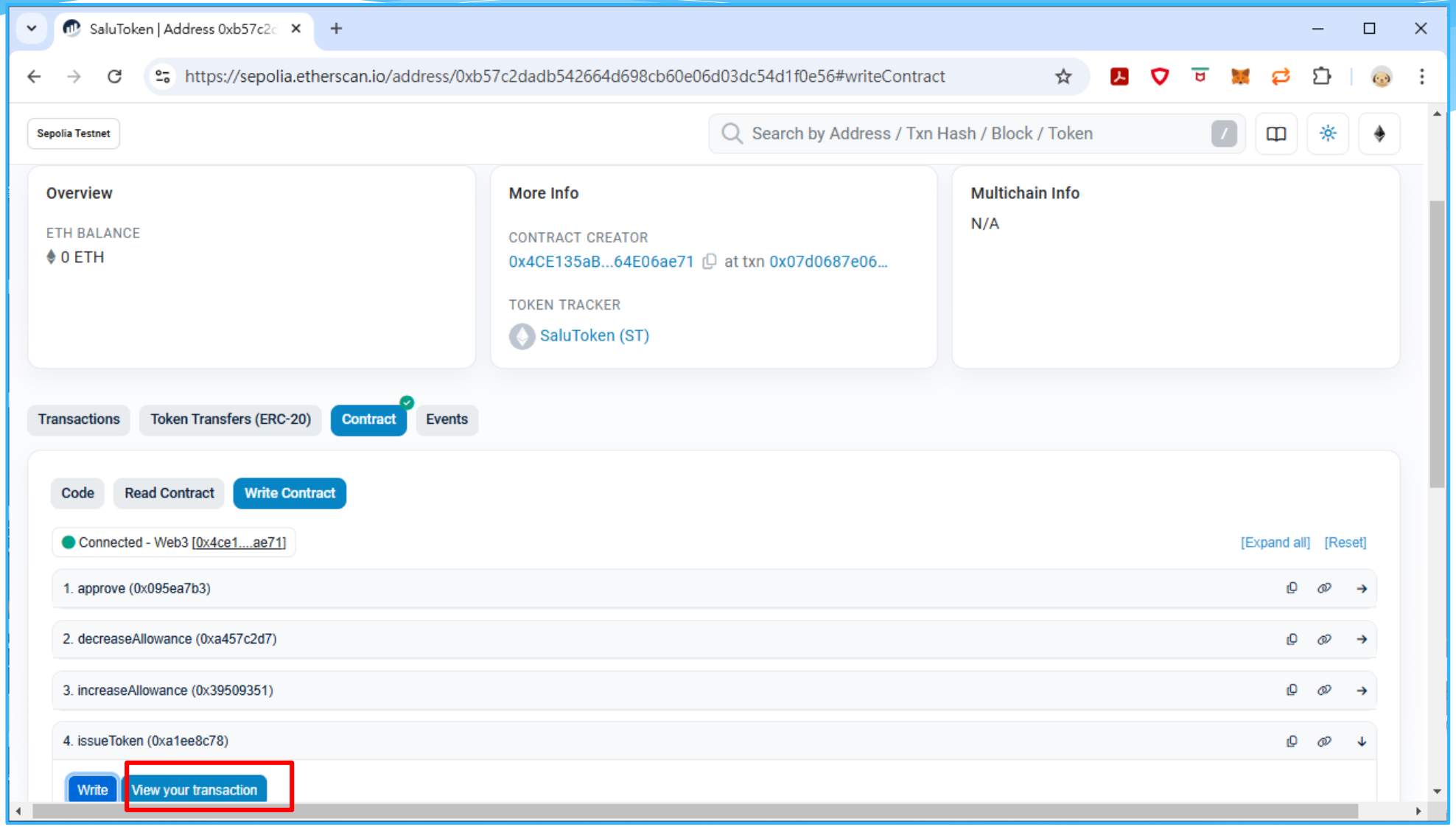
https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#writeContract

Write Contract - issueToken



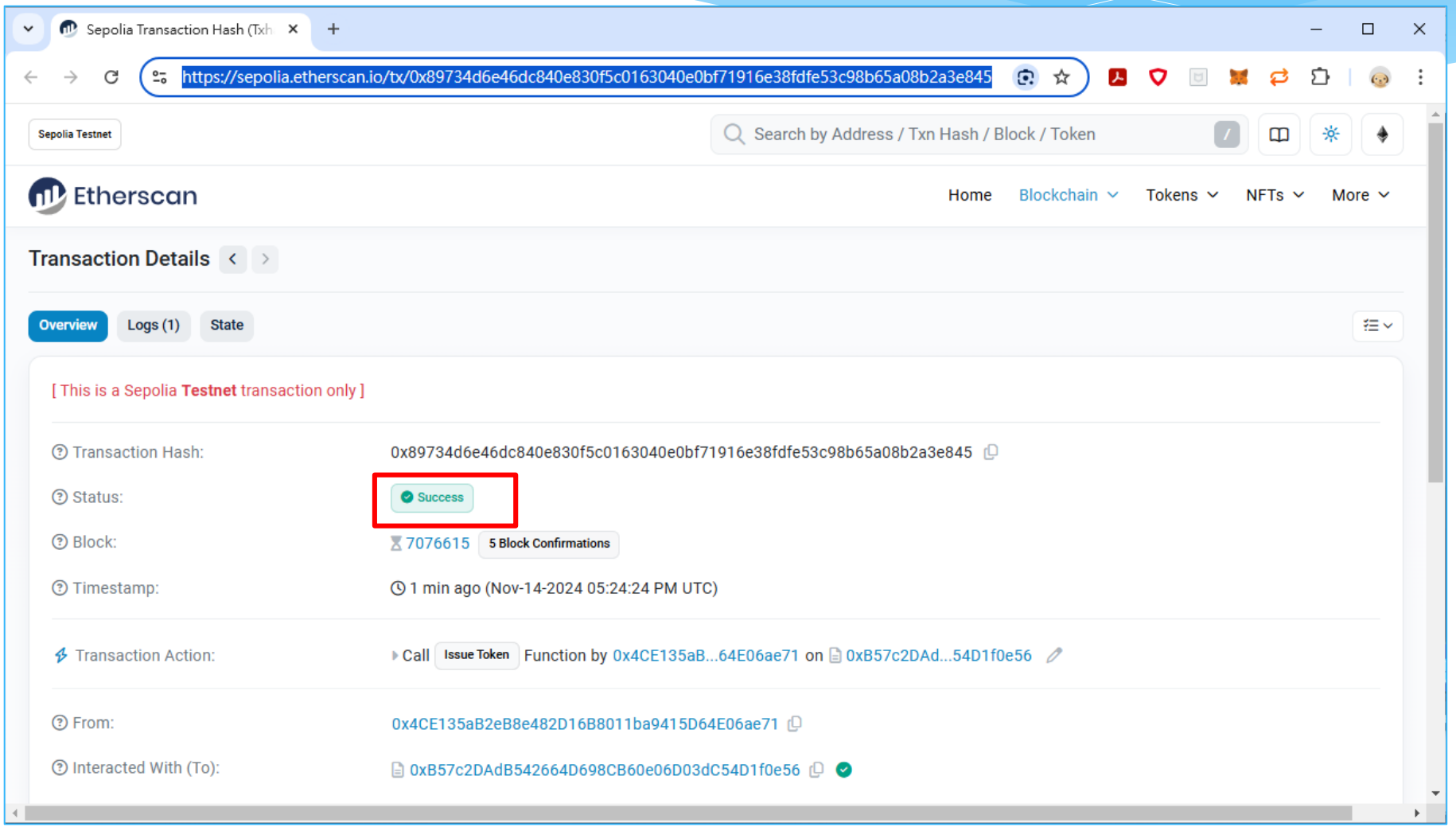
https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#writeContract

Write Contract - issueToken



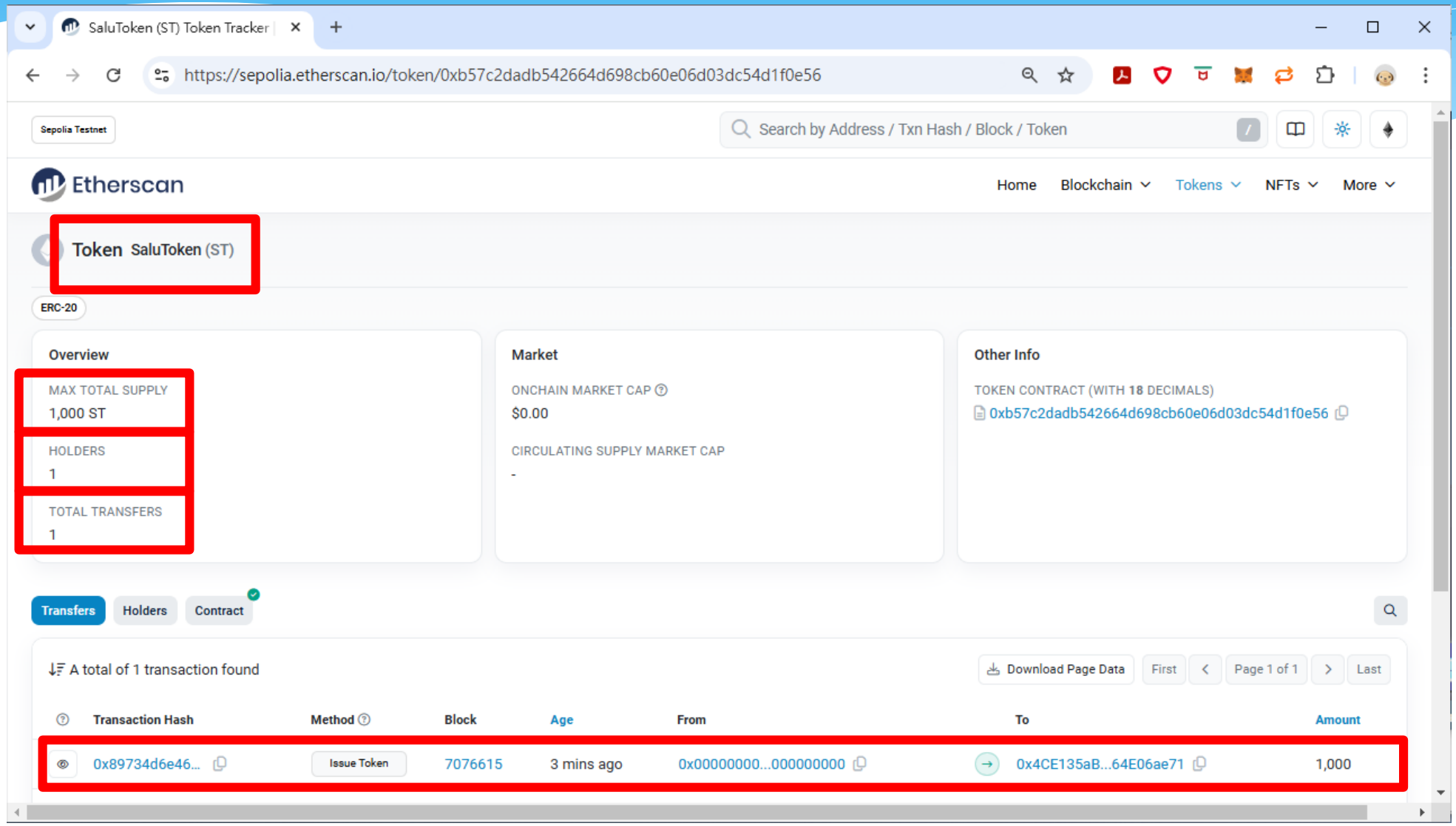
https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#writeContract

View Transaction



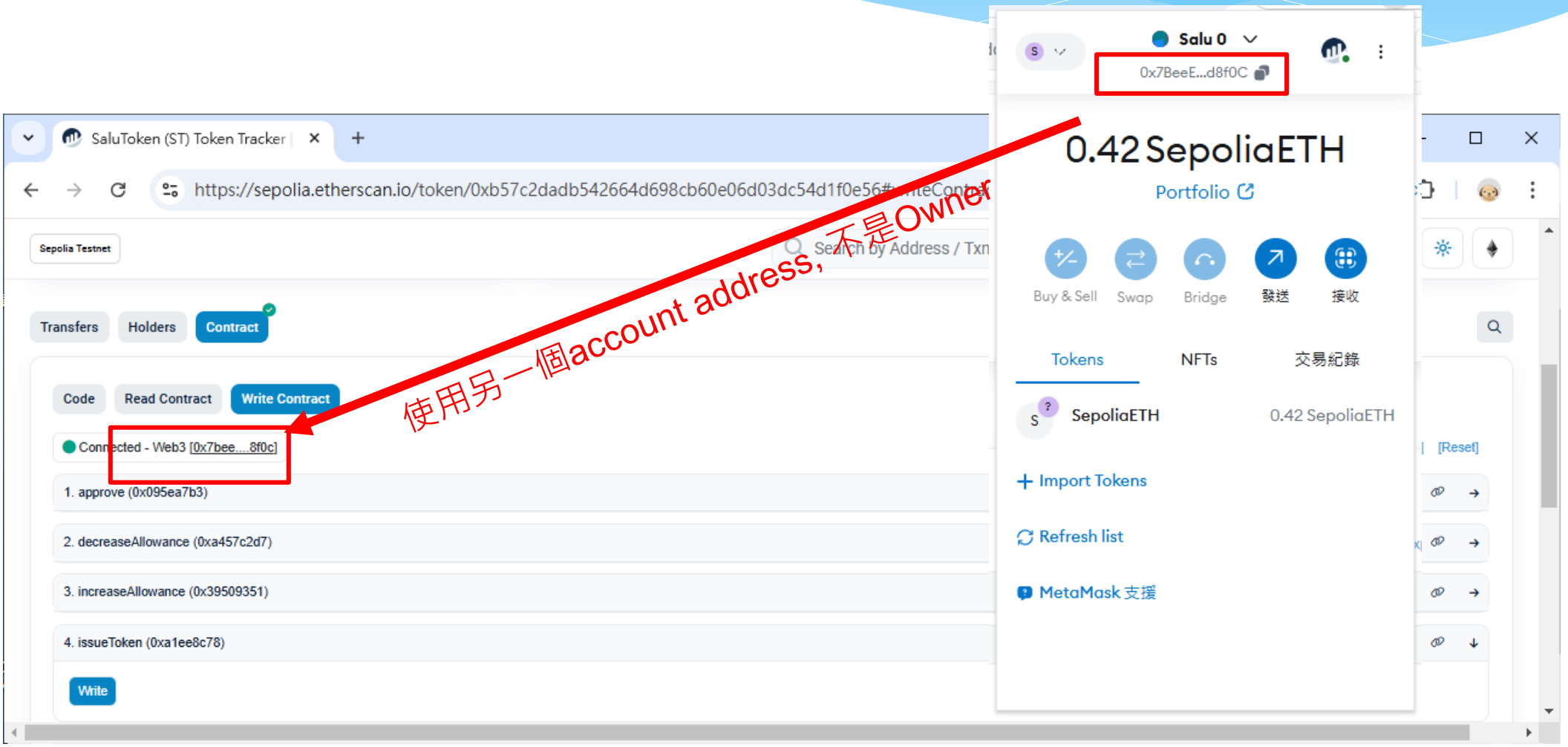
https://sepolia.etherscan.io/tx/0x89734d6e46dc840e830f5c0163040e0bf71916e38fdfe53c98b65a08b2a3e845

Smart Contract Information



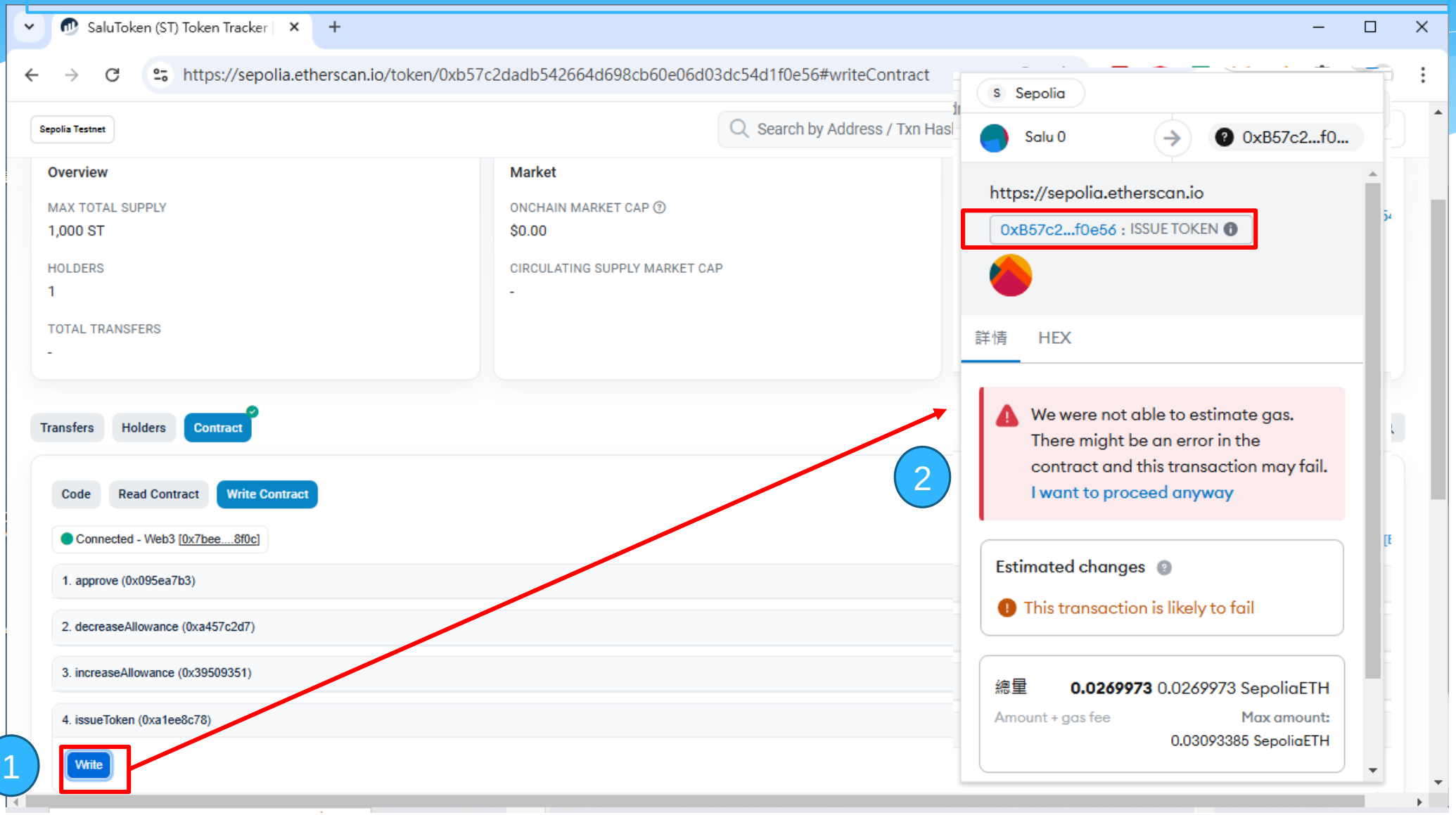
https://sepolia.etherscan.io/token/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56

Connected Web3 using account



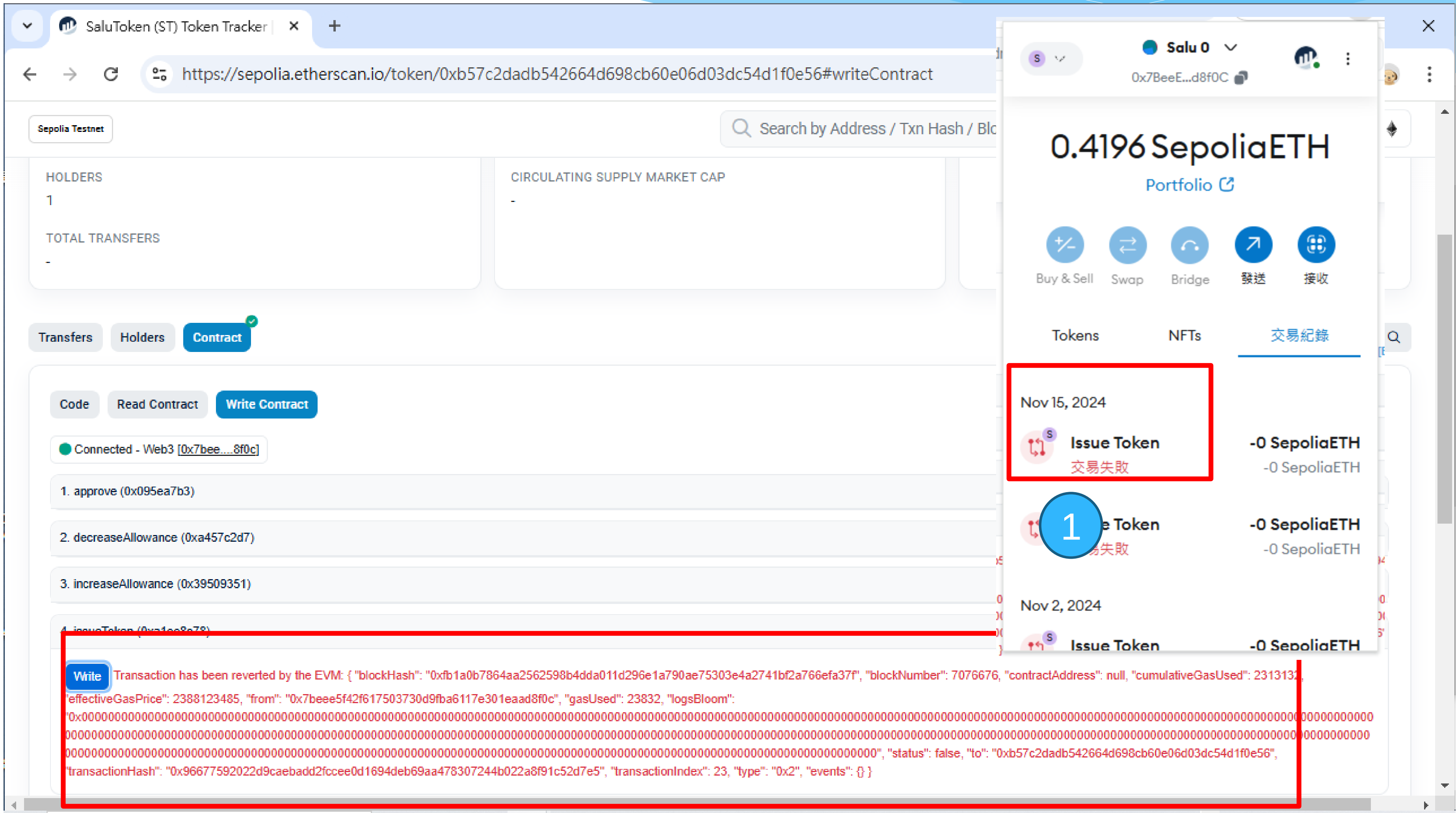
https://sepolia.etherscan.io/token/0x4acbdad8953c8c5a533a2f8b84c710b56bd99b4a#writeContract

Write Contract - issueToken



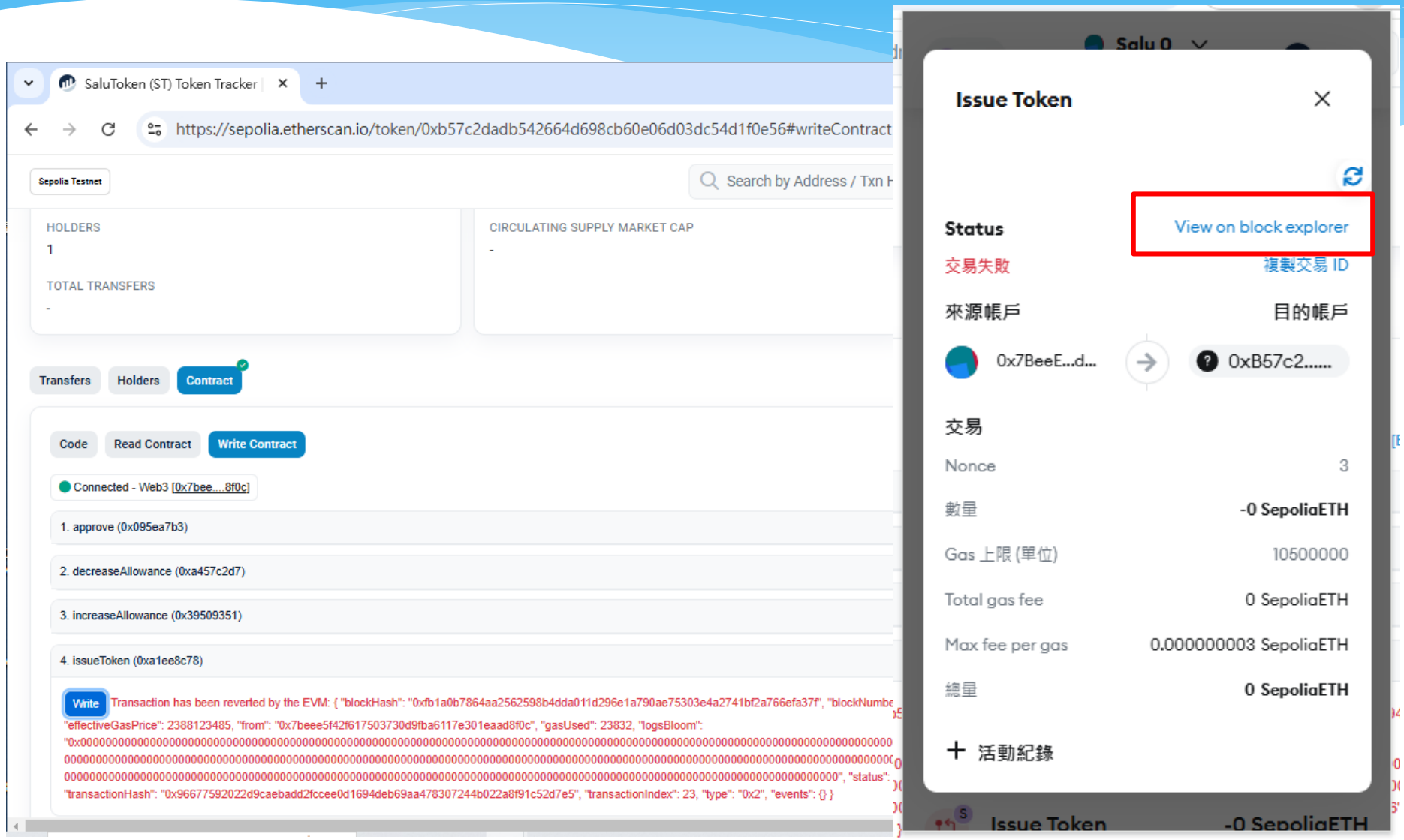
https://sepolia.etherscan.io/address/0x4acbdad8953c8c5a533a2f8b84c710b56bd99b4a#writeContract

Write Contract - issueToken



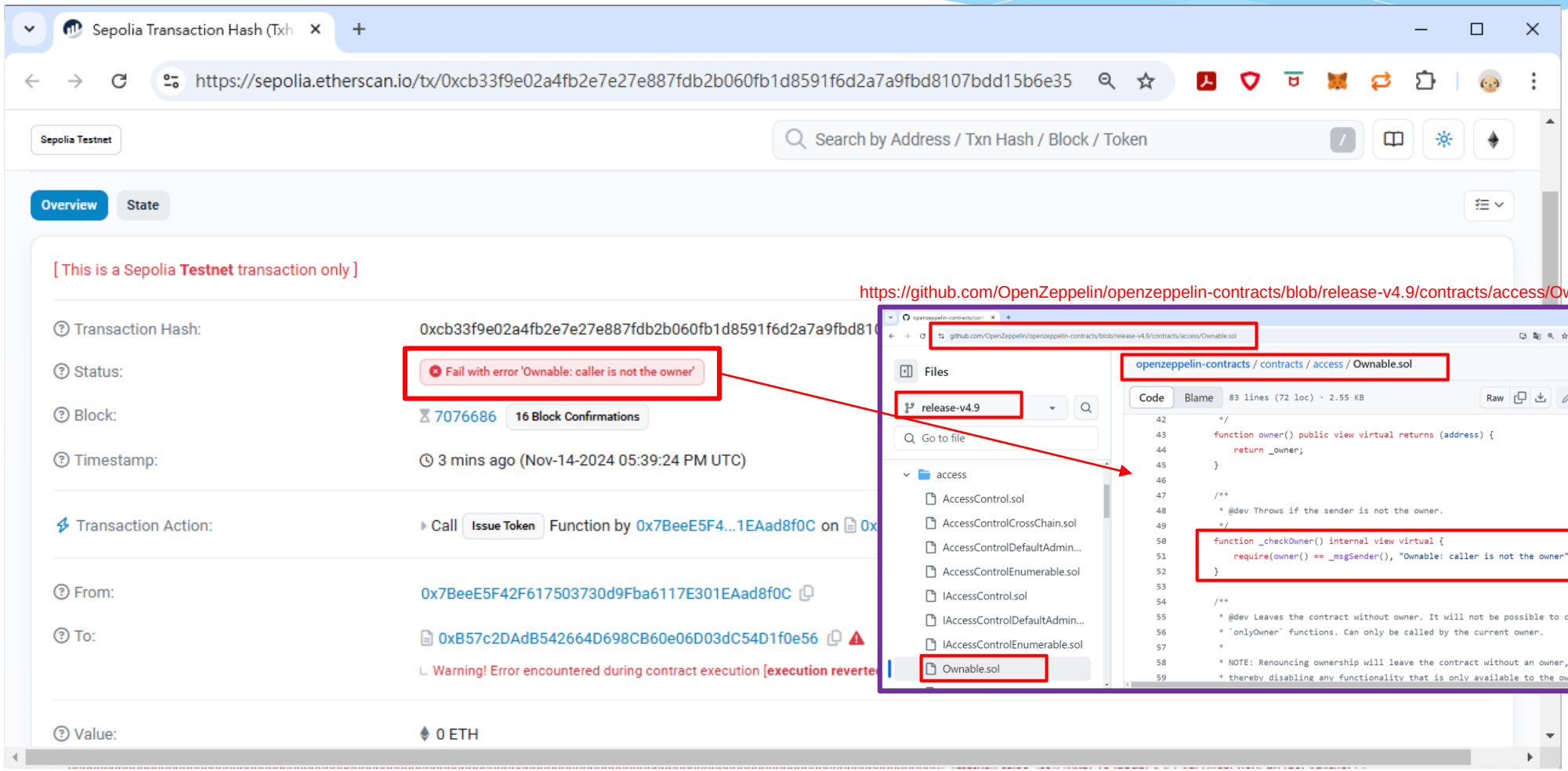
https://sepolia.etherscan.io/token/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#writeContract

View Transaction



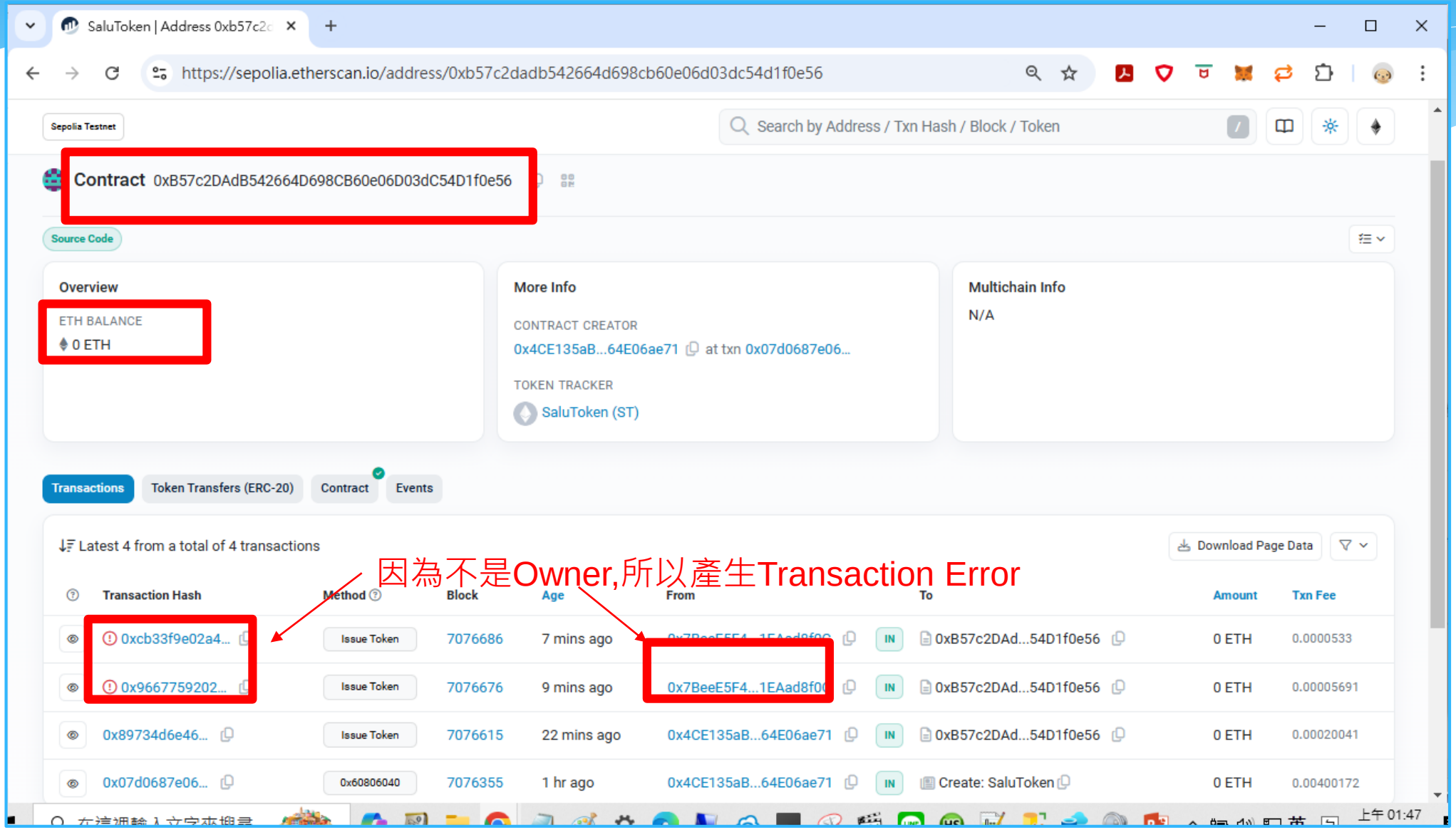
https://sepolia.etherscan.io/token/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56#writeContract

View Transaction



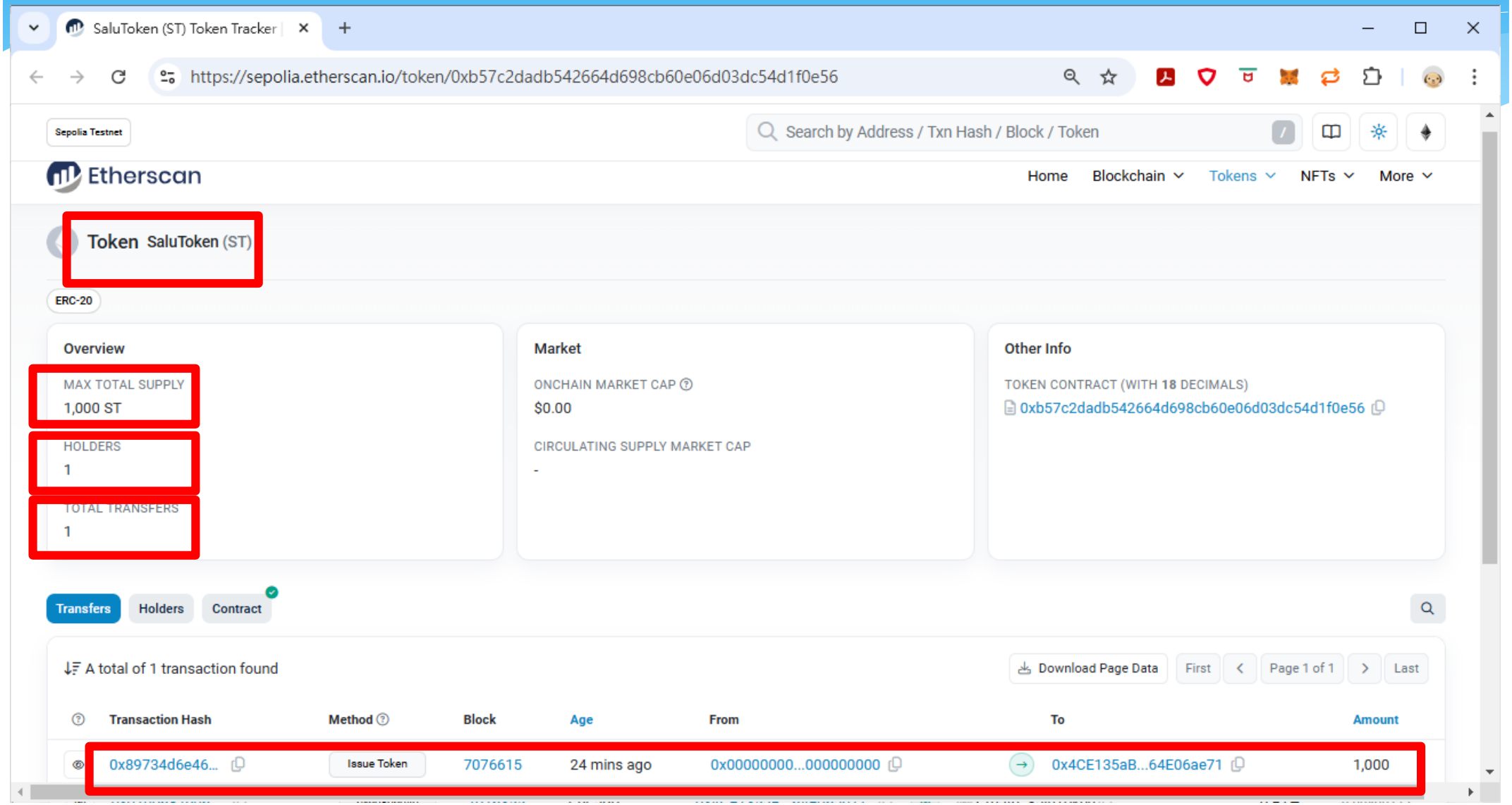
https://sepolia.etherscan.io/tx/0xcb33f9e02a4fb2e7e27e887fdb2b060fb1d8591f6d2a7a9fbd8107bdd15b6e35

Smart Contract Information



https://sepolia.etherscan.io/address/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56

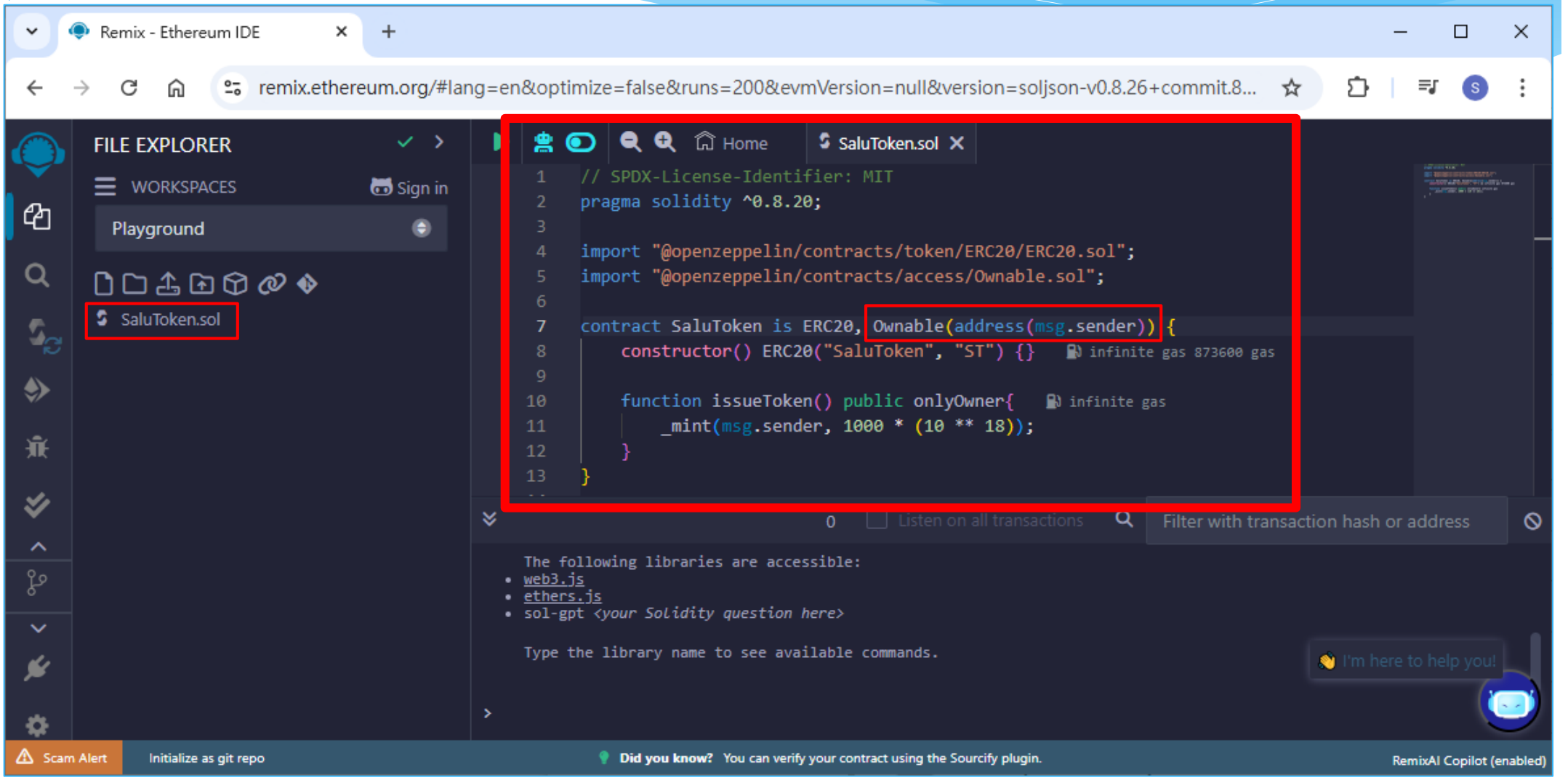
Smart Contract Information



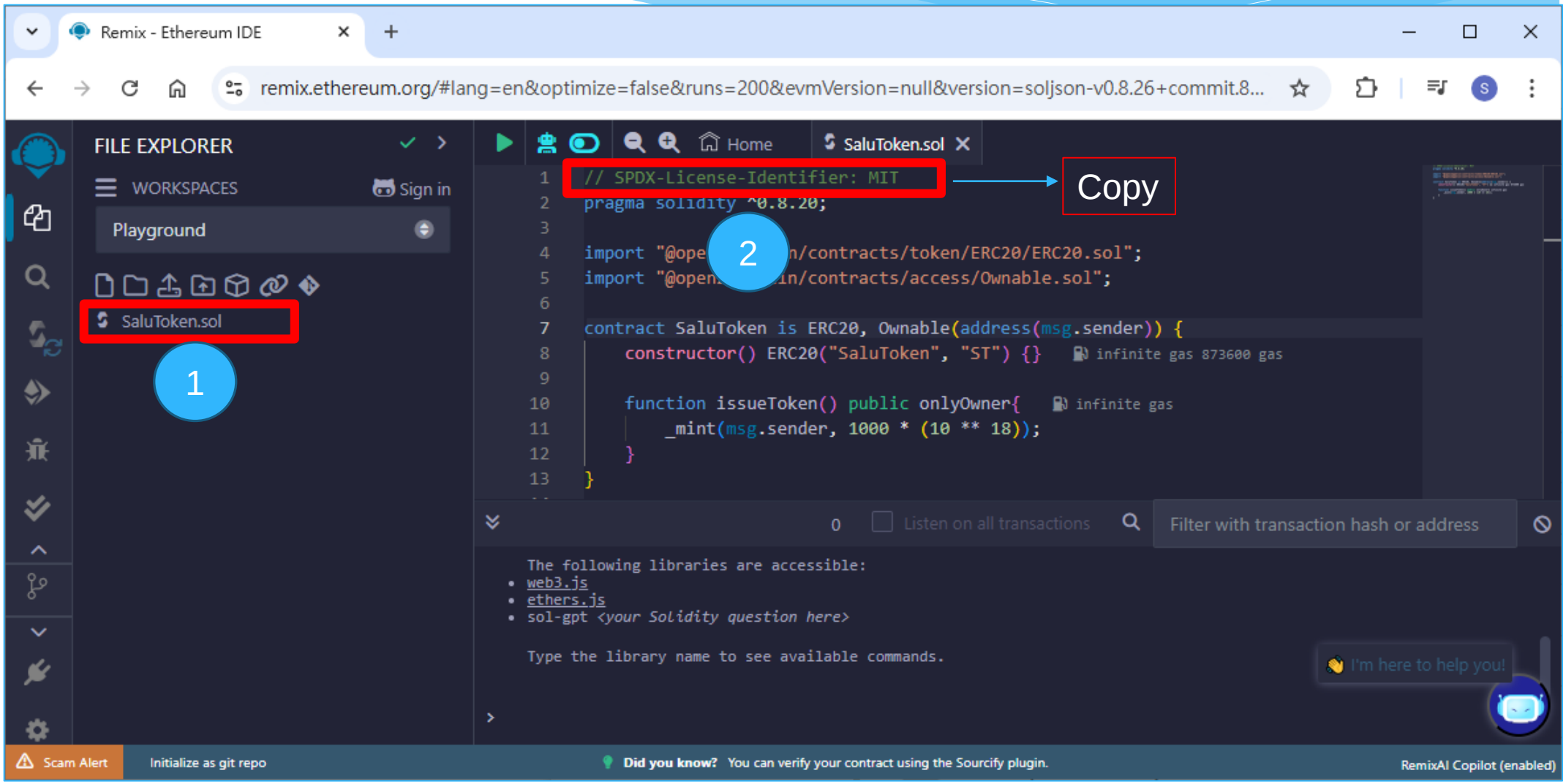
https://sepolia.etherscan.io/token/0xb57c2dadb542664d698cb60e06d03dc54d1f0e56

Openzeppelin v5.1.0

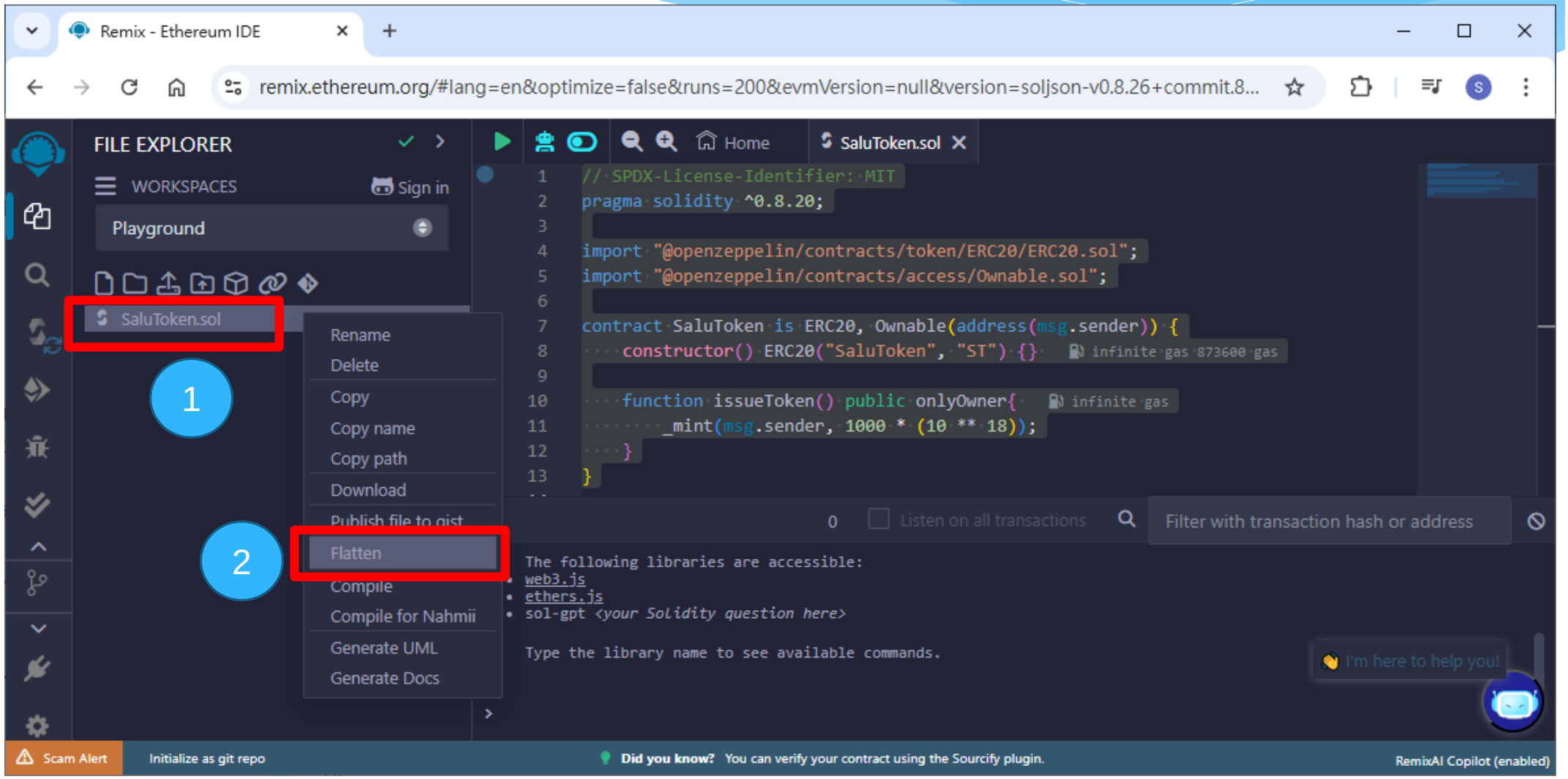
Modify SaluToken.sol



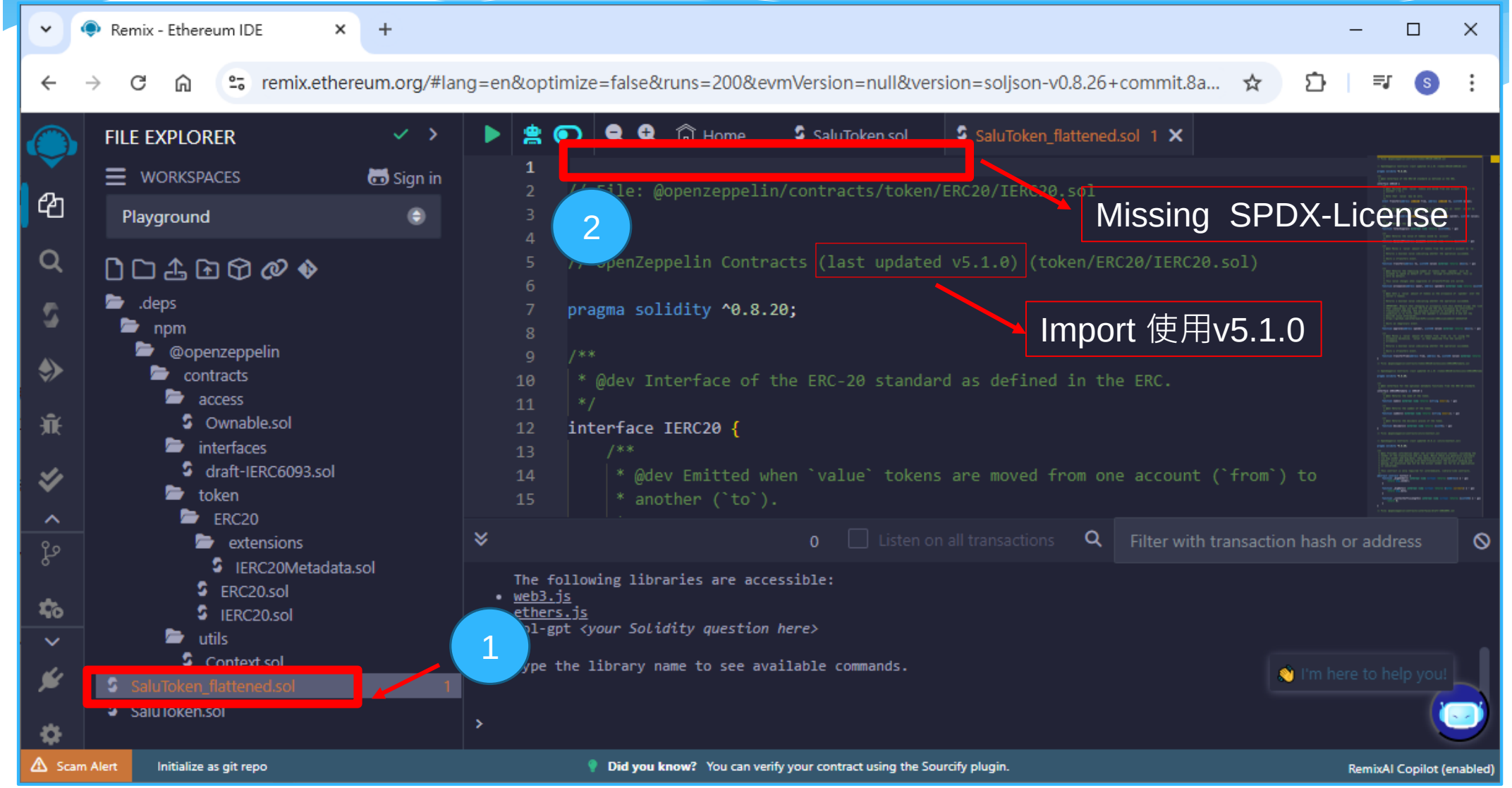
Copy SPDX-License



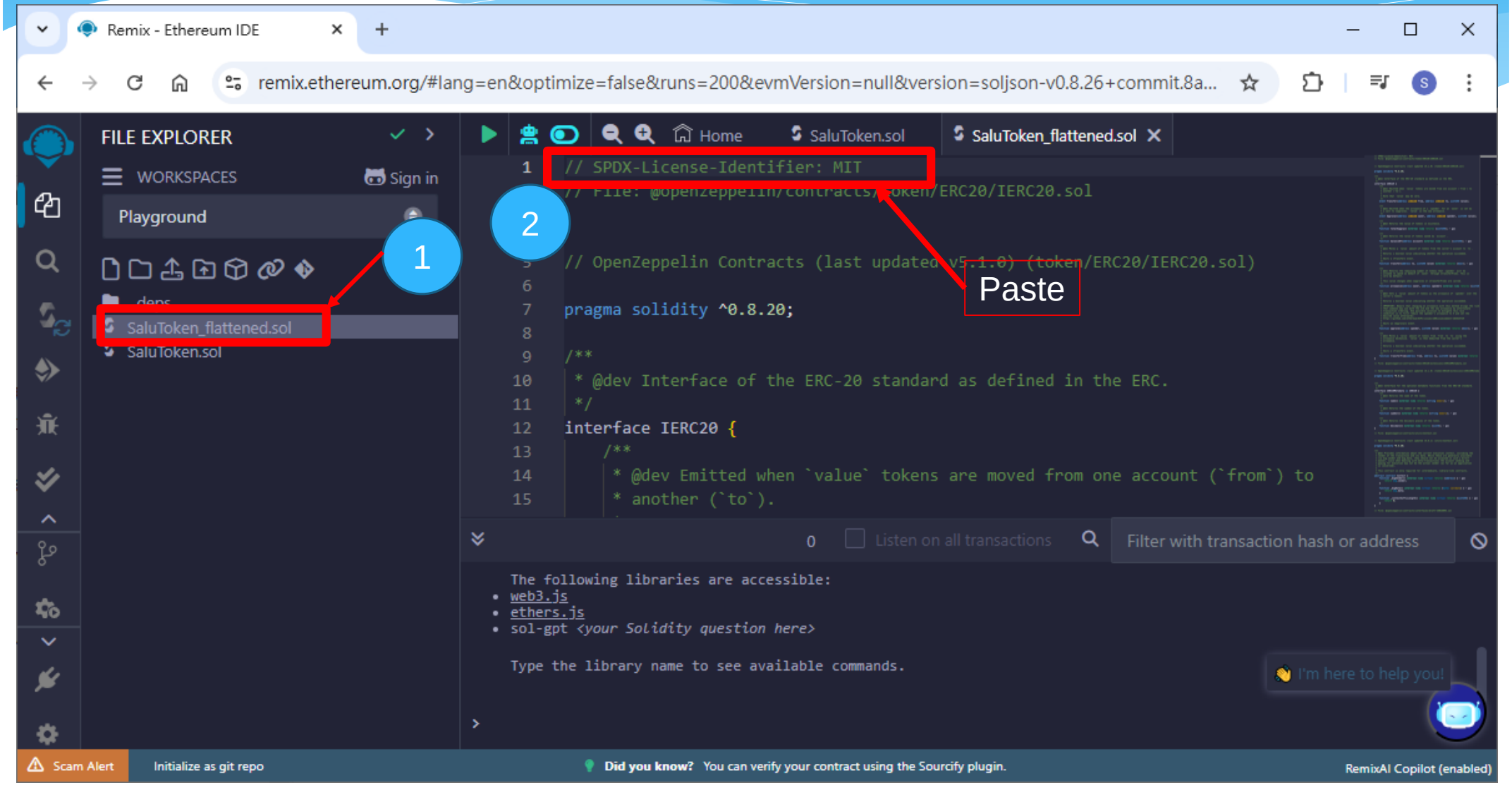
Flatten SaluToken.sol



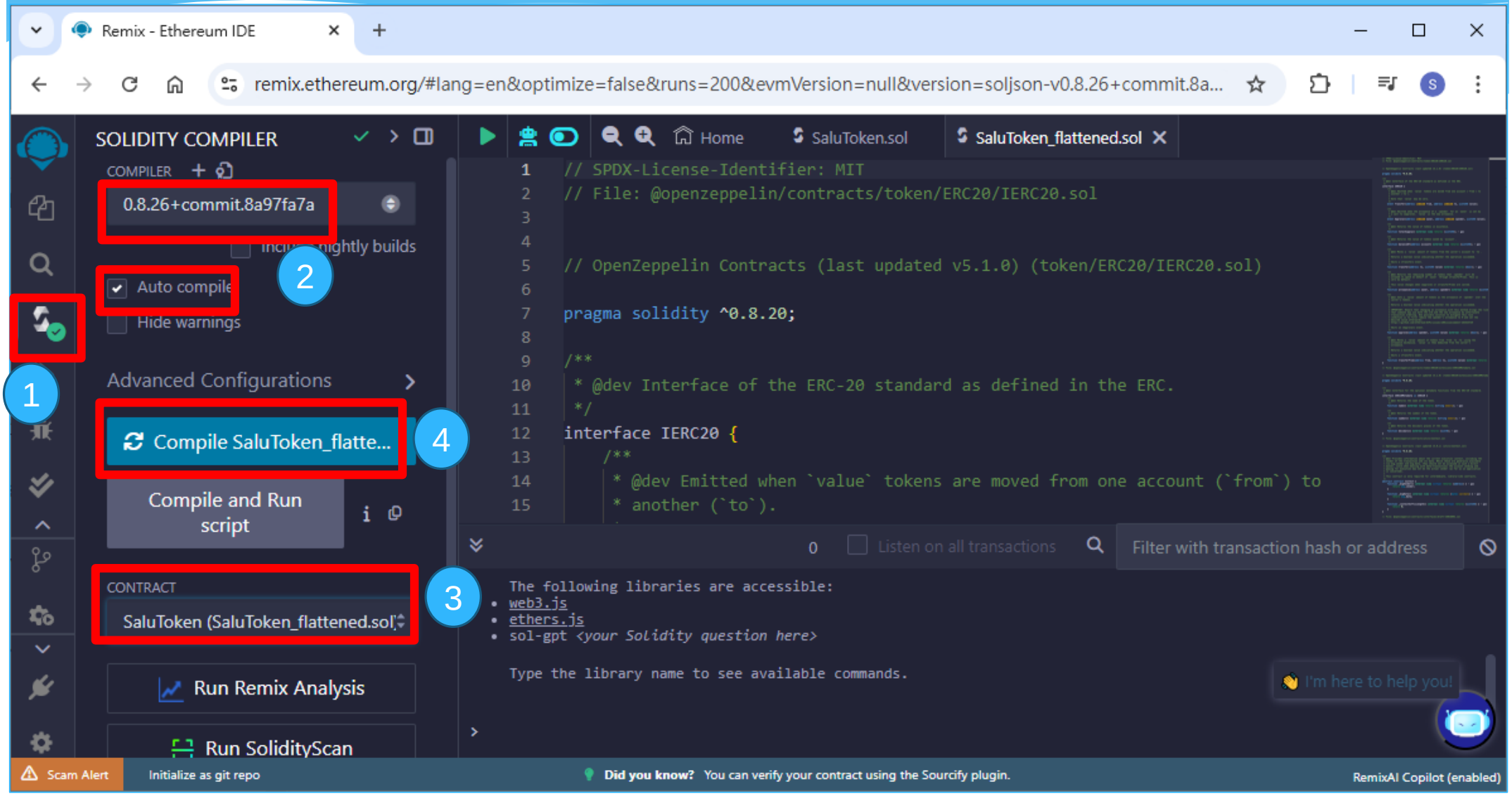
SaluToken_flattened.sol



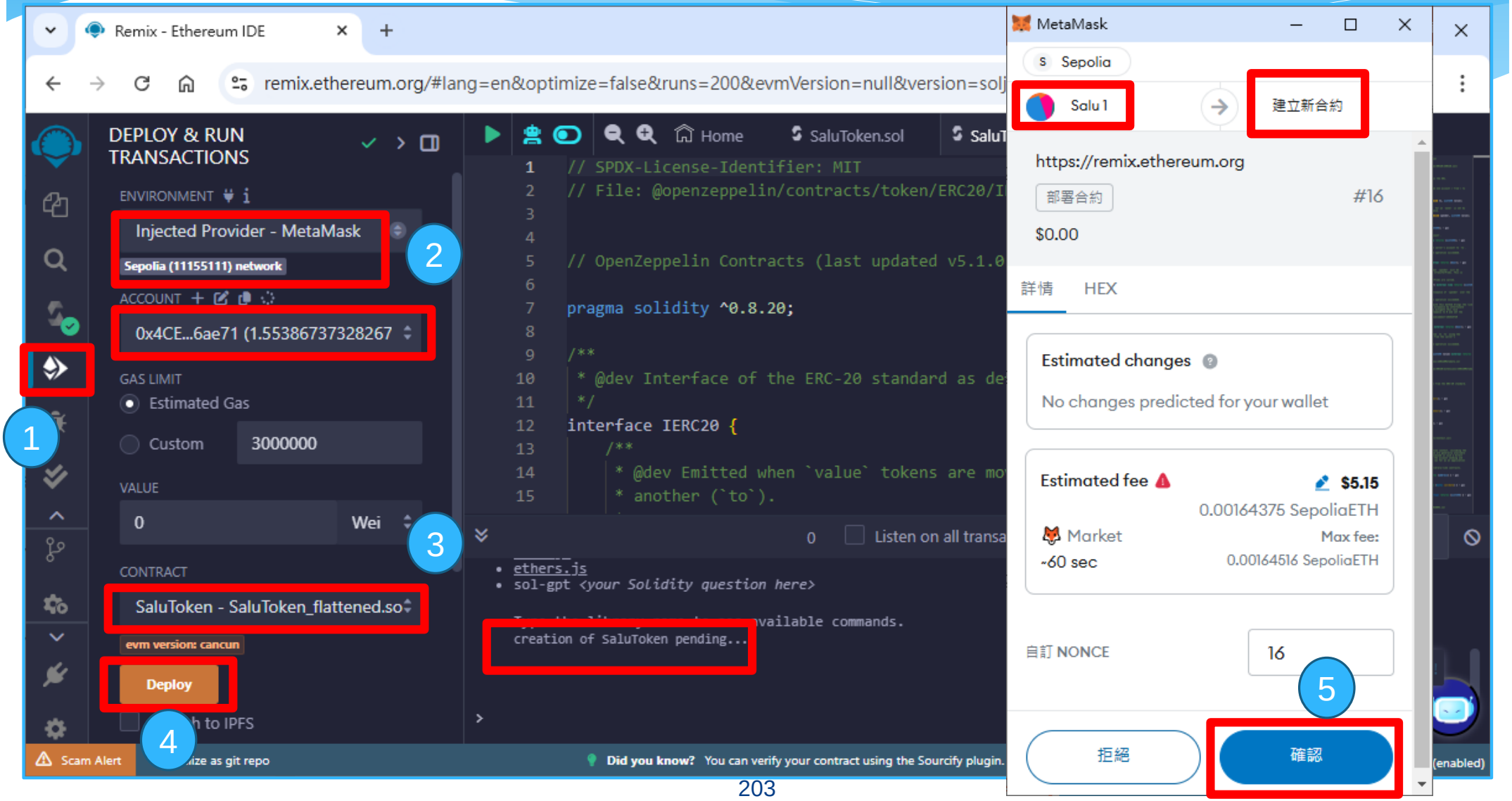
SaluToken_flattened.sol



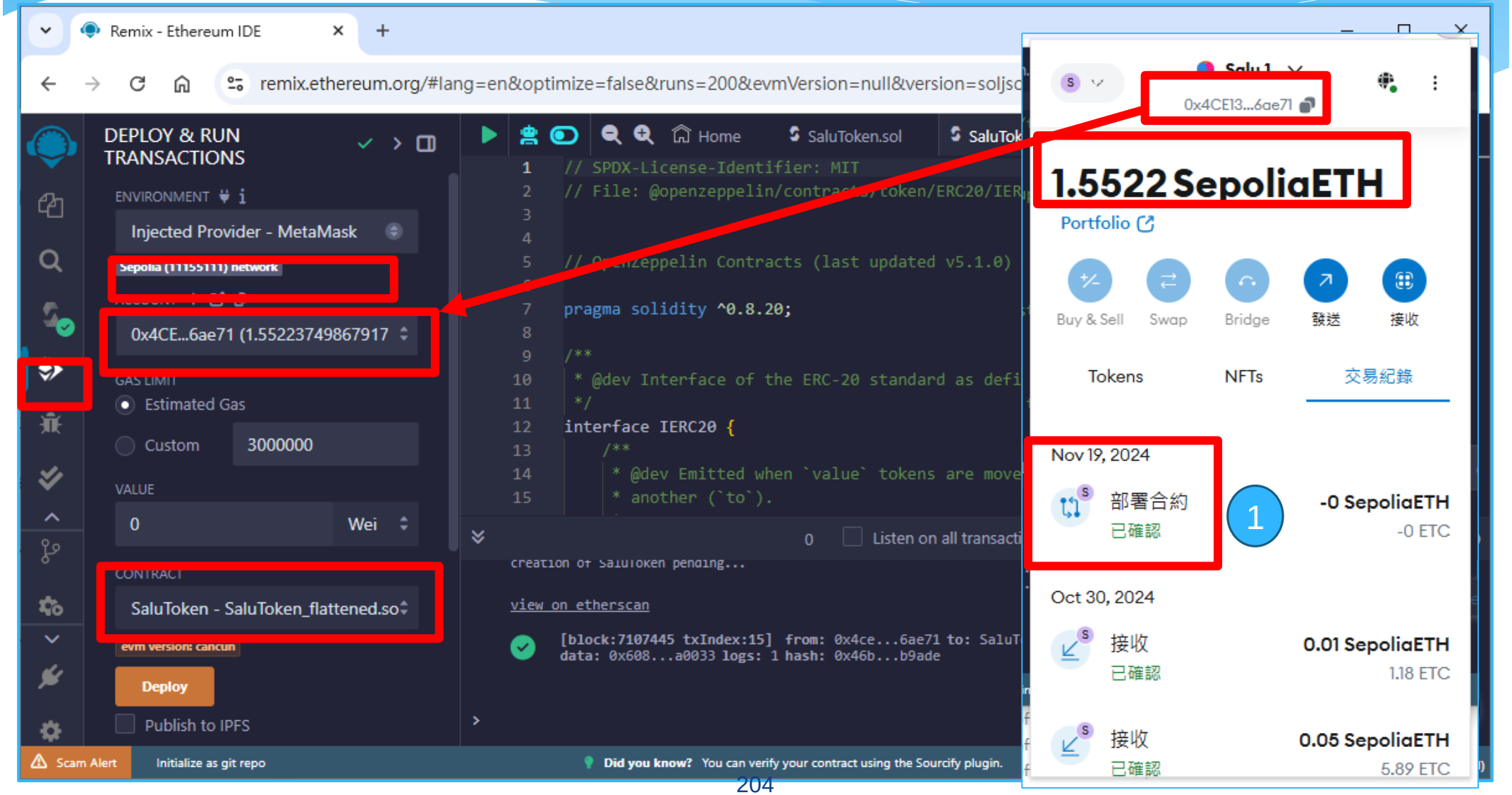
Compile Smart Contract



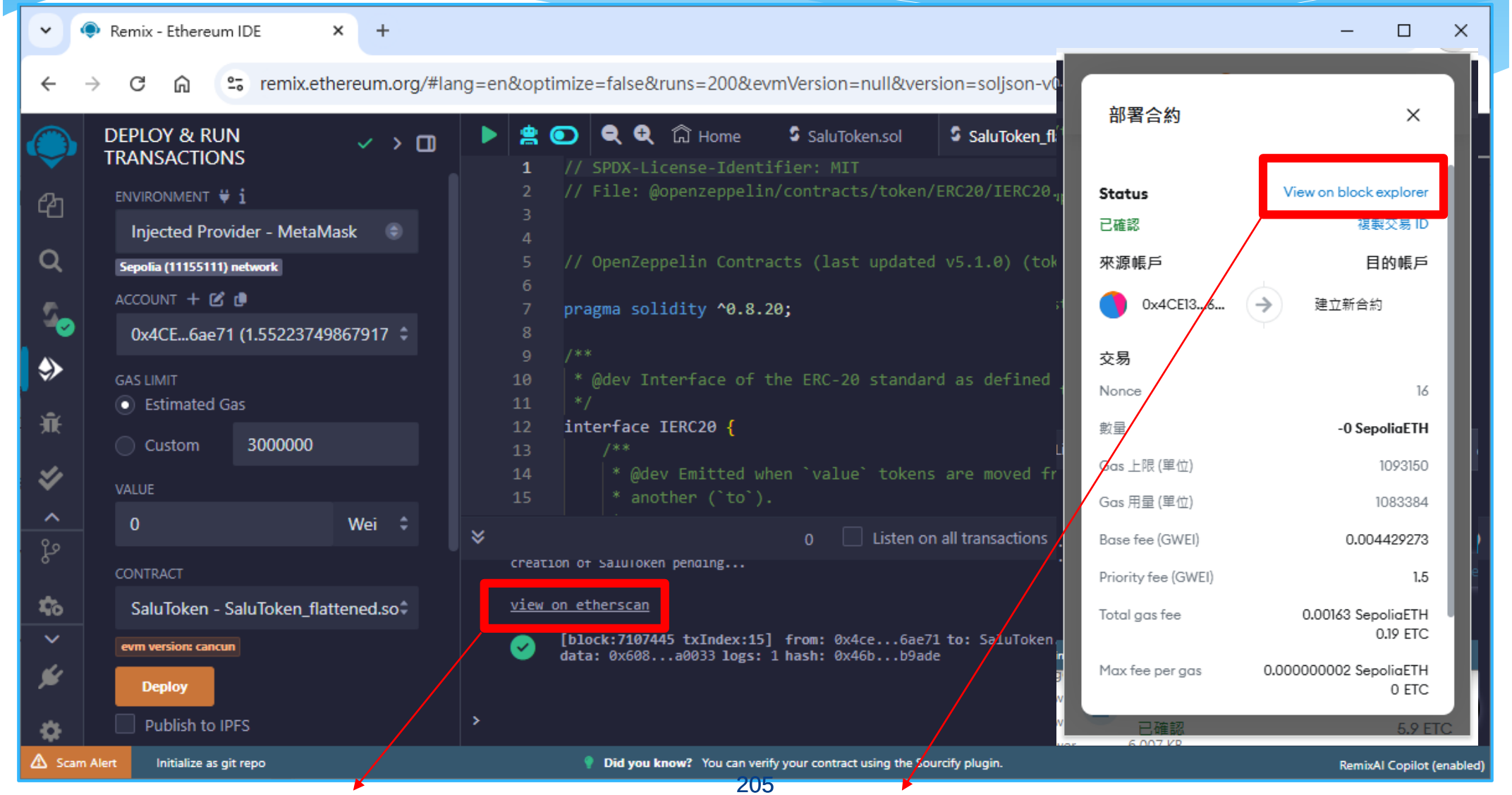
Deploy an ERC-20 Token



Deploy an ERC-20 Token

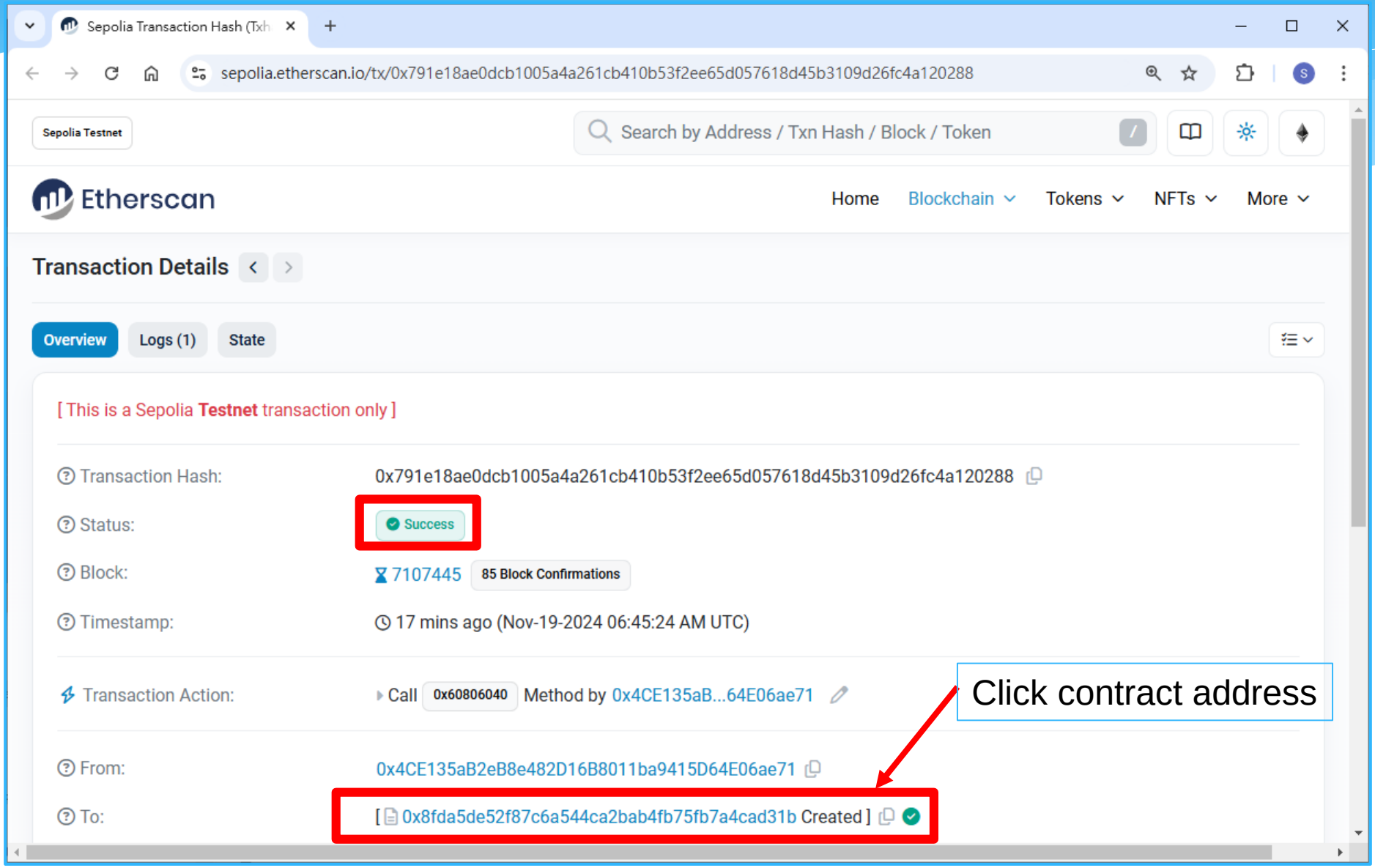


Deploy an ERC-20 Token



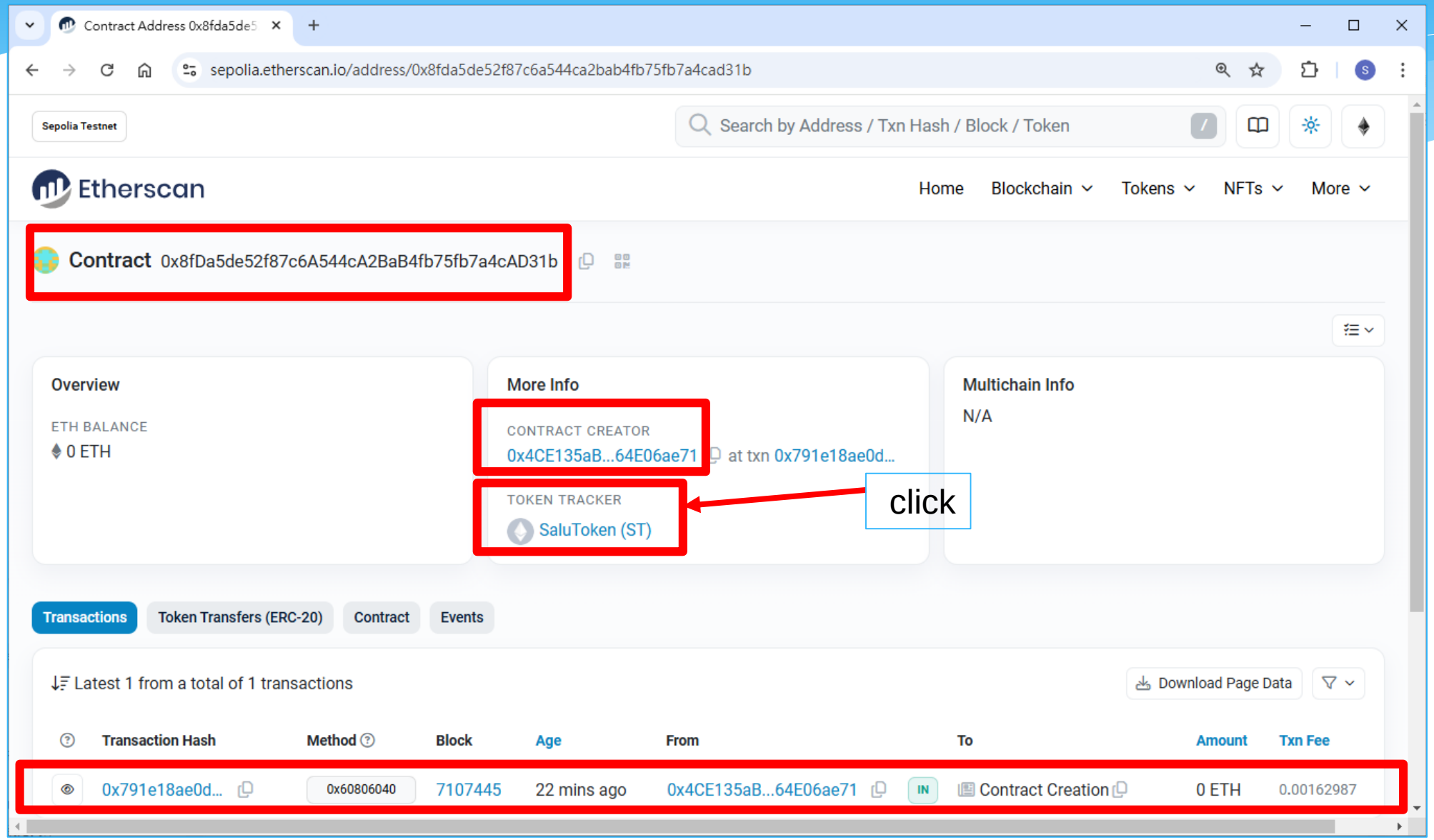
https://sepolia.etherscan.io/tx/0x791e18ae0dcb1005a4a261cb410b53f2ee65d057618d45b3109d26fc4a120288

View Transaction on Sepolia



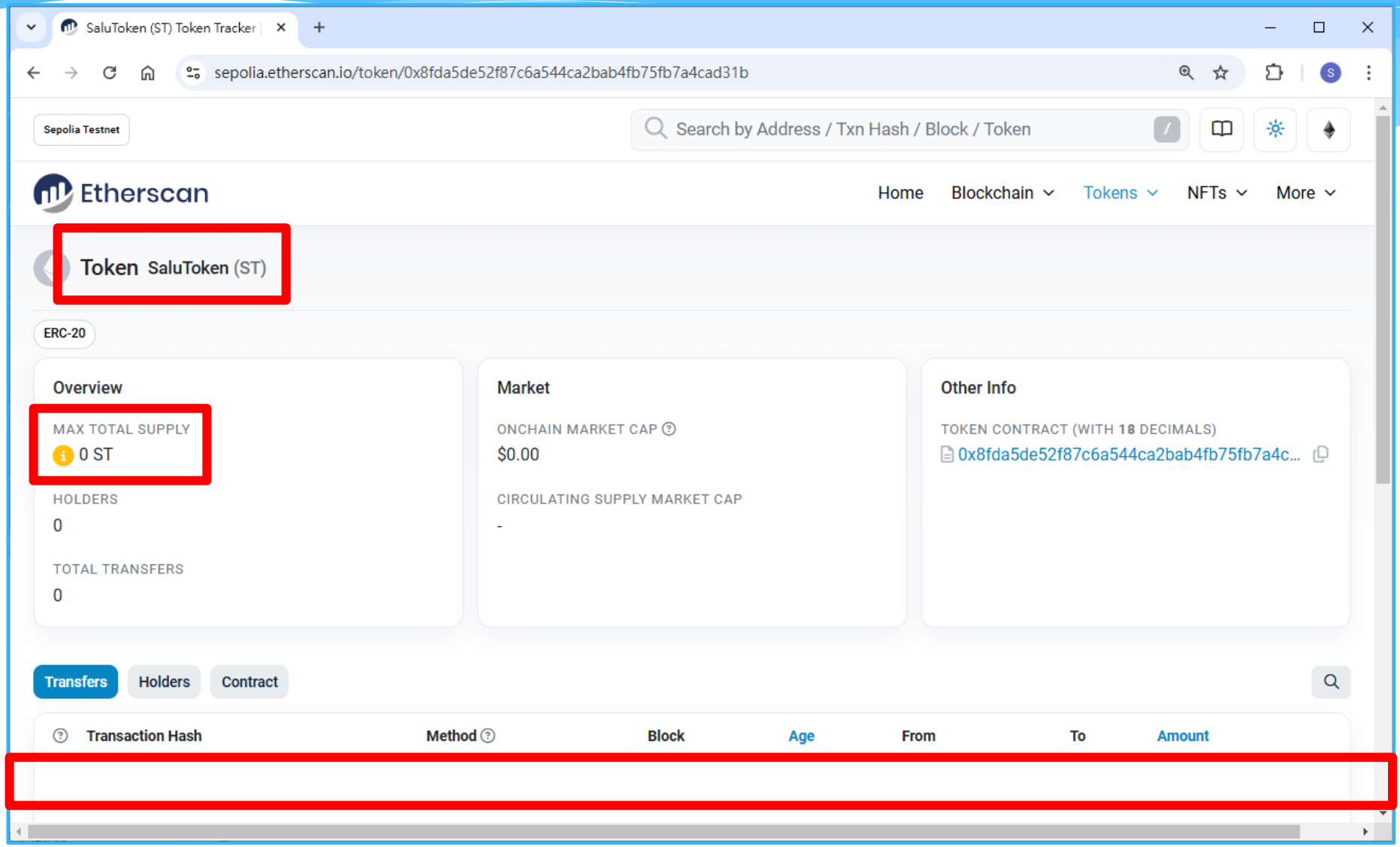
https://sepolia.etherscan.io/tx/0x791e18ae0dcb1005a4a261cb410b53f2ee65d057618d45b3109d26fc4a120288

Smart Contract Information



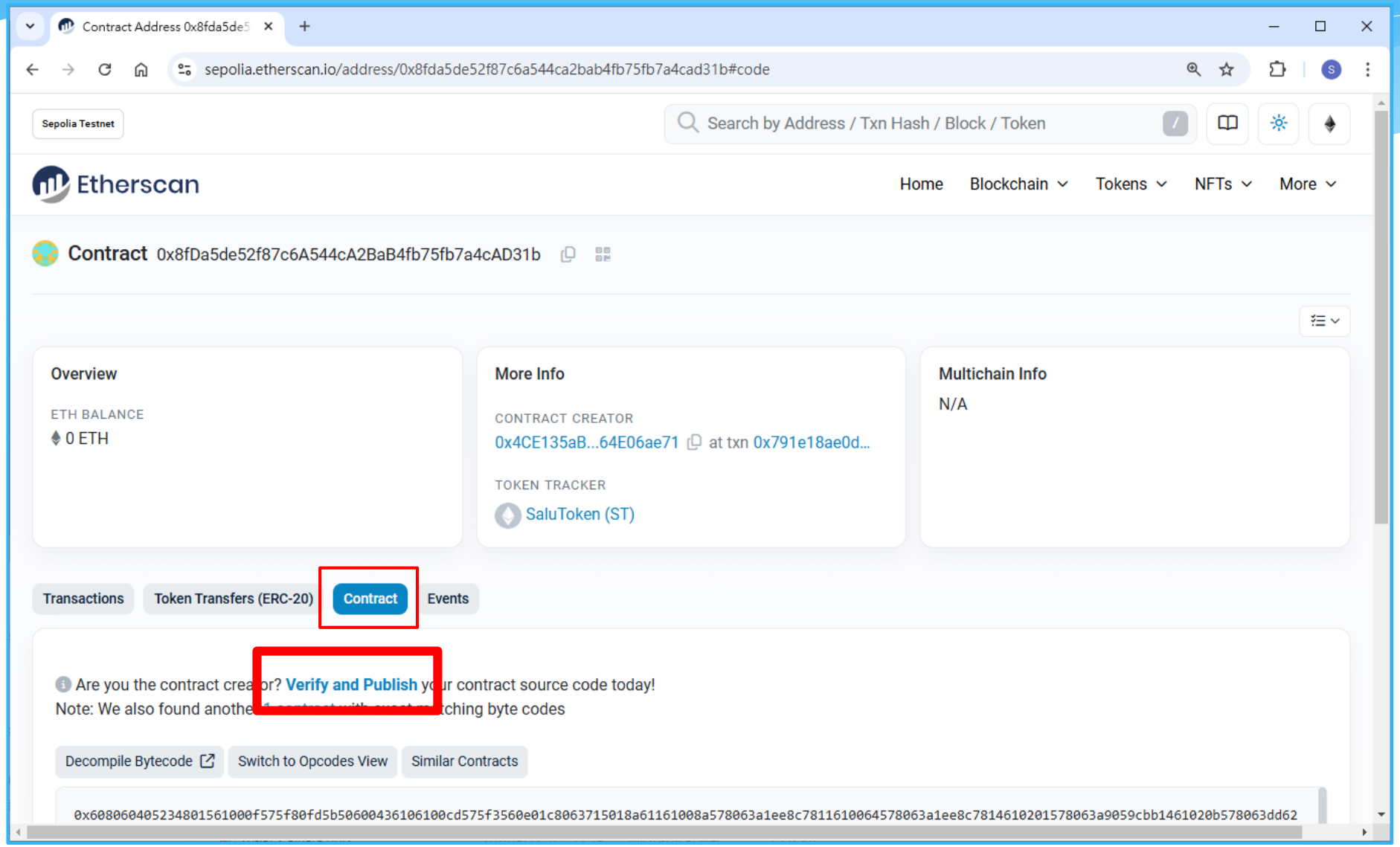
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b

Smart Contract Information



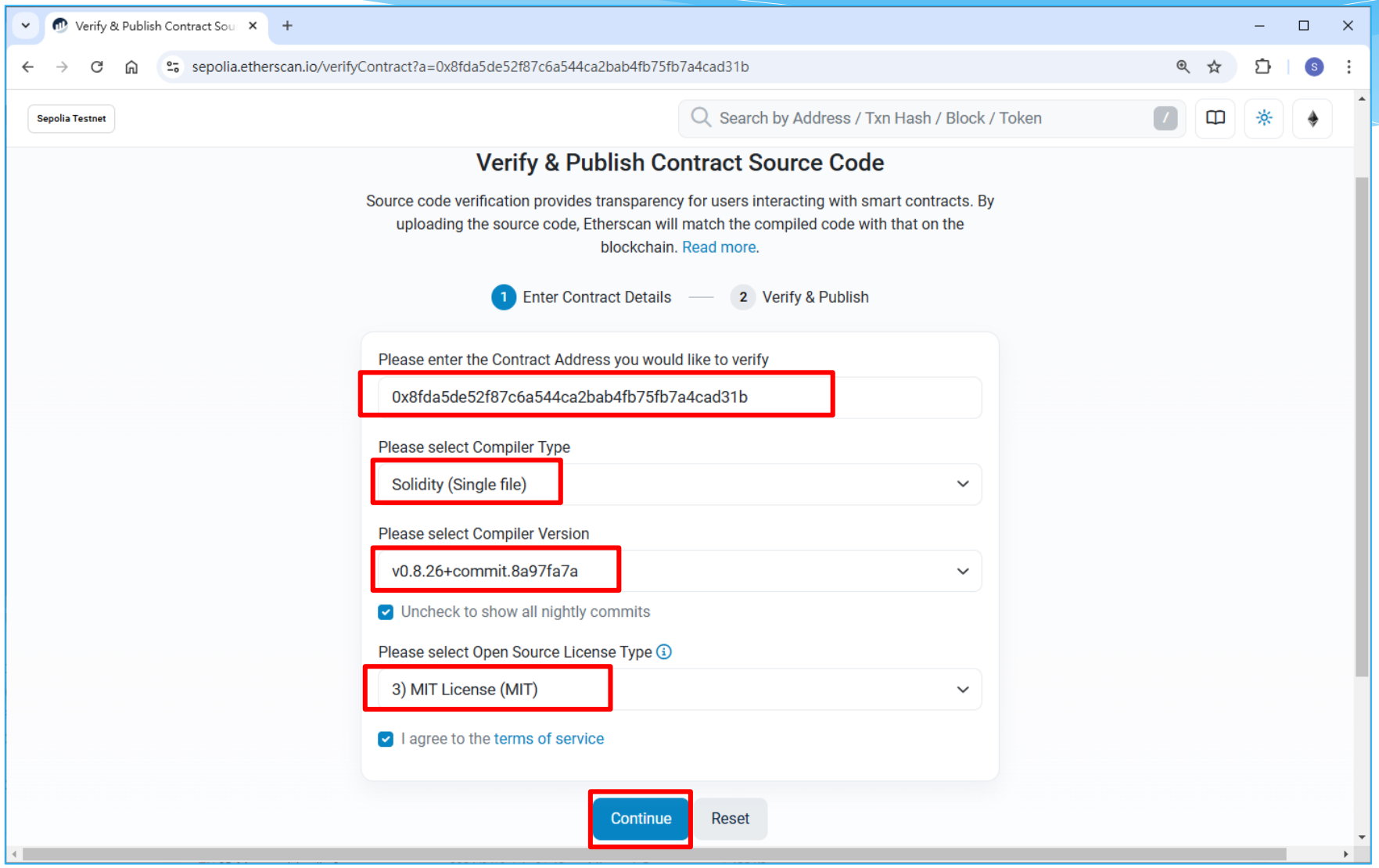
https://sepolia.etherscan.io/token/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b

View & Publish



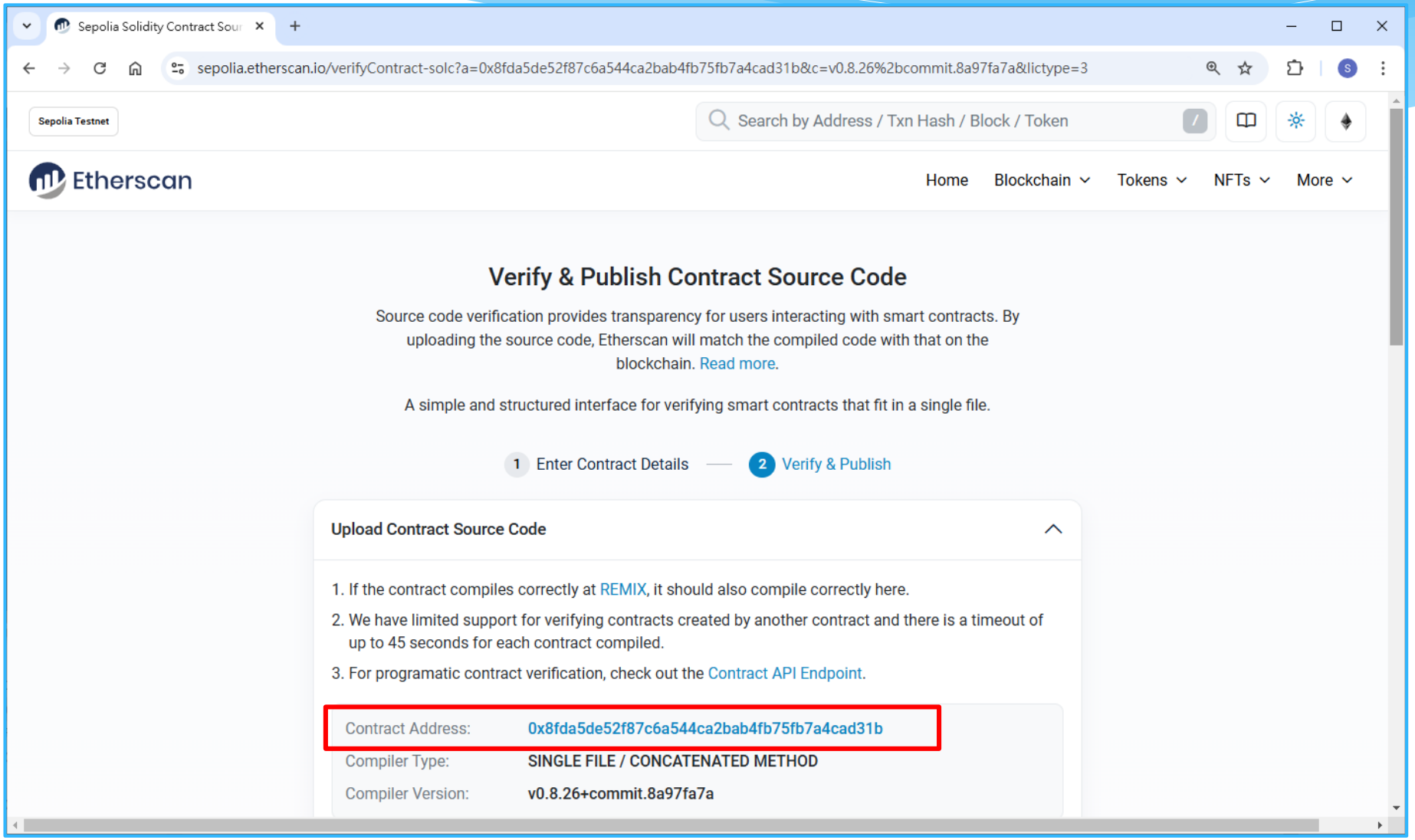
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#code

View & Publish Contract Source Code

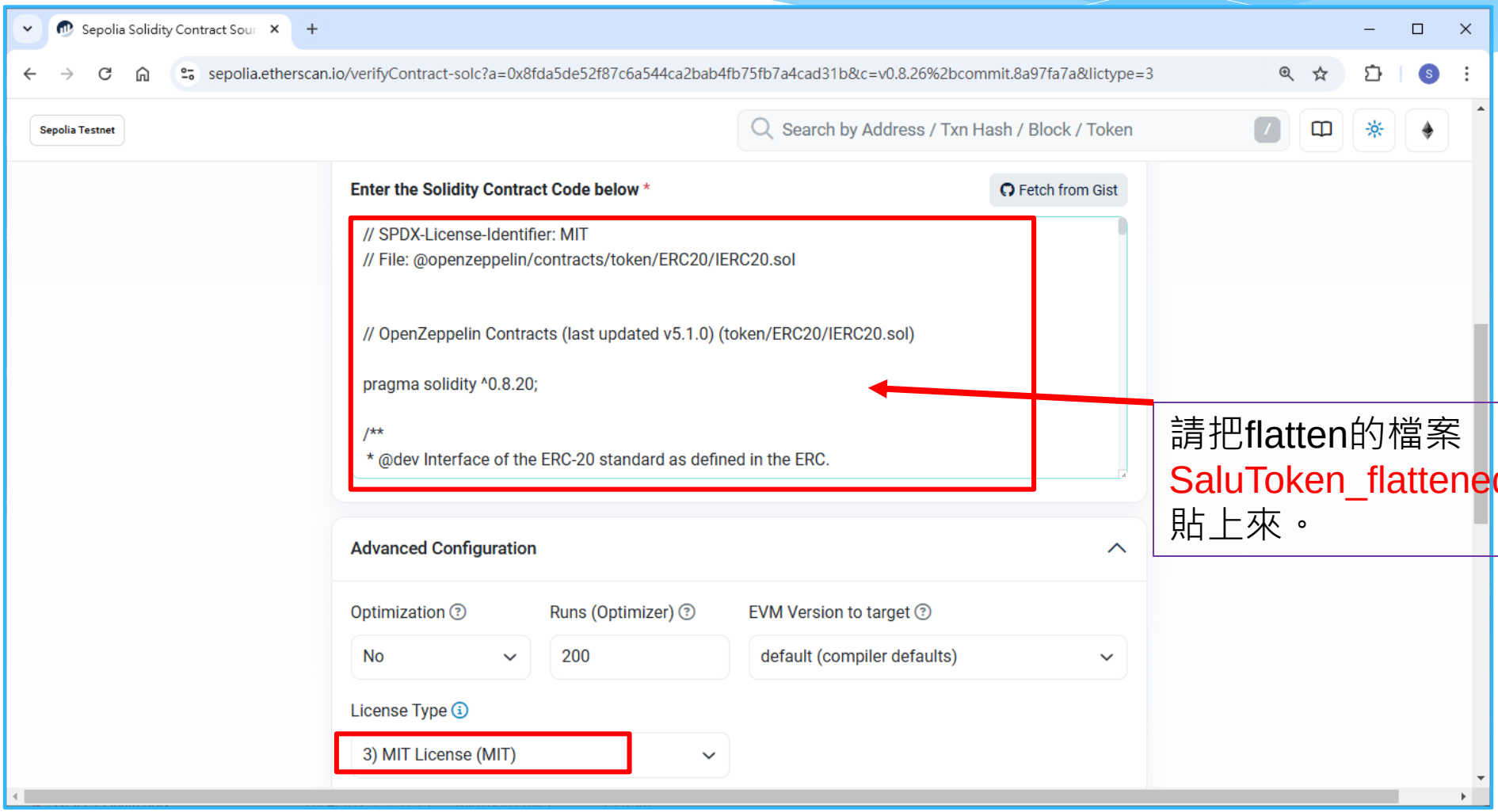


https://sepolia.etherscan.io/verifyContract?a=0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b

View & Publish Contract Source Code

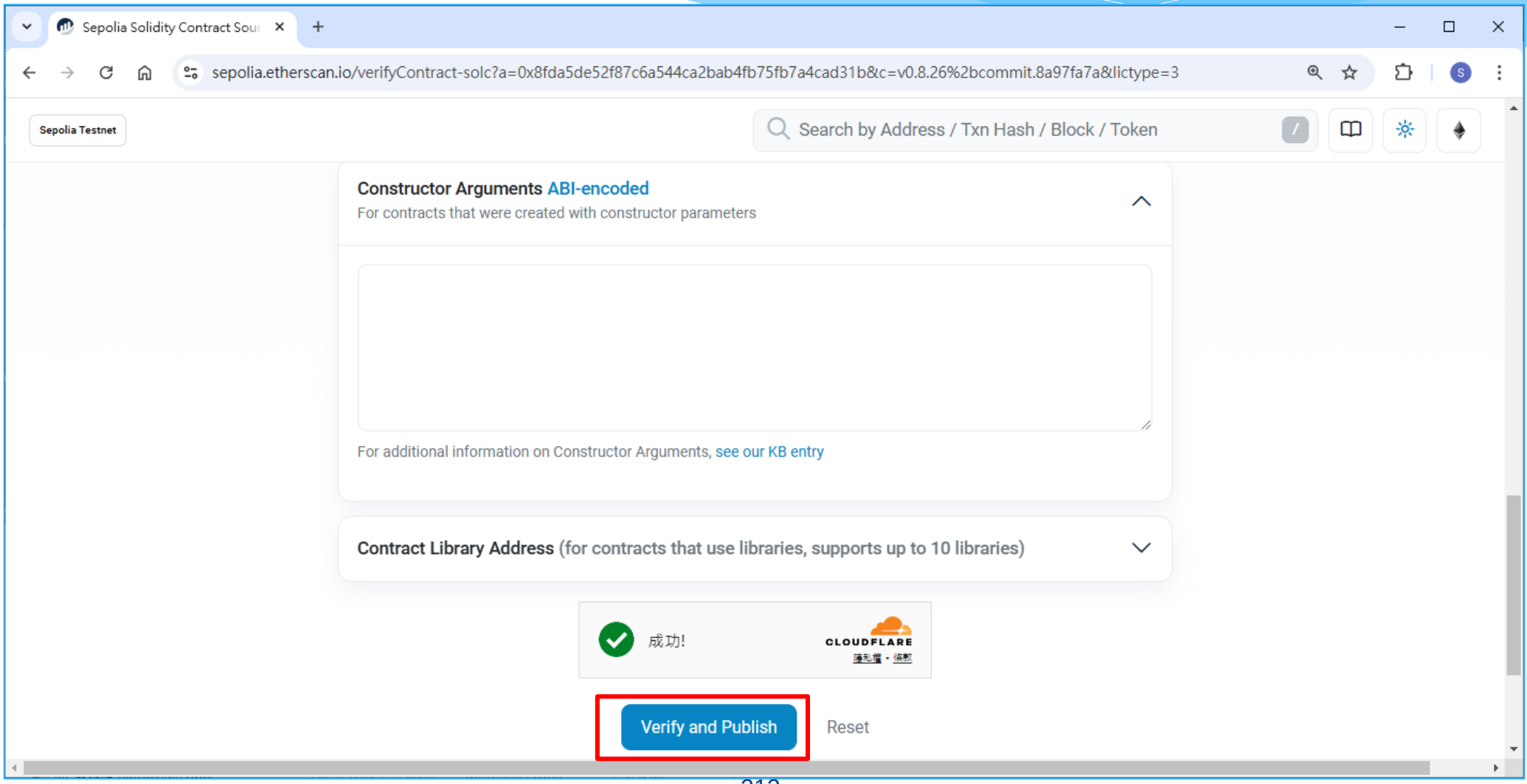


View & Publish Contract Source Code

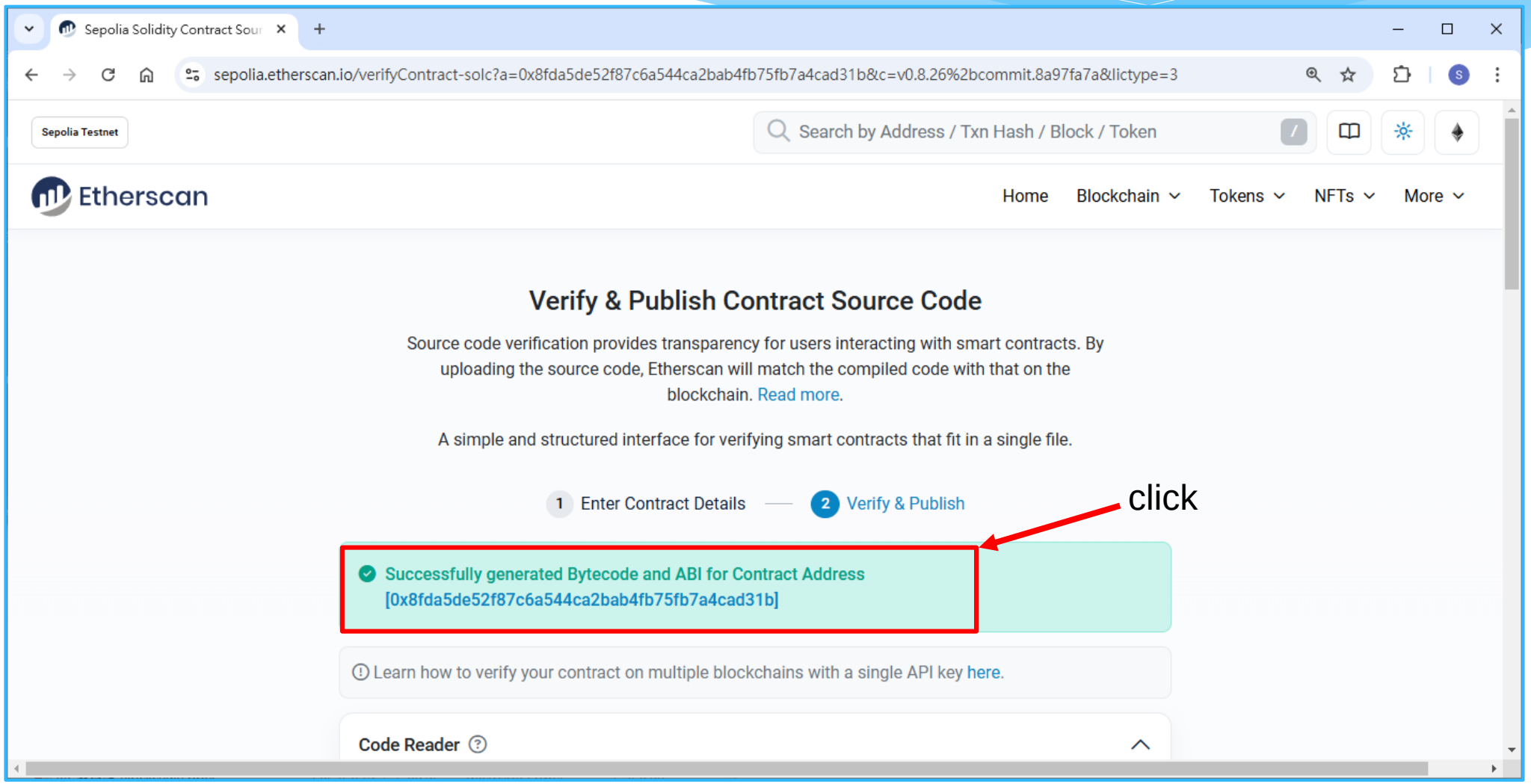


請把flatten的檔案 SaluToken_flattened.sol 貼上來。

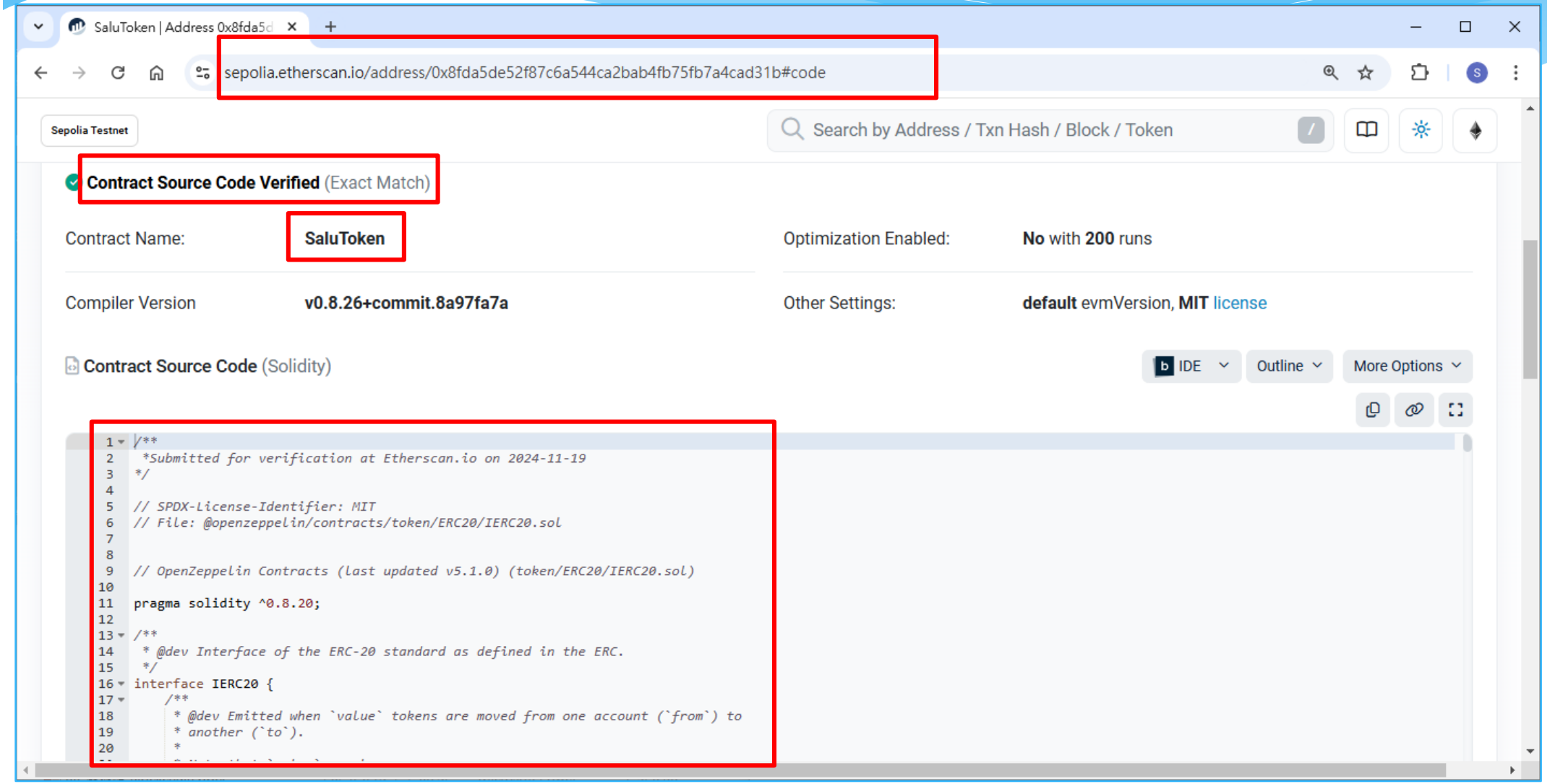
View & Publish Contract Source Code



View & Publish Contract Source Code

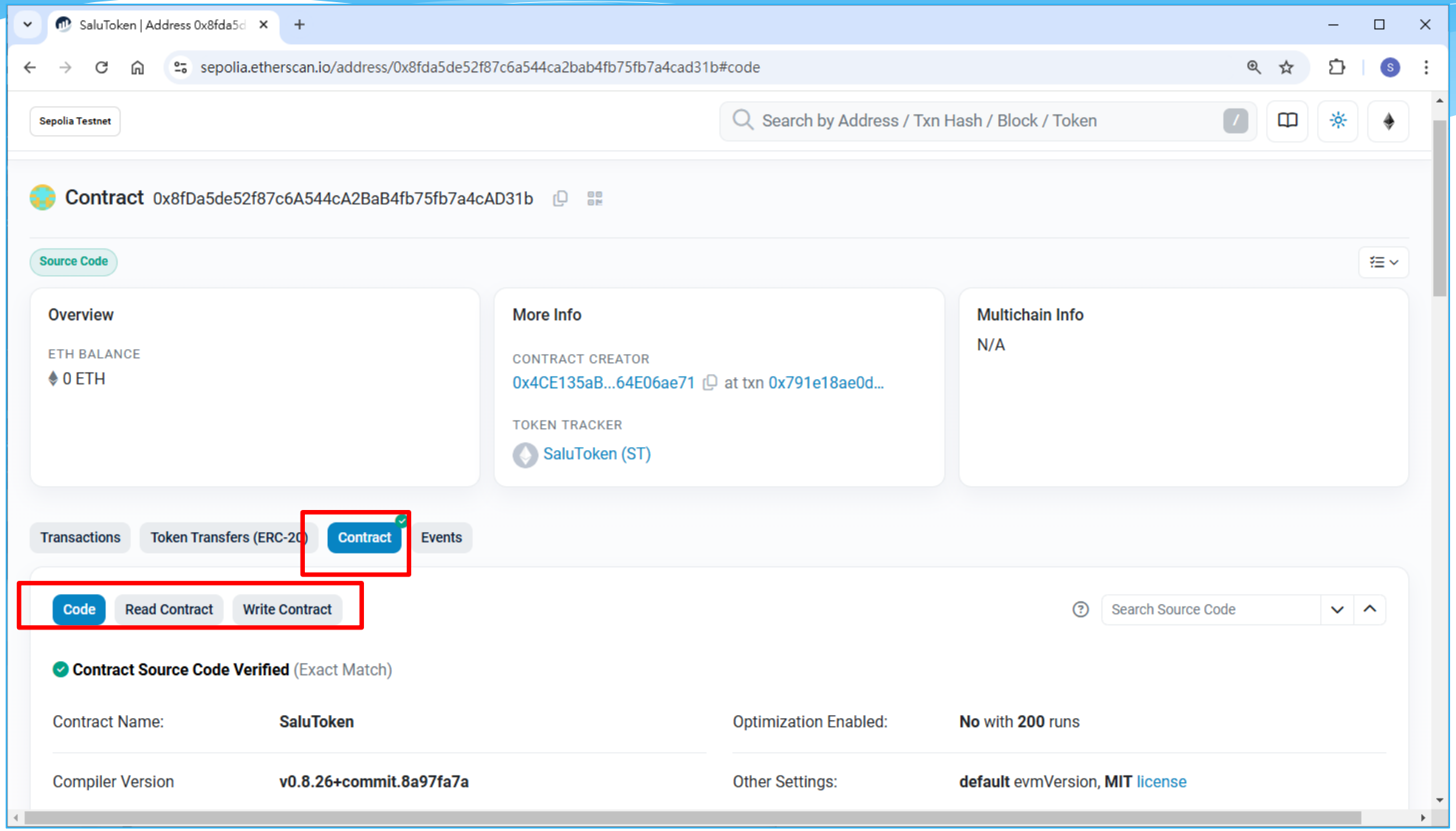


View Contract Source Code



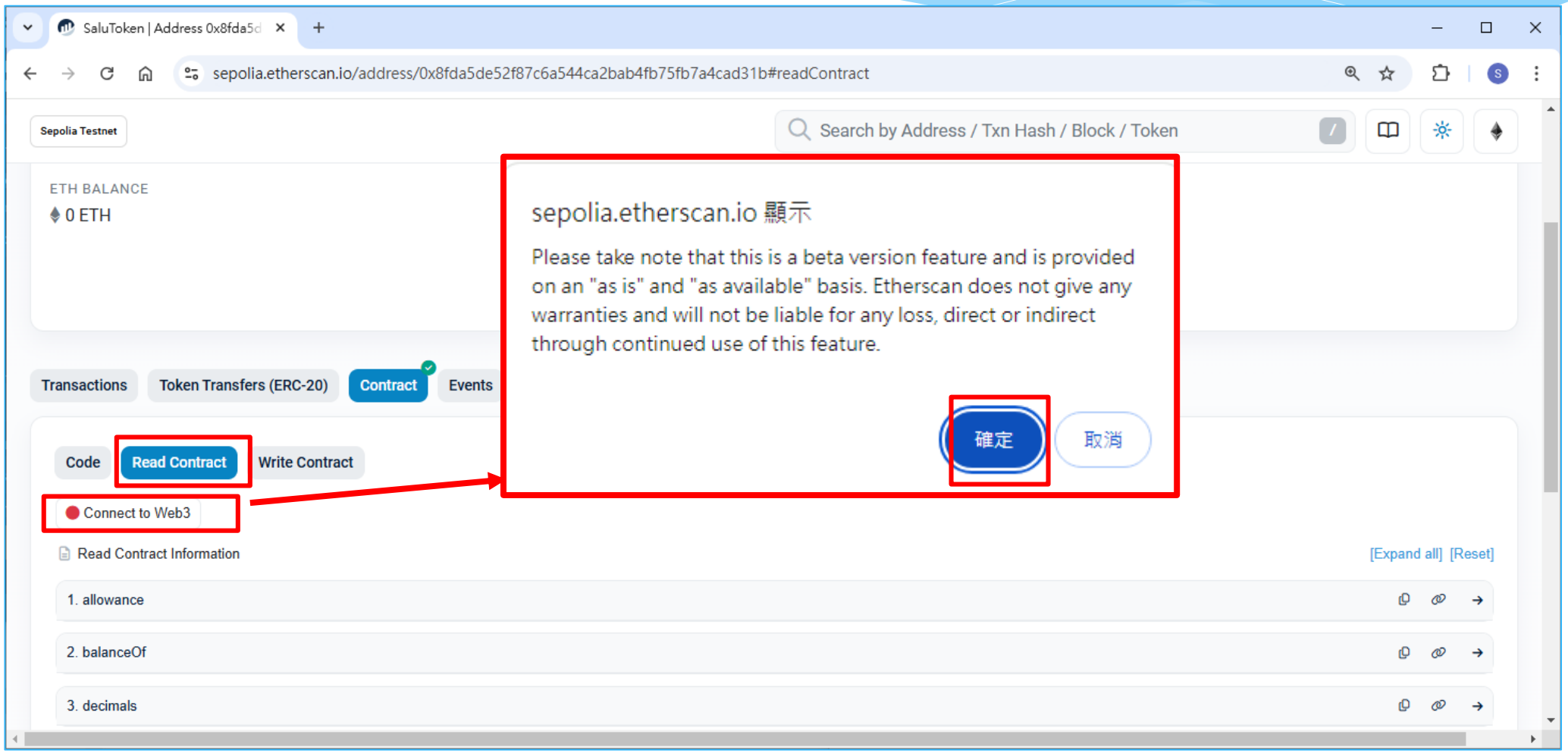
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#code

View & Publish Contract Source Code



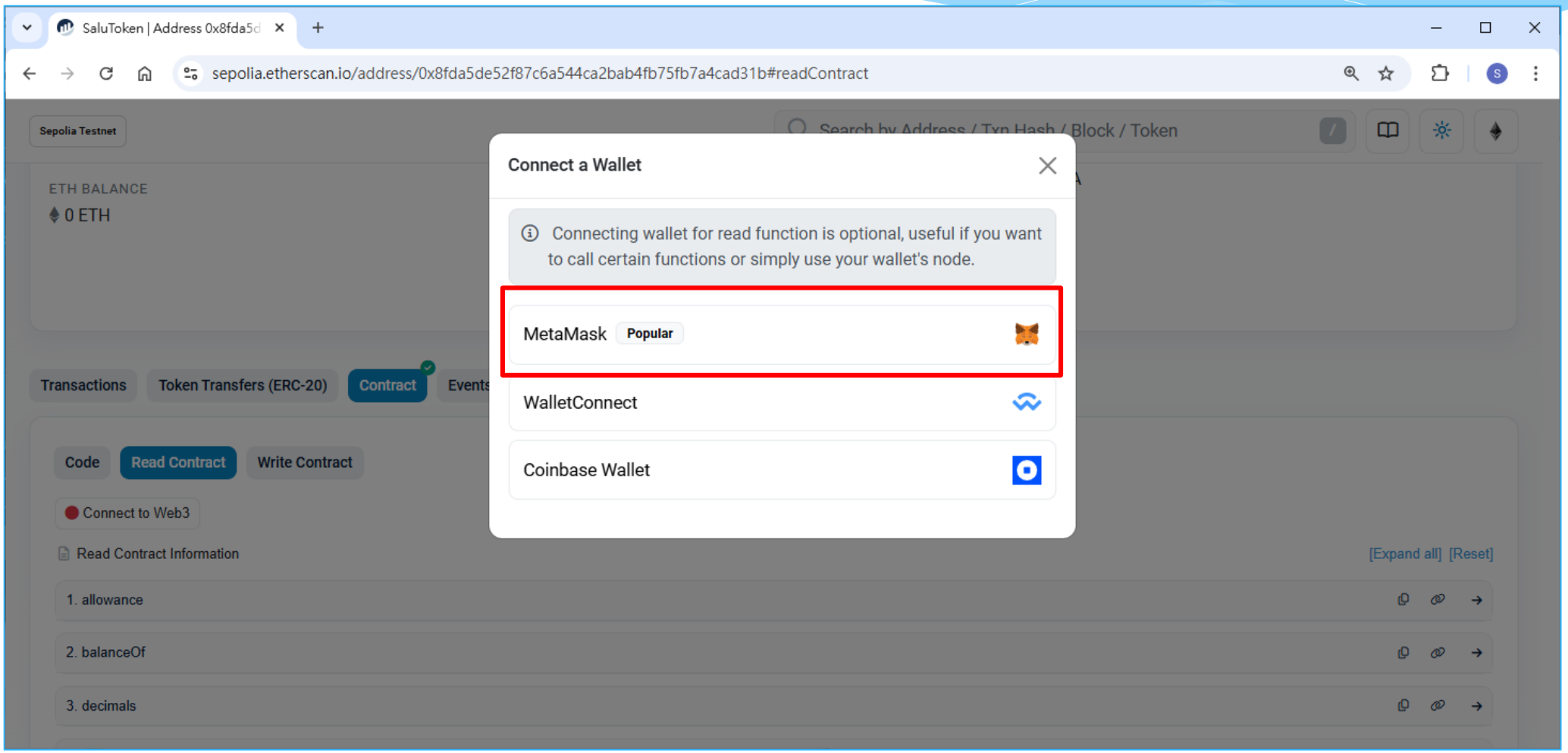
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#code

Connect to Web3



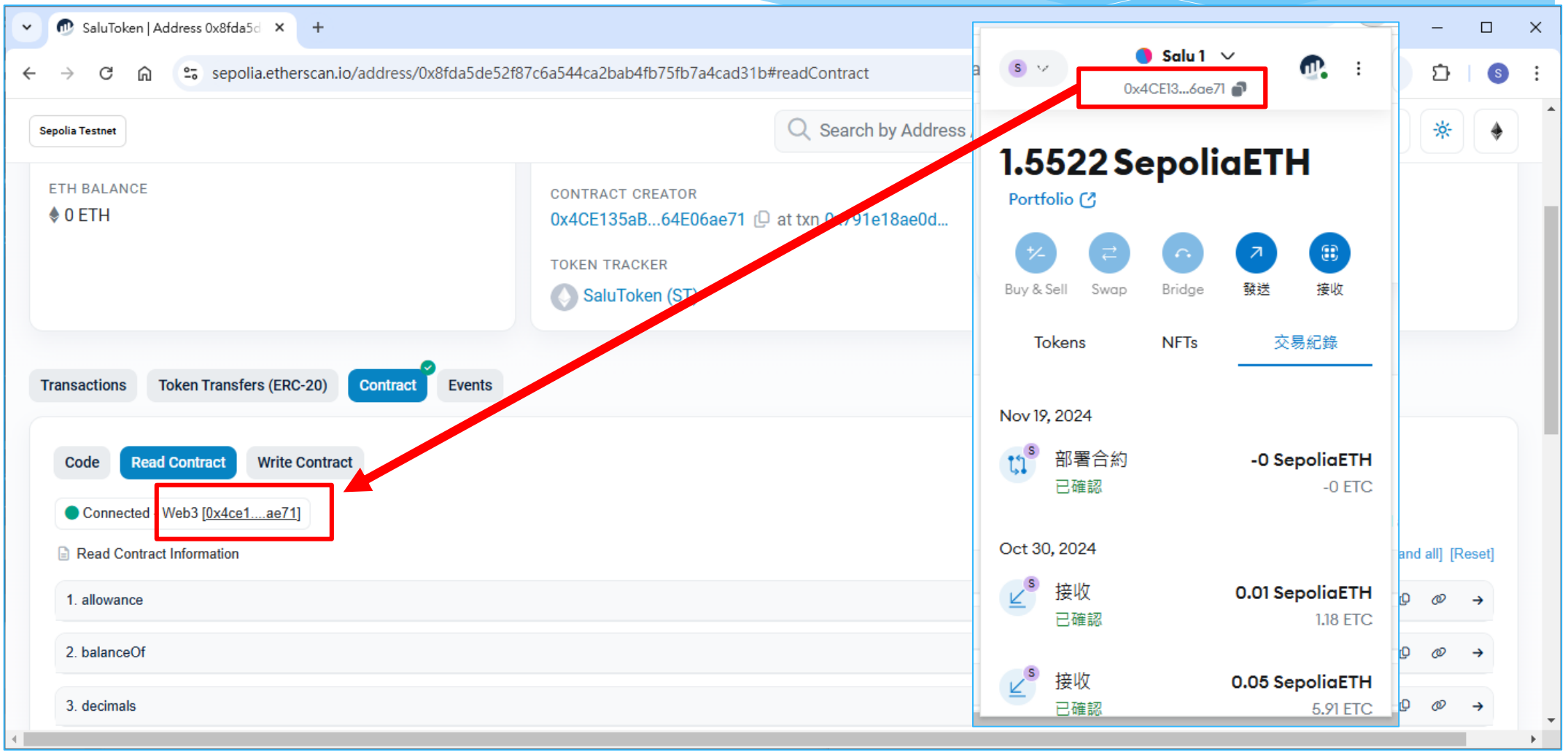
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#readContract

Connect a Wallet-MetaMask



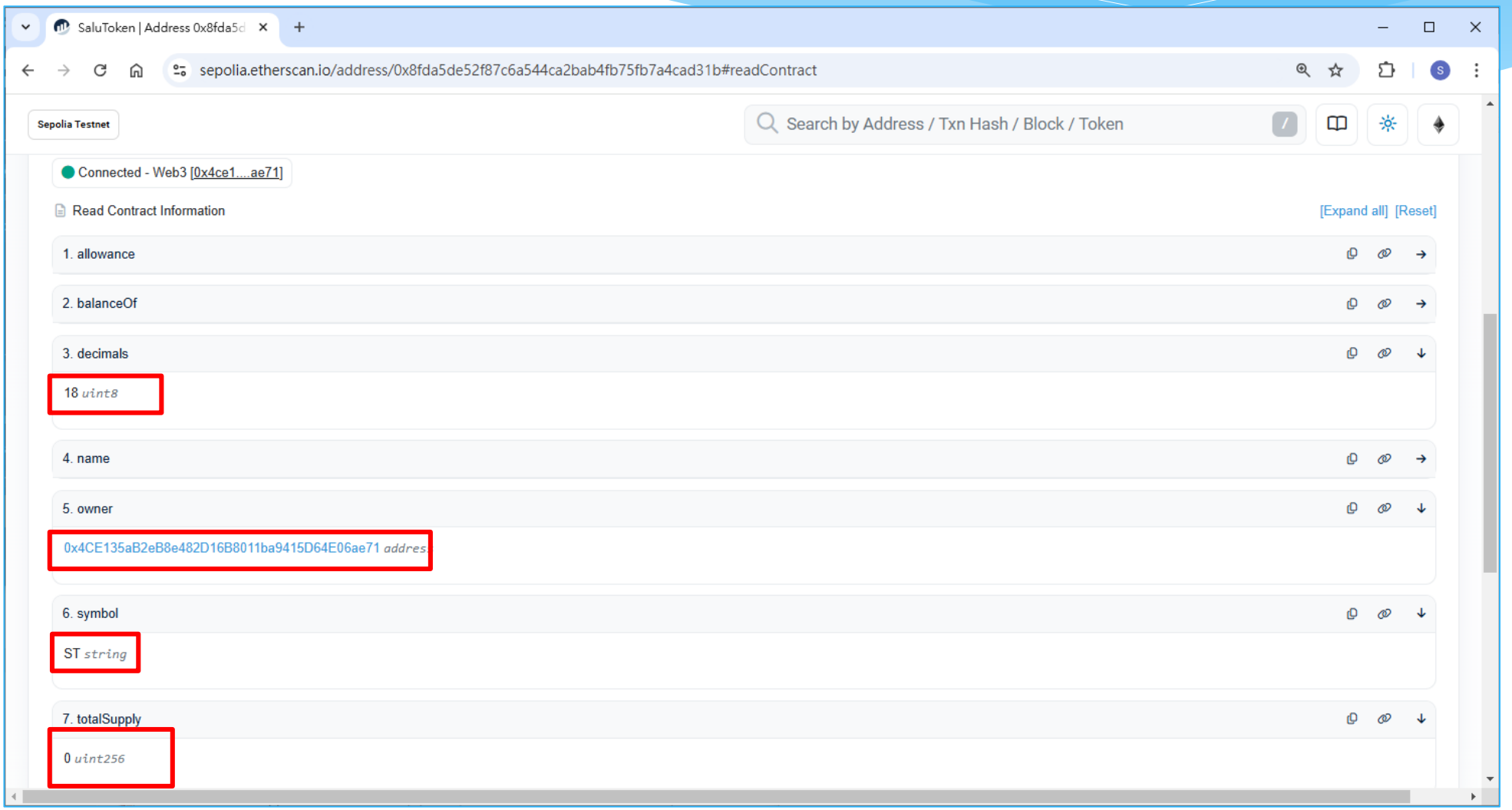
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#readContract

Connected Web3 using account



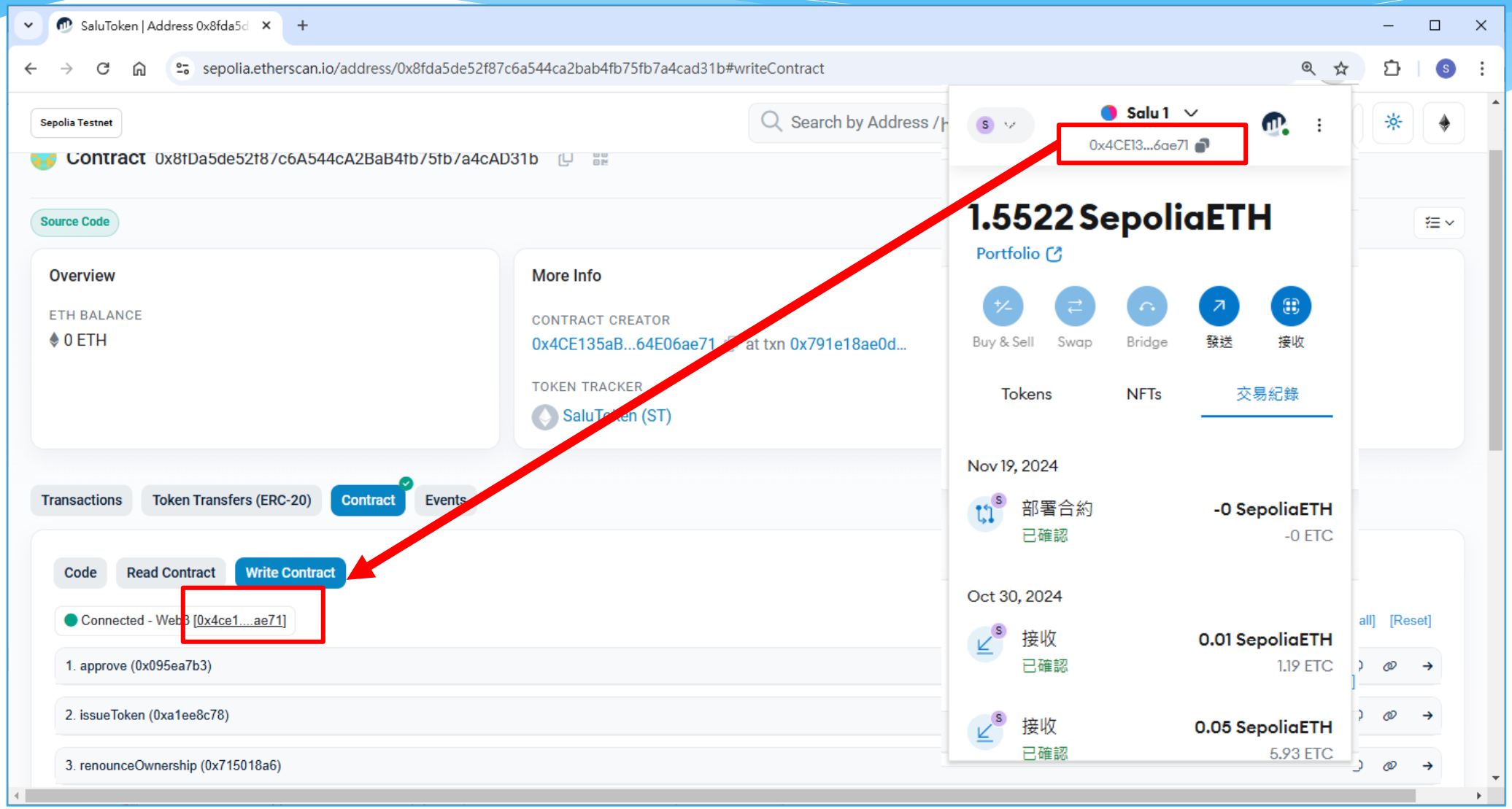
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#readContract

Read Contract



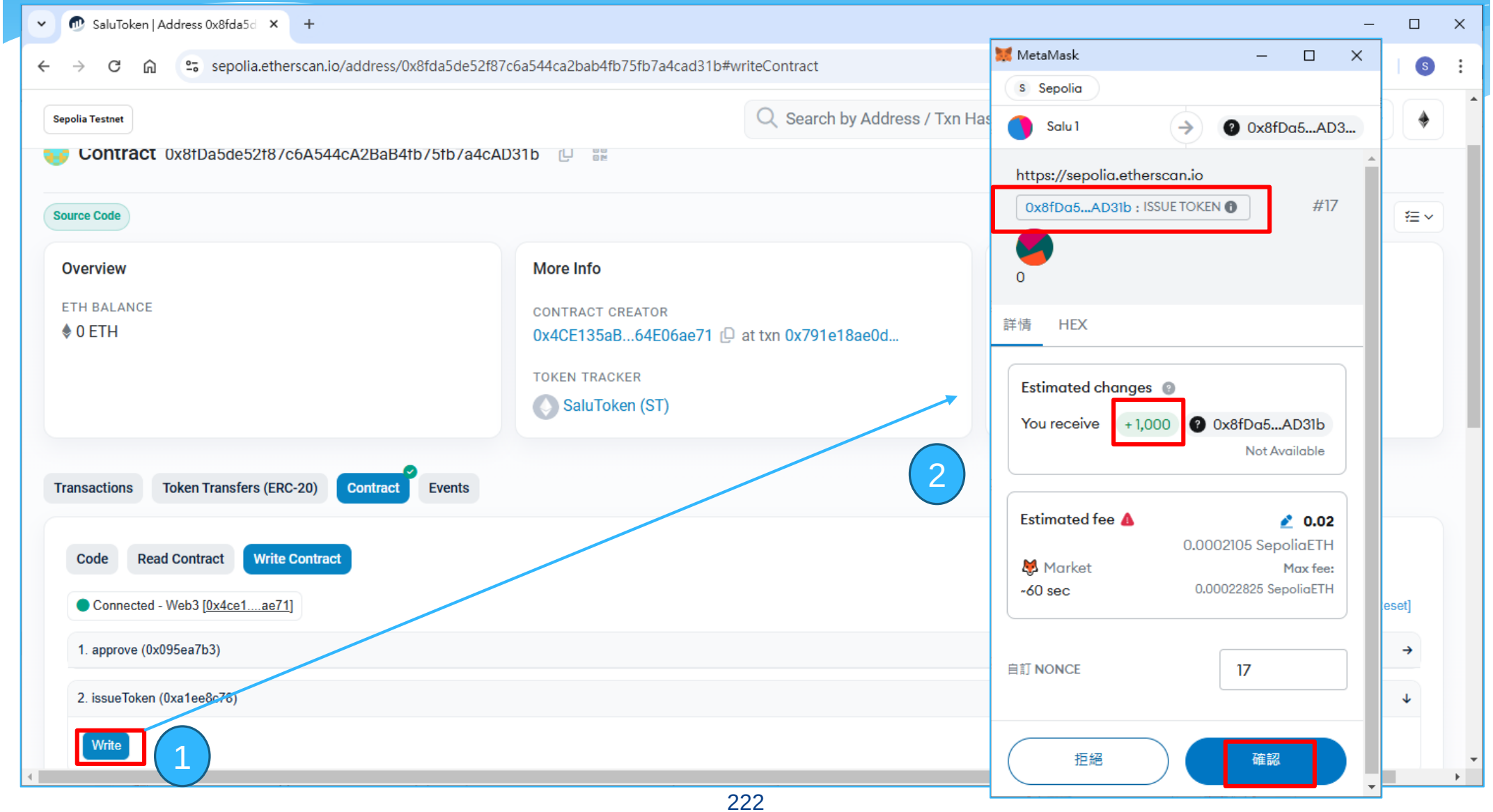
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#readContract

Connected Web3 using account



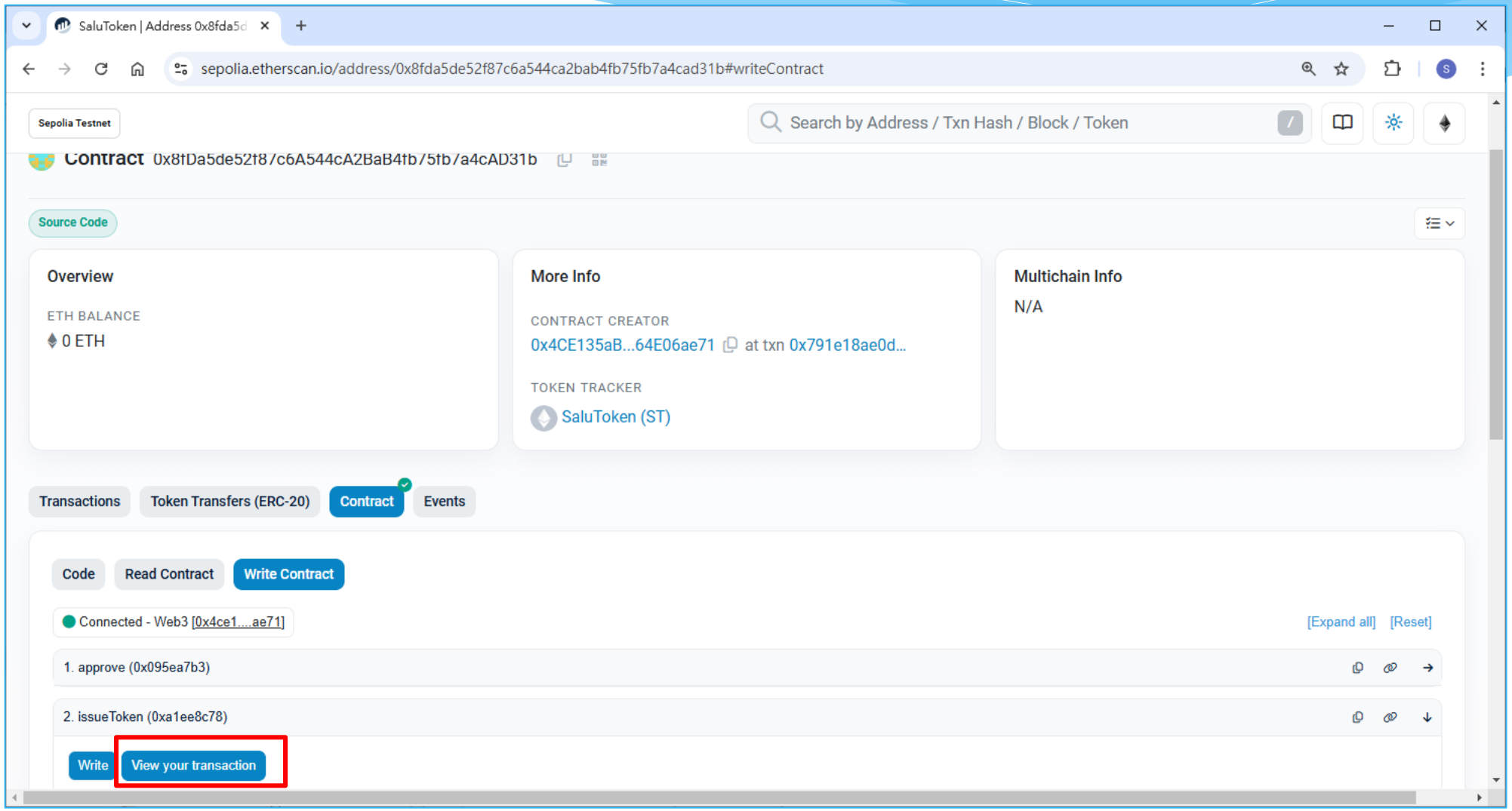
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#writeContract

Write Contract - issueToken



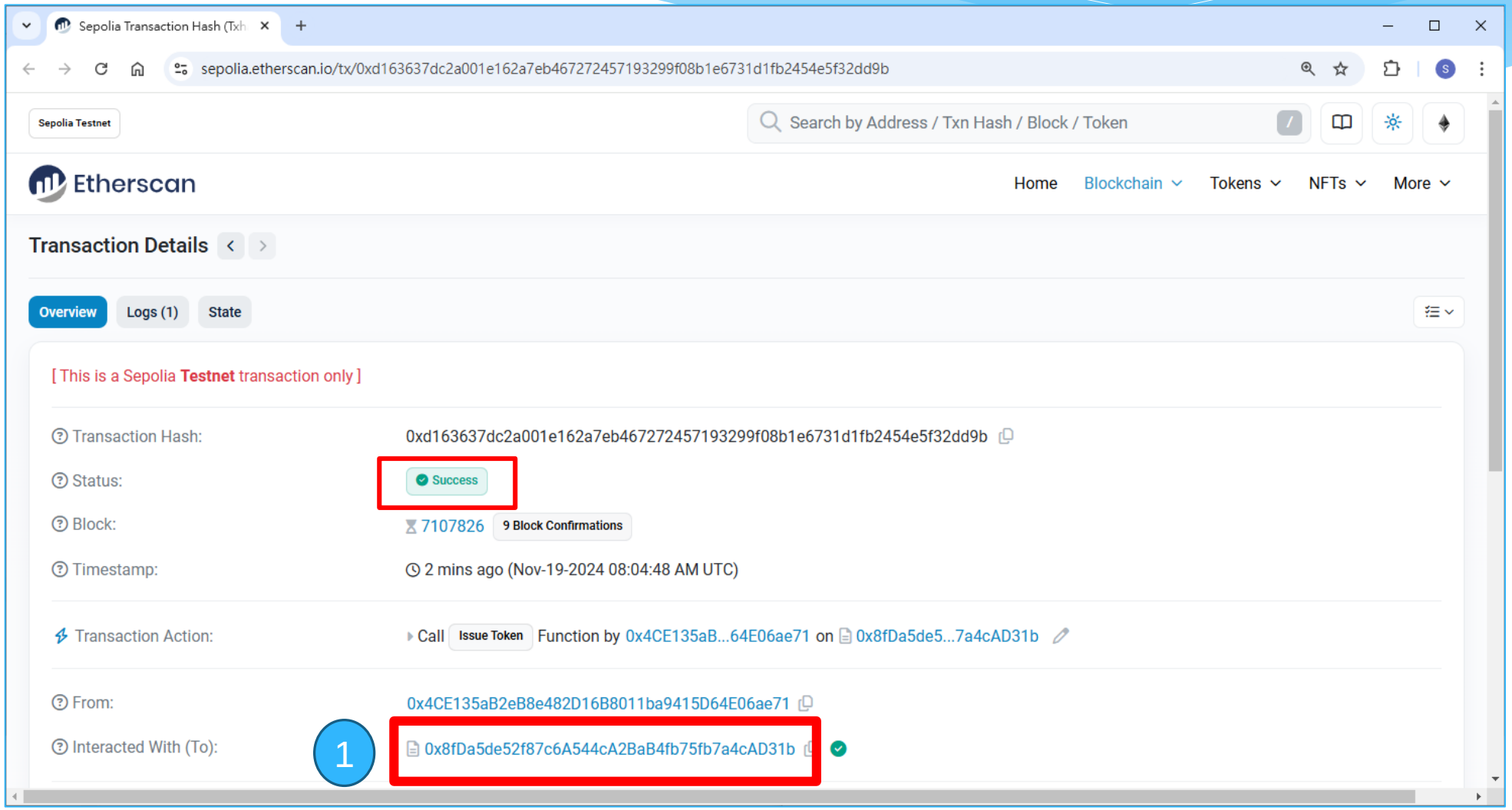
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#writeContract

Write Contract - issueToken



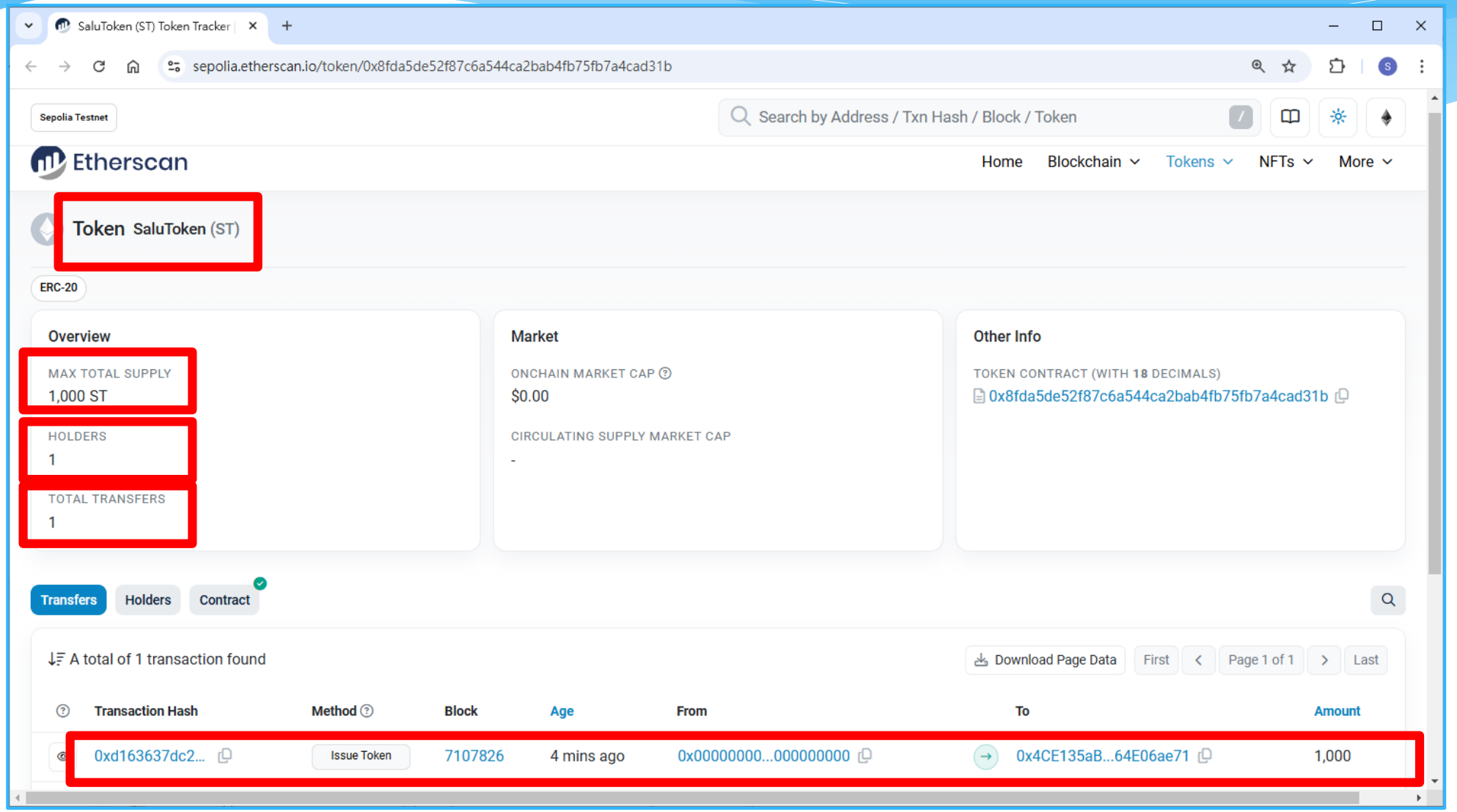
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#writeContract

View Transaction



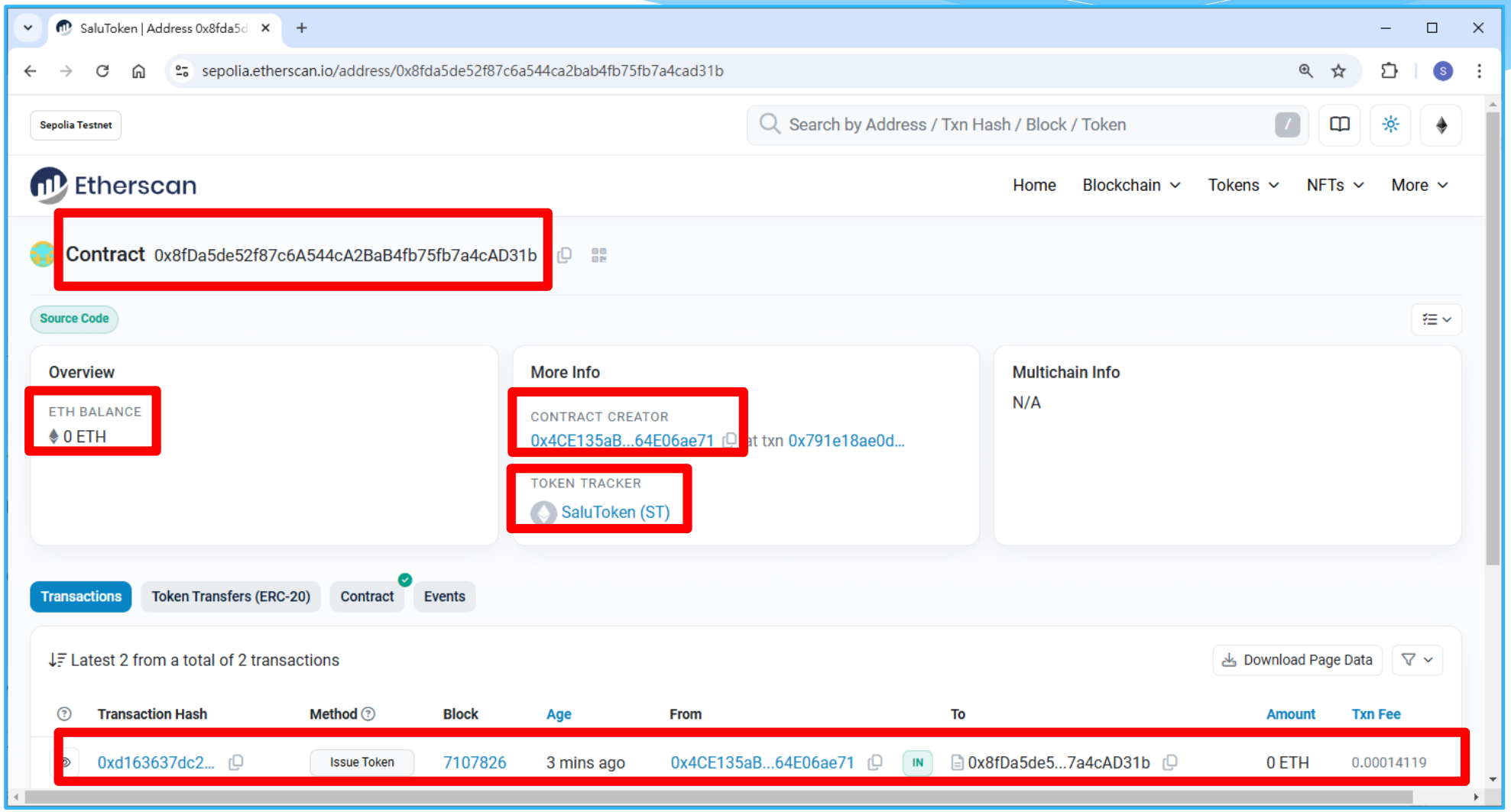
https://sepolia.etherscan.io/tx/0xd163637dc2a001e162a7eb467272457193299f08b1e6731d1fb2454e5f32dd9b

Smart Contract Information



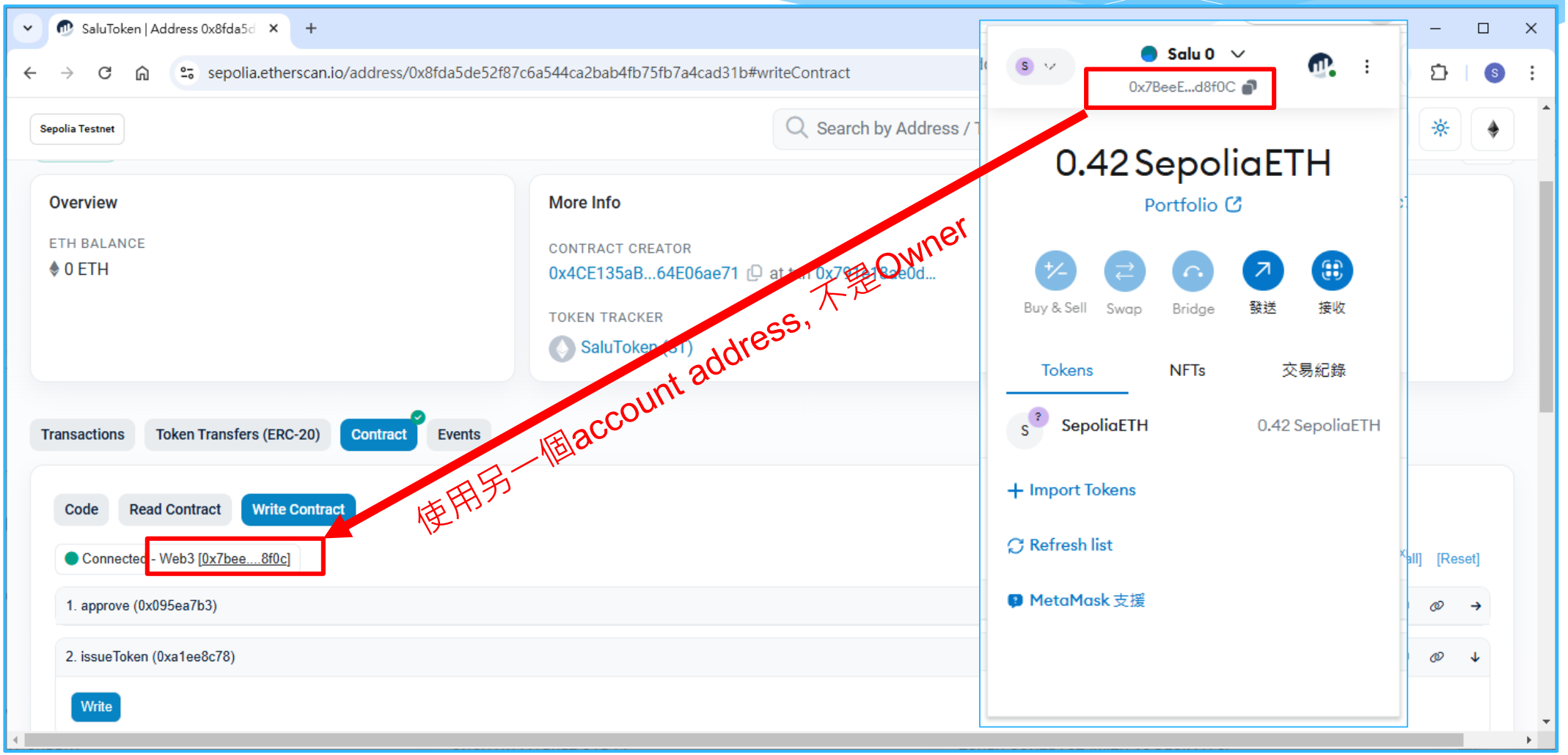
https://sepolia.etherscan.io/token/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b

Smart Contract Information



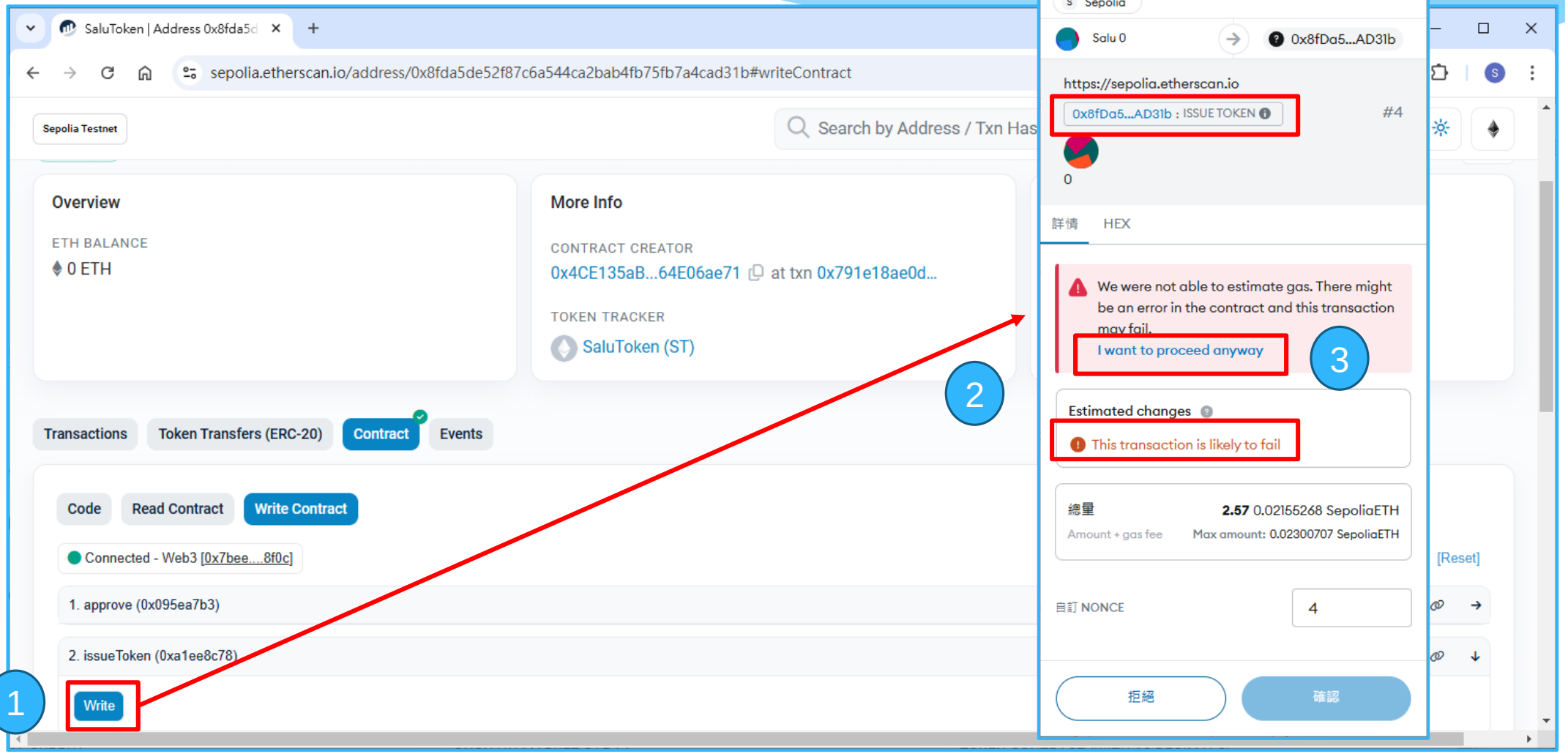
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b

Connected Web3 using account



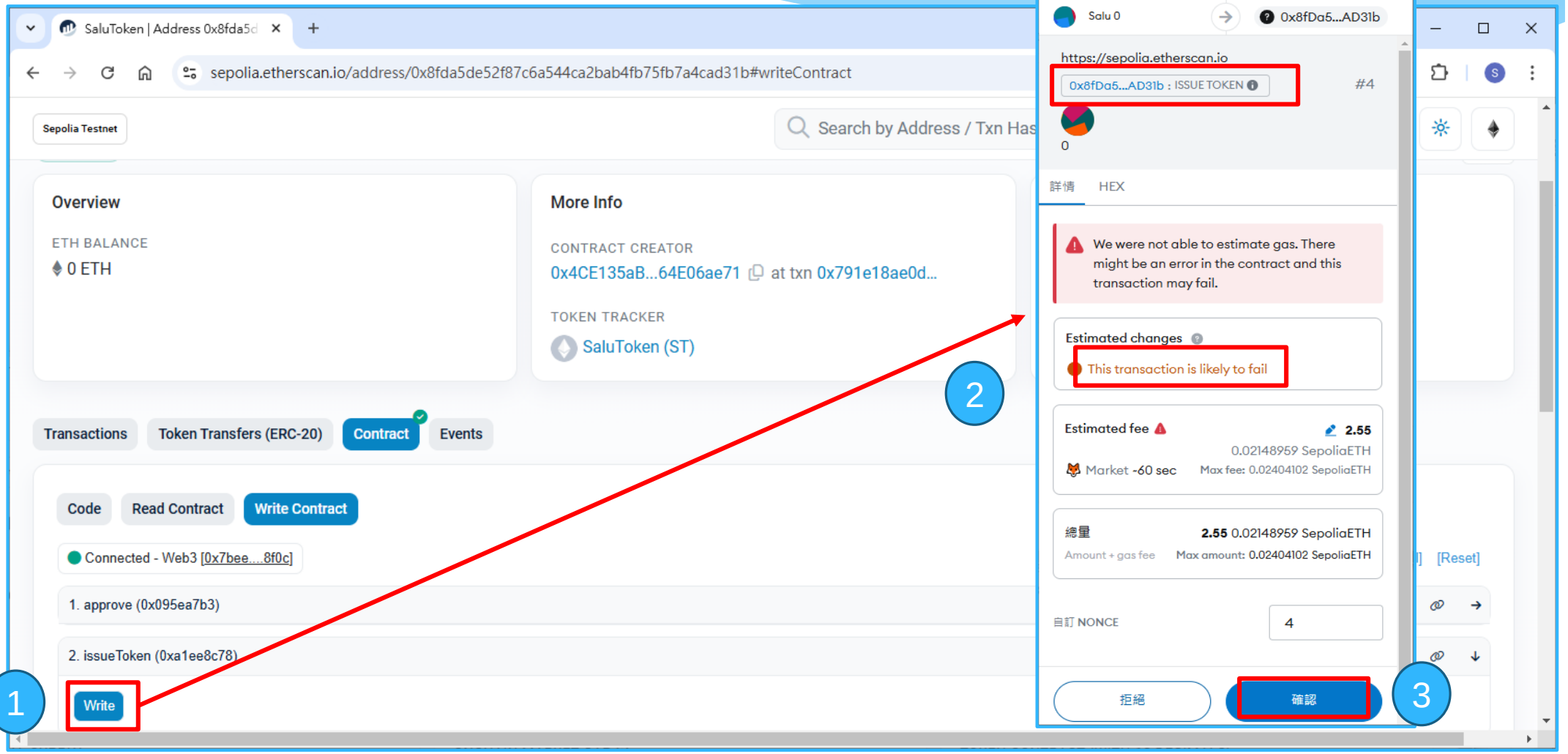
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#writeContract

Write Contract - issueToken



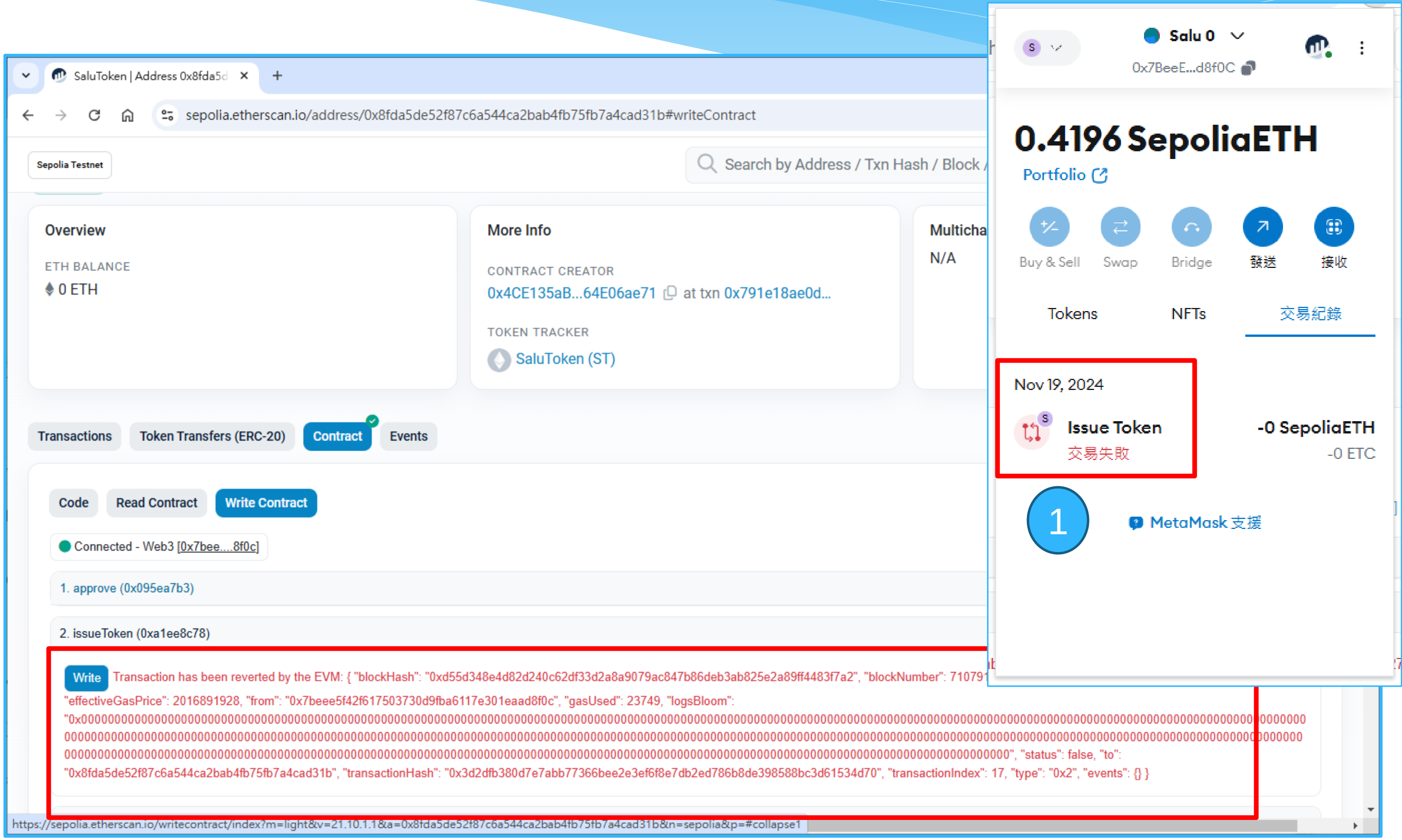
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#writeContract

Write Contract - issueToken



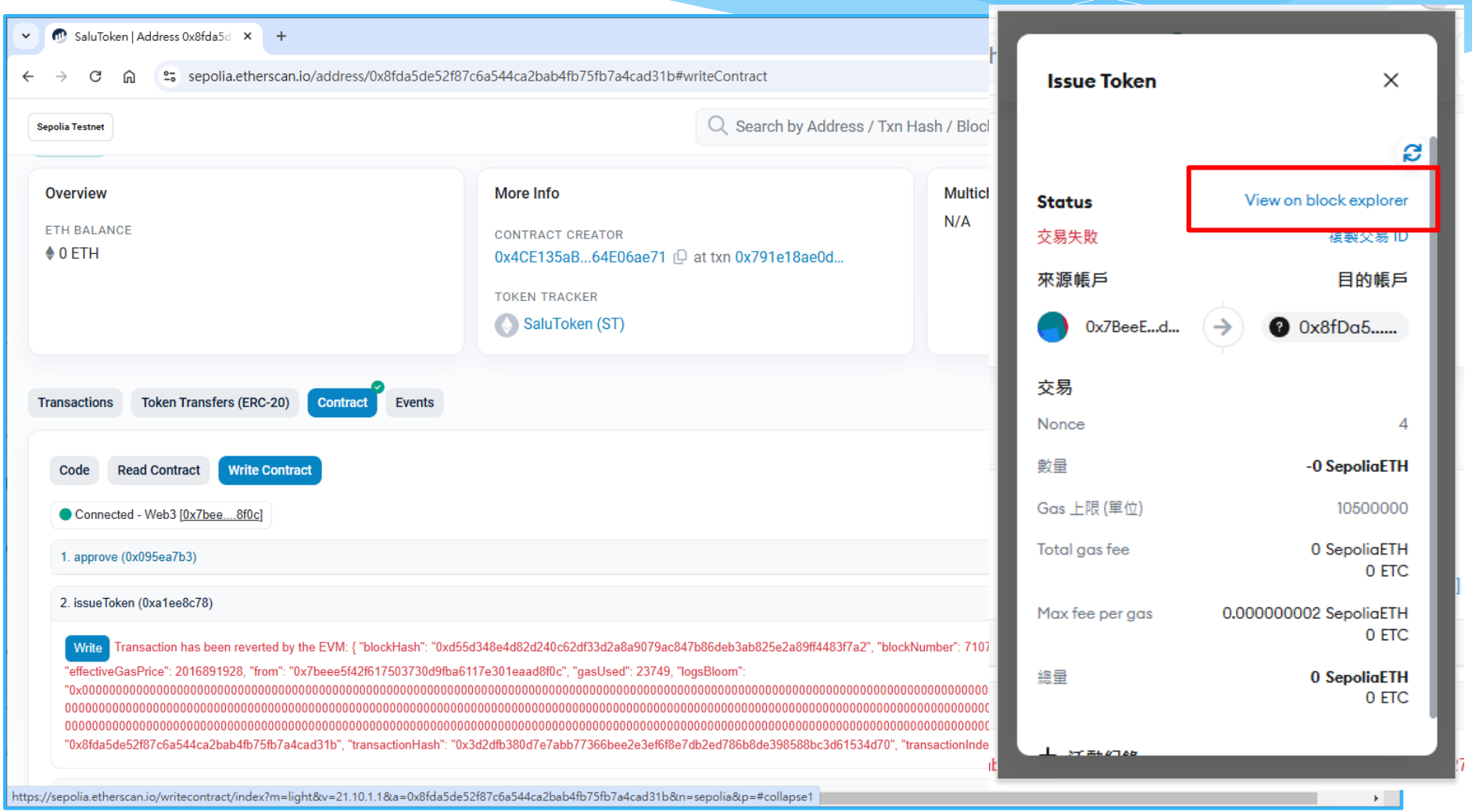
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#writeContract

Write Contract - issueToken



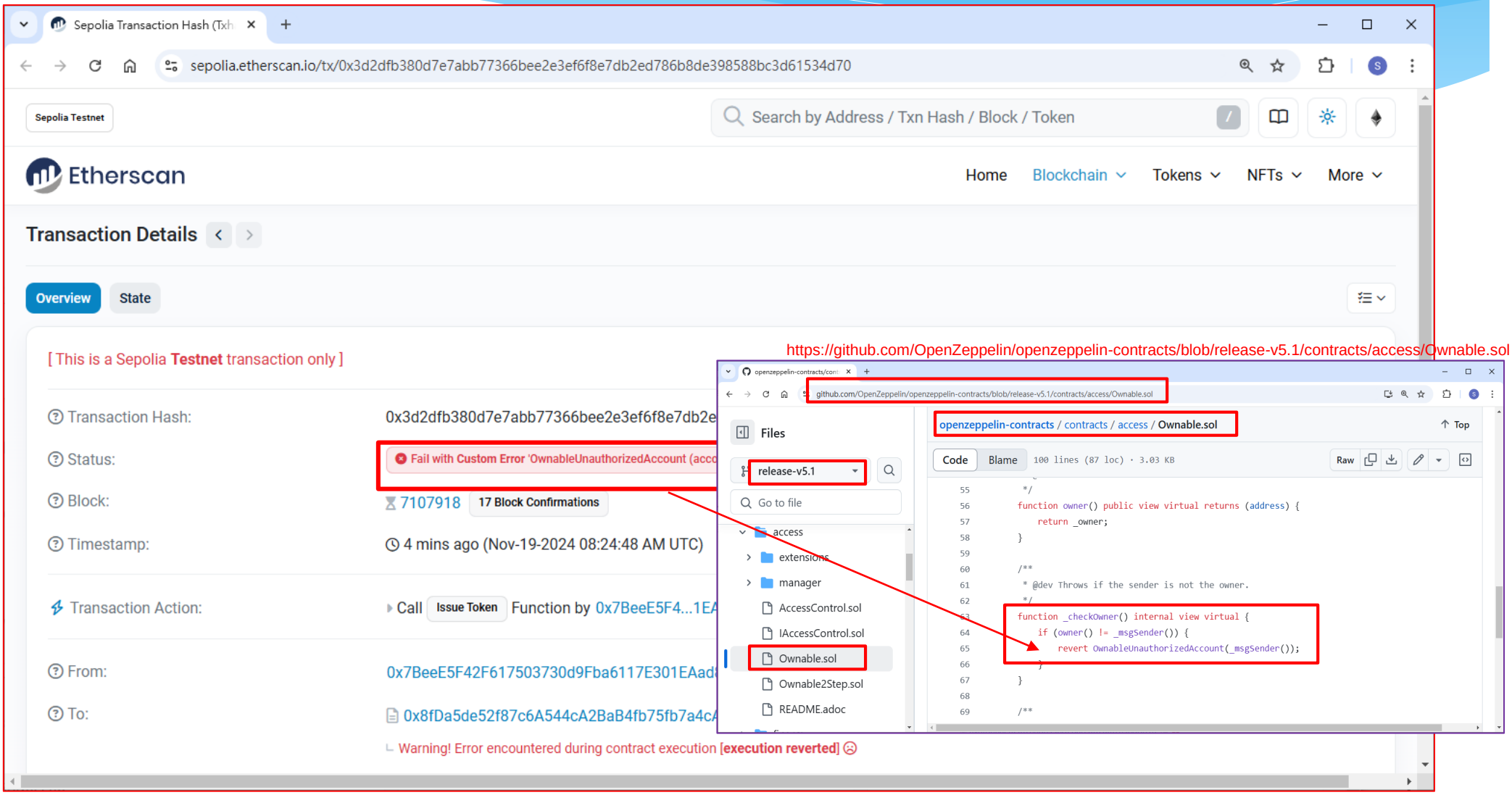
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#writeContract

View Transaction



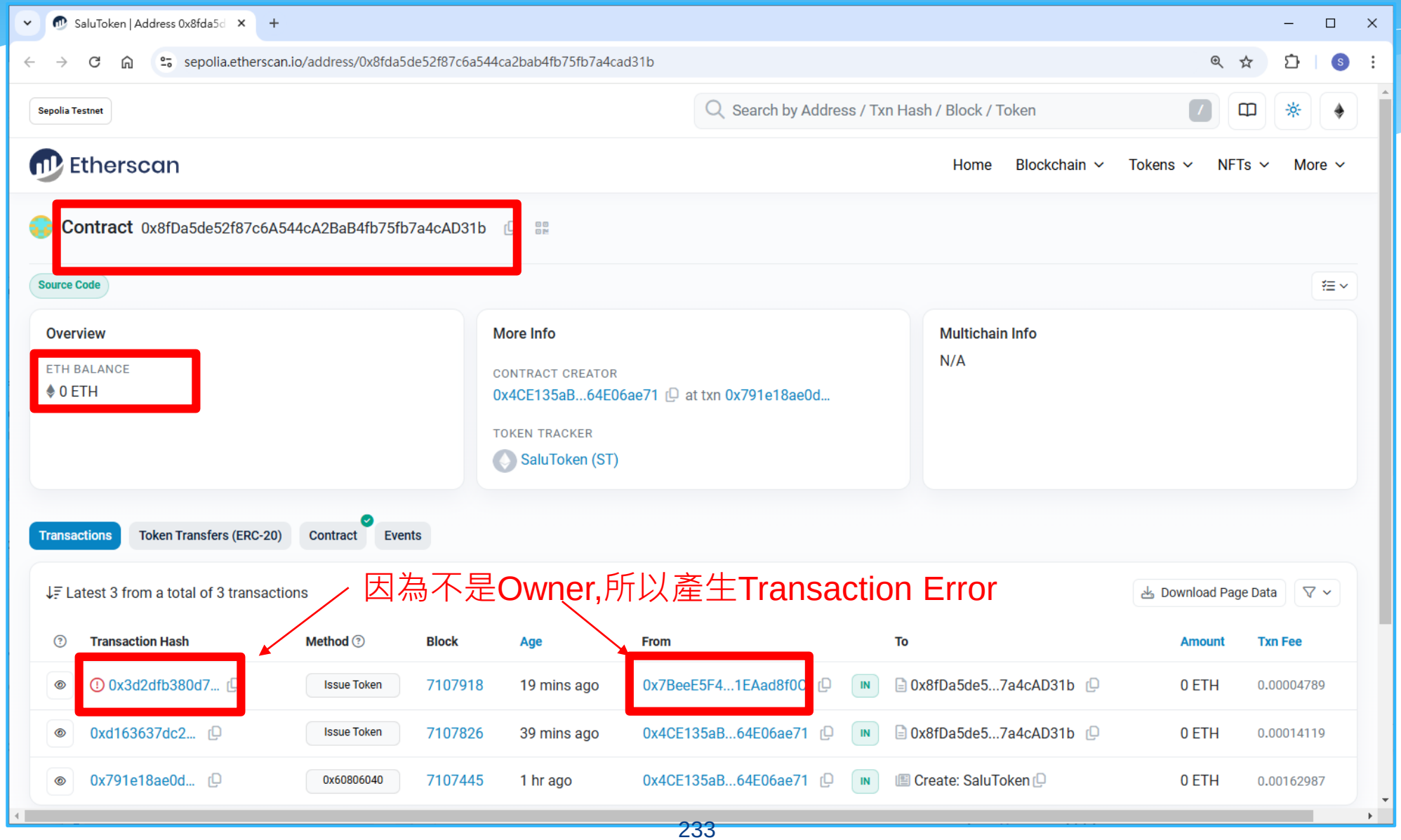
https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b#writeContract

View Transaction



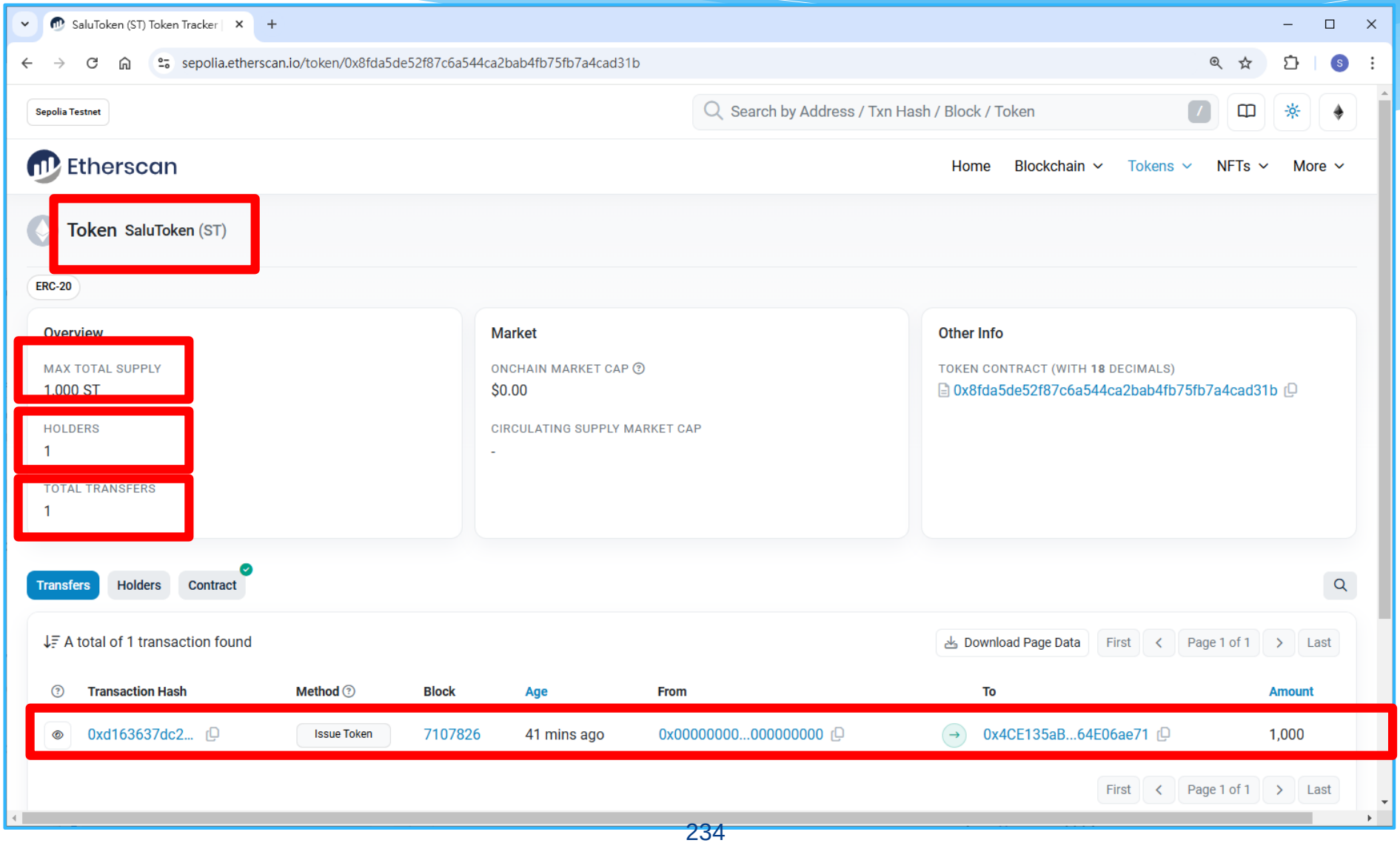
https://sepolia.etherscan.io/tx/0x3d2dfb380d7e7abb77366bee2e3ef6f8e7db2ed786b8de398588bc3d61534d70

Smart Contract Information



https://sepolia.etherscan.io/address/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b

Smart Contract Information



https://sepolia.etherscan.io/token/0x8fda5de52f87c6a544ca2bab4fb75fb7a4cad31b

Case 3 – ETH

Ethereum genesis block

- The ICO carried out by ethereum.org starting from 2014/07/22 to 2014/09/02.
- The launch of the Ethereum mainnet occurred on 2015/07/30.
- Ethereum Genesis block reference
  - https://github.com/ethereum/ethereumj/blob/develop/ethereumj-core/src/main/resources/genesis/frontier.json
  - https://github.com/odaceo/ethereum-genesis-block/blob/master/mainnet.json

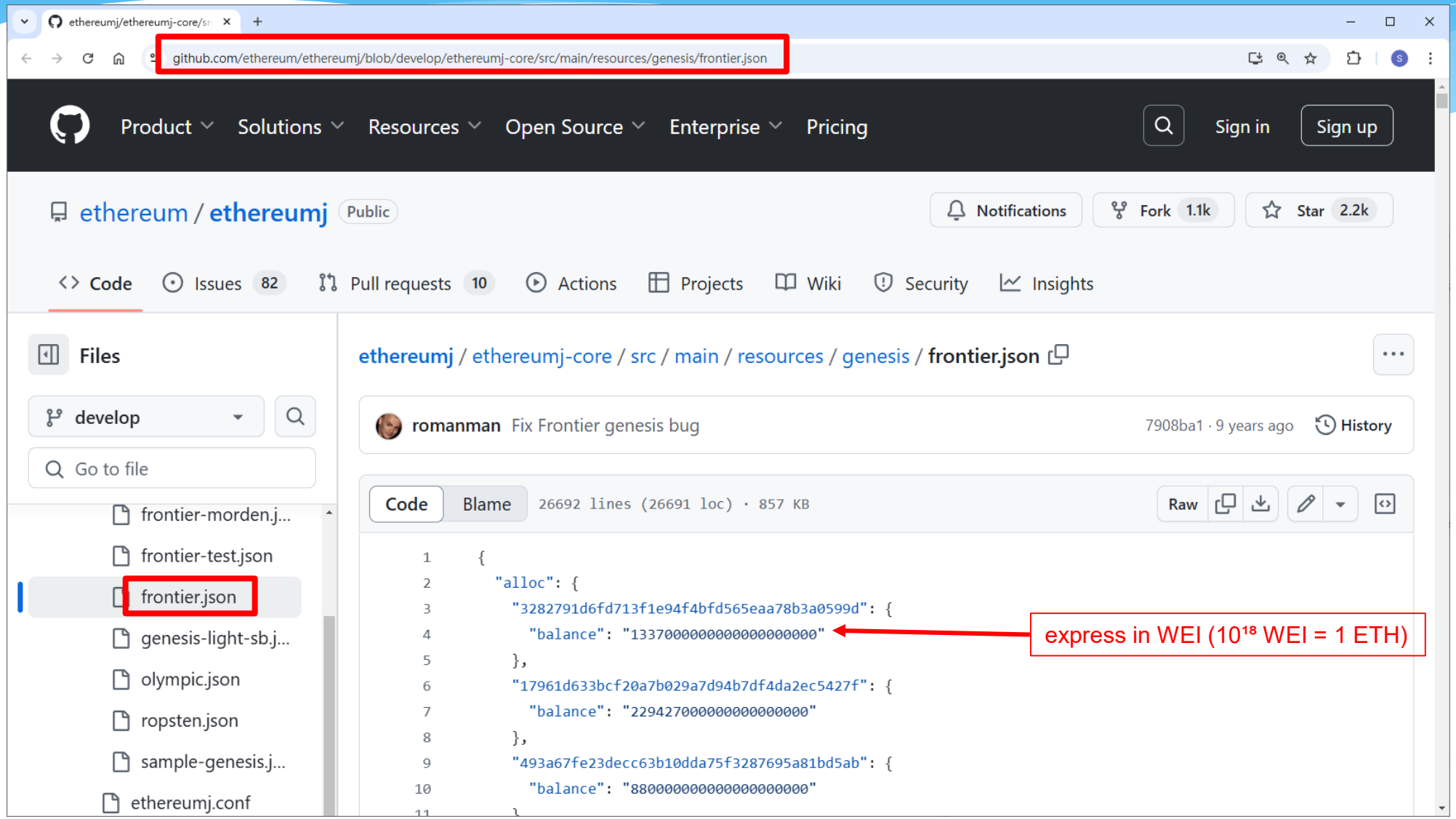
- Ethereum Contract Address
  - https://etherscan.io/address/0xd76b5c2a23ef78368d8e34288b5b65d616b746ae
  - 此smart contract沒有mint function，而是把token在一開始時放在Genesis block內給各個相關或有幫助的account address。
  - 原本是PoW機制，現已改成為PoS機制。

Creating The Genesis Block

- Every blockchain starts with a genesis block. When Geth is run with default settings for the first time, it commits the Mainnet genesis to the database. For a private network, it is generally preferable to use a different genesis block. The genesis block is configured using a genesis.json file whose path must be provided to Geth on start-up. When creating a genesis block, a few initial parameters for the private blockchain must be defined.
- https://geth.ethereum.org/docs/fundamentals/private-network

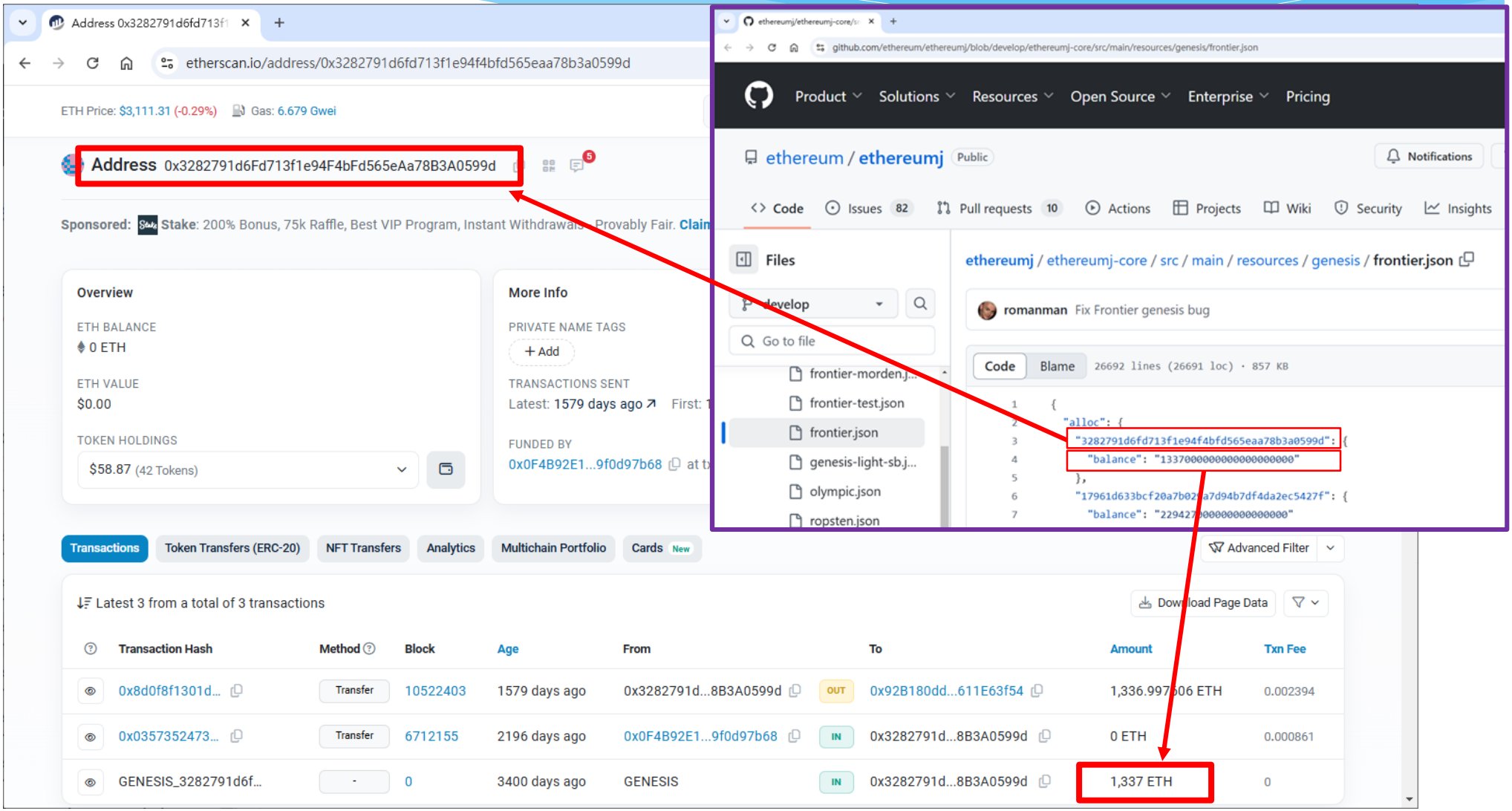
```
## to create a blockchain node
$ geth init --datadir data genesis.json

## When Geth is started using --datadir data the genesis block defined in genesis.json will be used
$ geth --datadir data --networkid 12345
```

Ethereum Genesis block



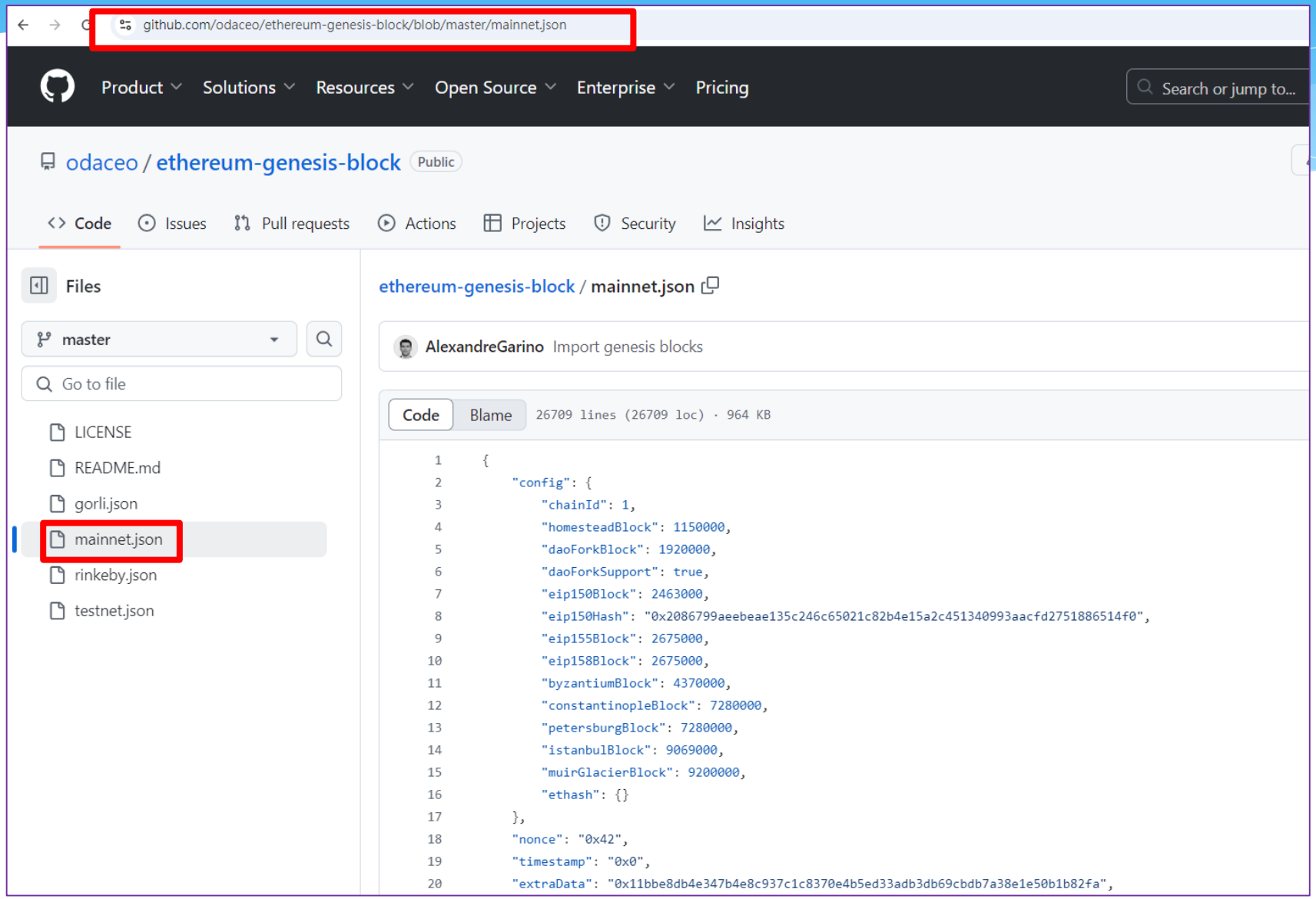
https://github.com/ethereum/ethereumj/blob/develop/ethereumj-core/src/main/resources/genesis/frontier.json

Ethereum Genesis block



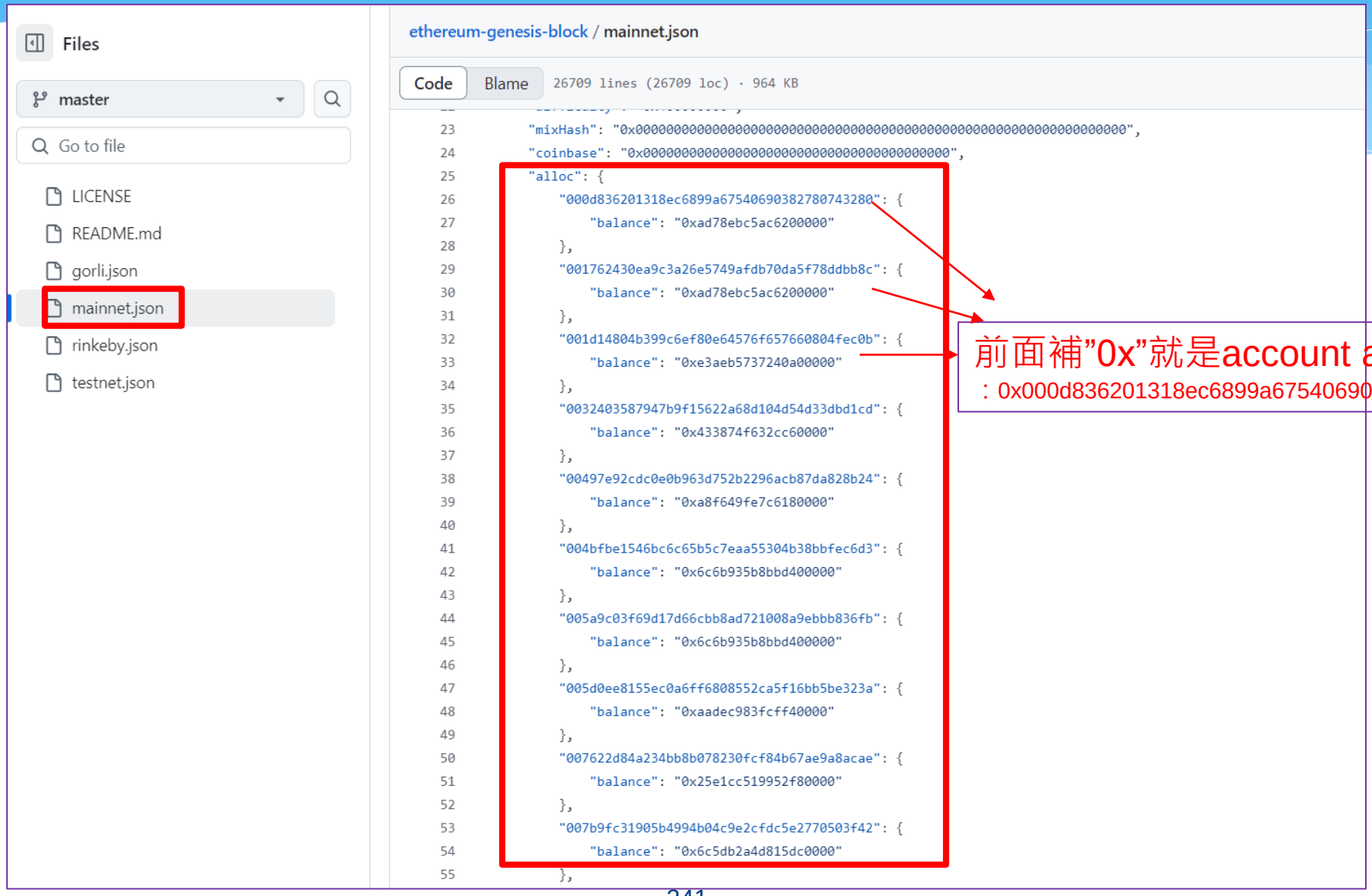
https://github.com/ethereum/ethereumj/blob/develop/ethereumj-core/src/main/resources/genesis/frontier.json

Ethereum Genesis block



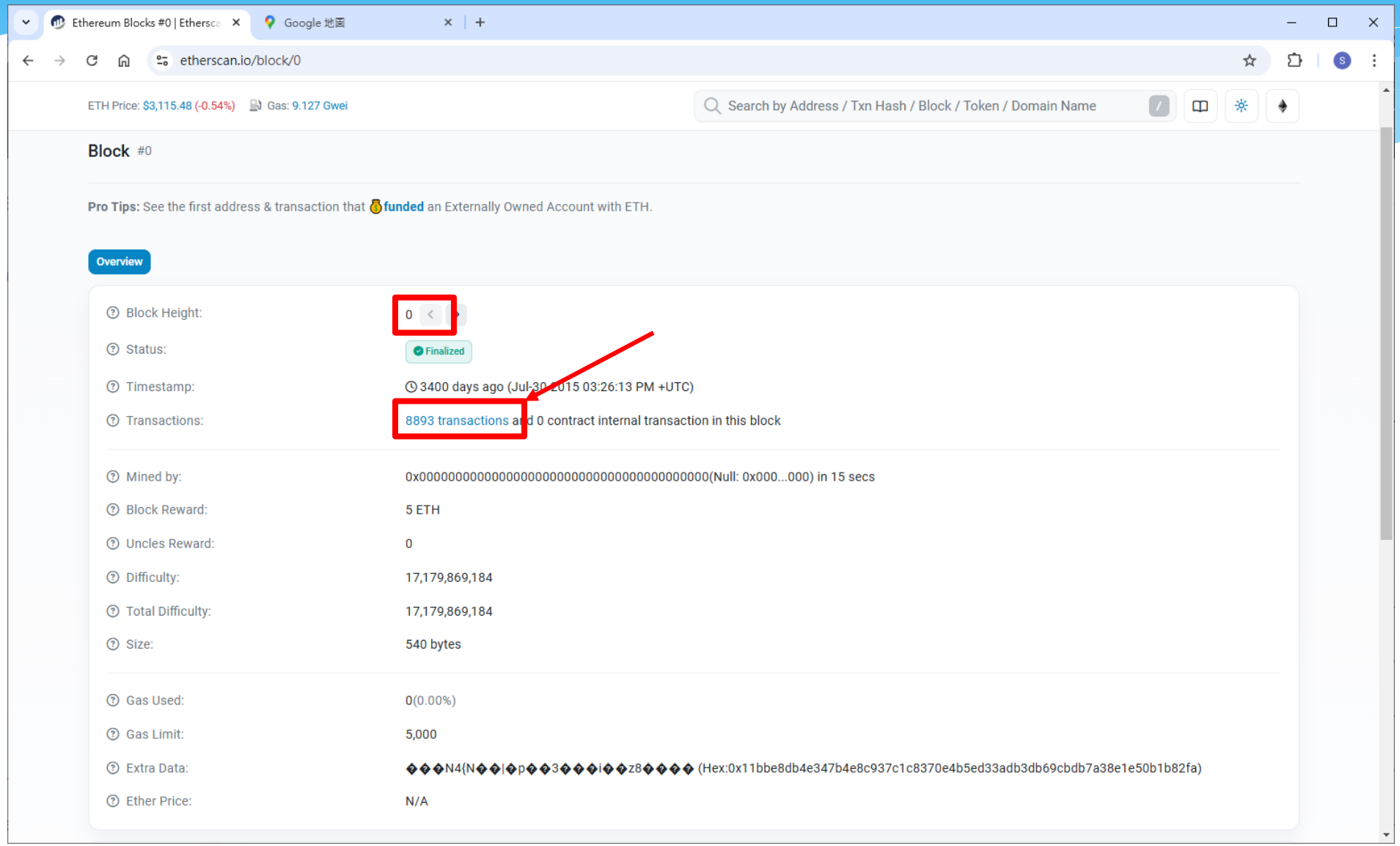
https://github.com/odaceo/ethereum-genesis-block/blob/master/mainnet.json

Ethereum Genesis block



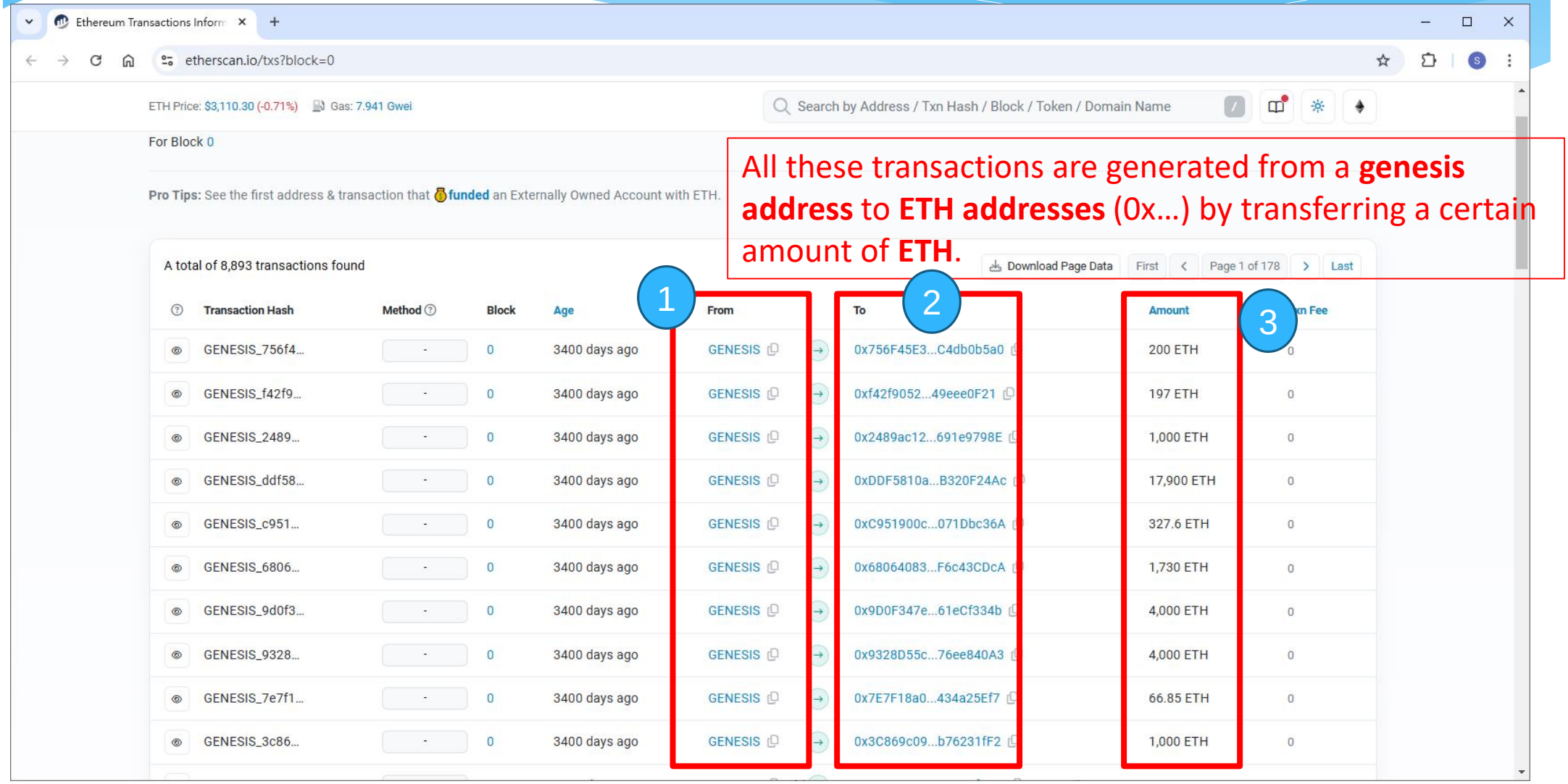
https://github.com/odaceo/ethereum-genesis-block/blob/master/mainnet.json

The Block 0 of Ethereum



https://etherscan.io/block/0

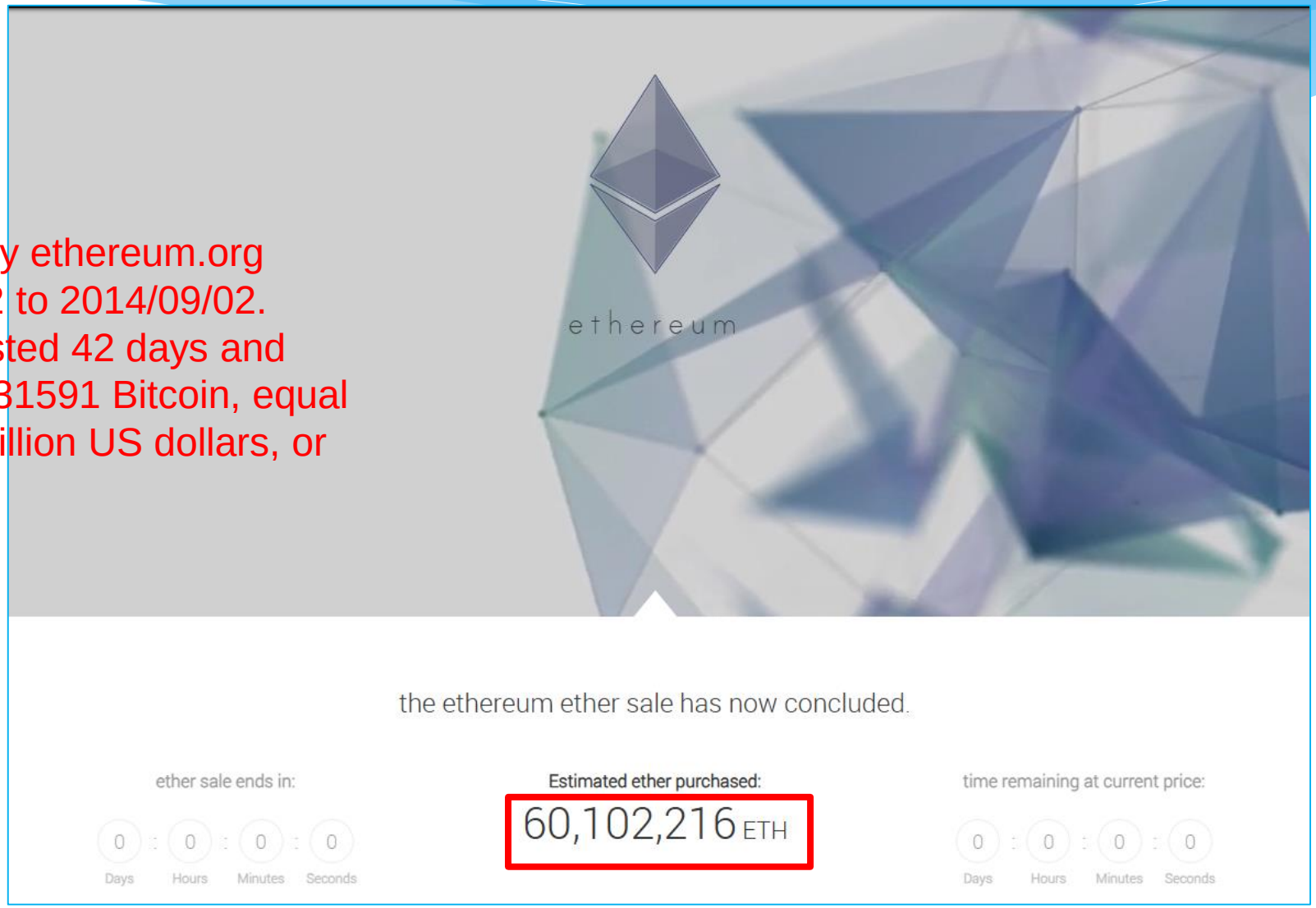
The transactions present in the genesis block



https://etherscan.io/txs?block=0

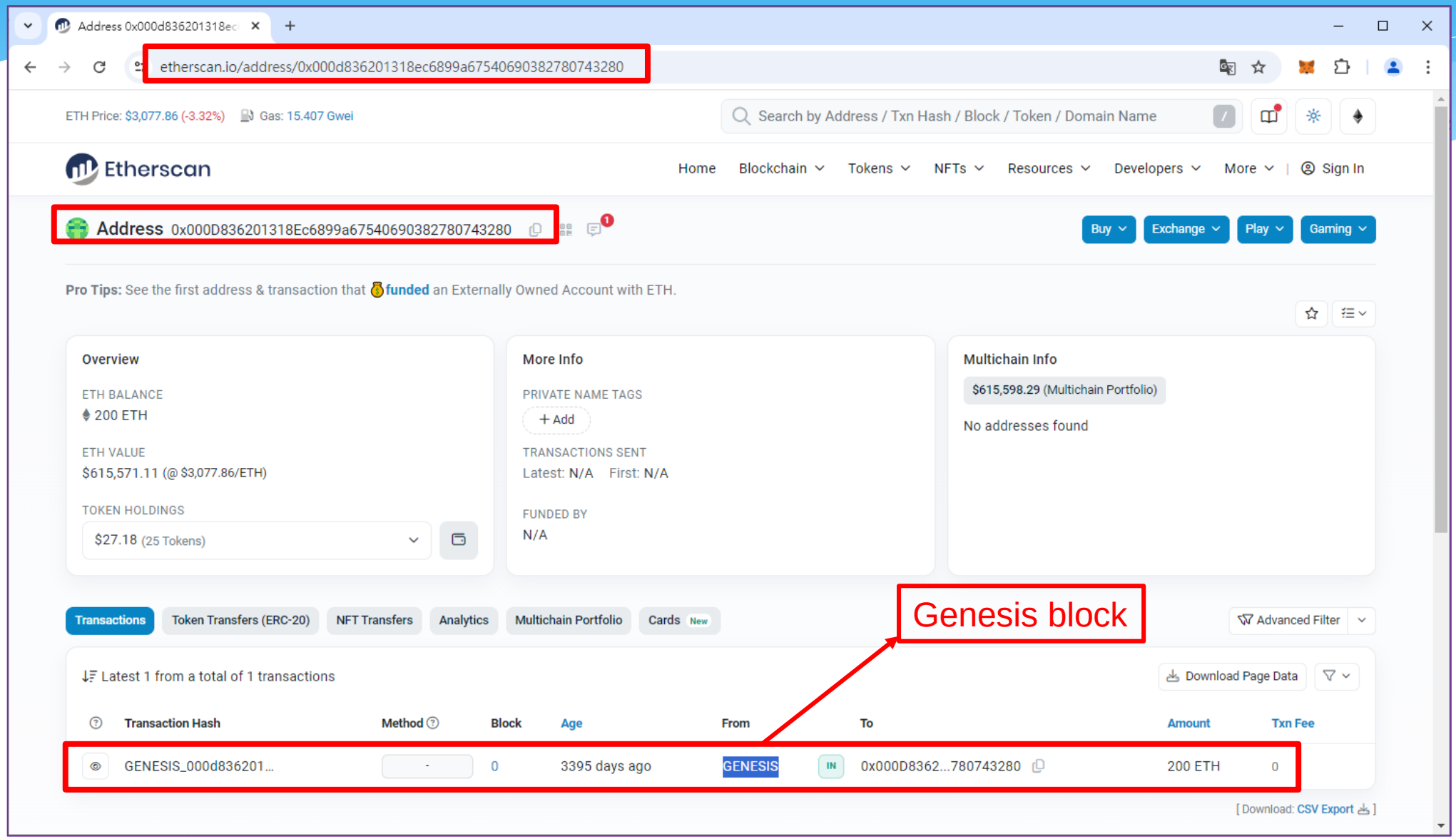
Ethereum ICO page

1. The ICO carried out by ethereum.org starting from 2014/07/22 to 2014/09/02.
2. The public offering lasted 42 days and totaled the collection of 31591 Bitcoin, equal to approximately 18.4 million US dollars, or 60,102,216 ETH.



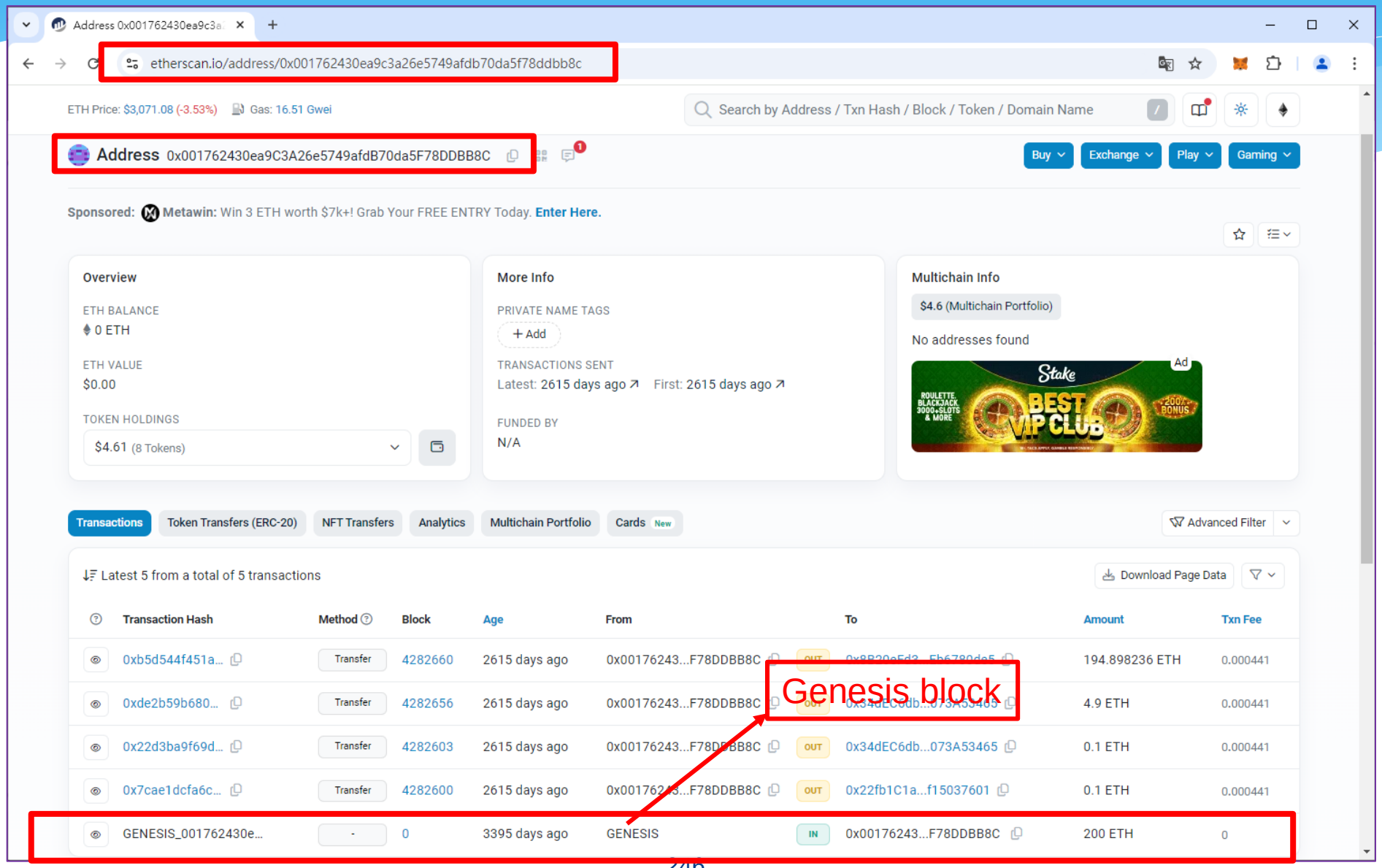
https://web.archive.org/web/20140904093914/https://www.ethereum.org/

Account Address-1



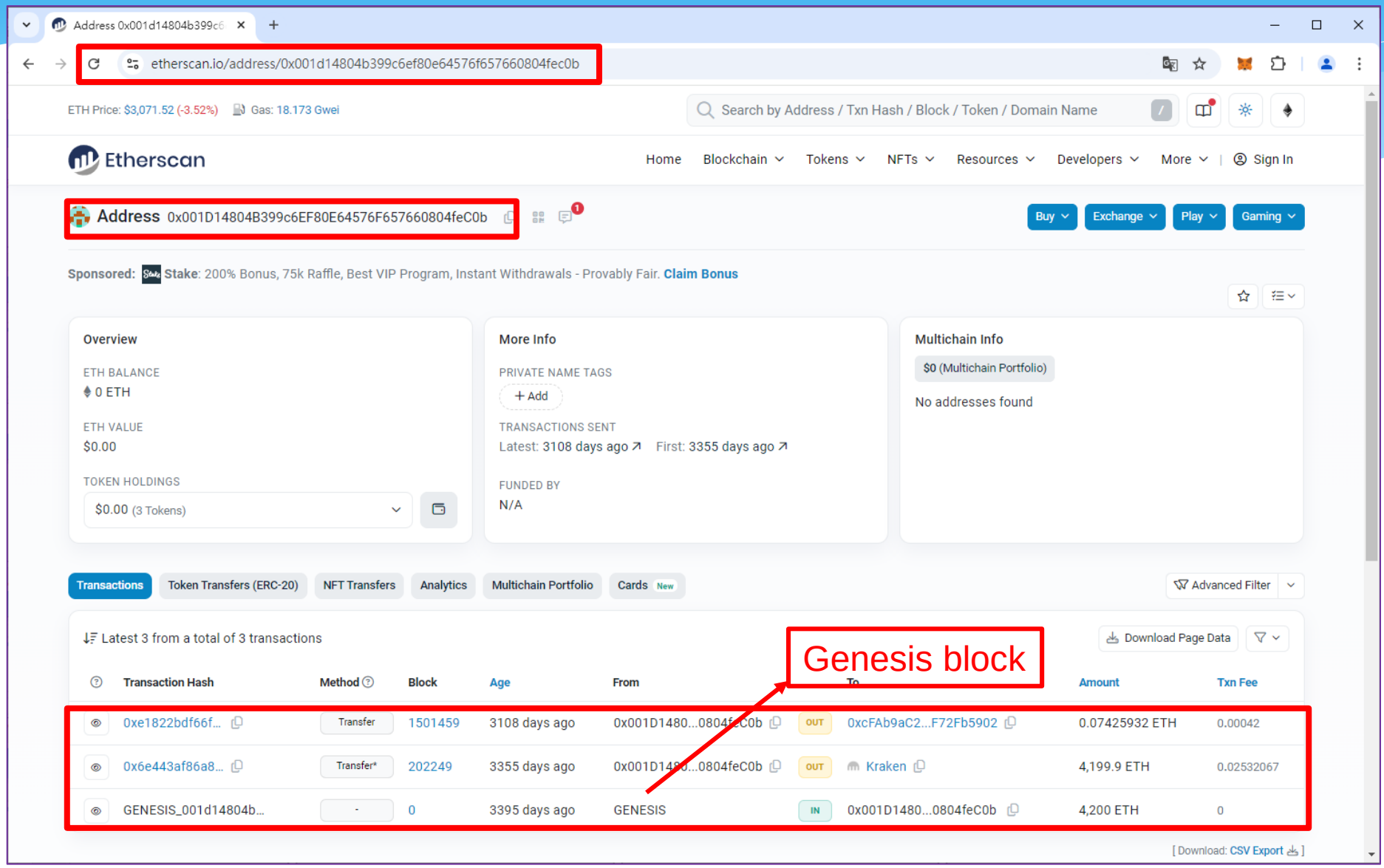
https://etherscan.io/address/0x000d836201318ec6899a67540690382780743280

Account Address-2



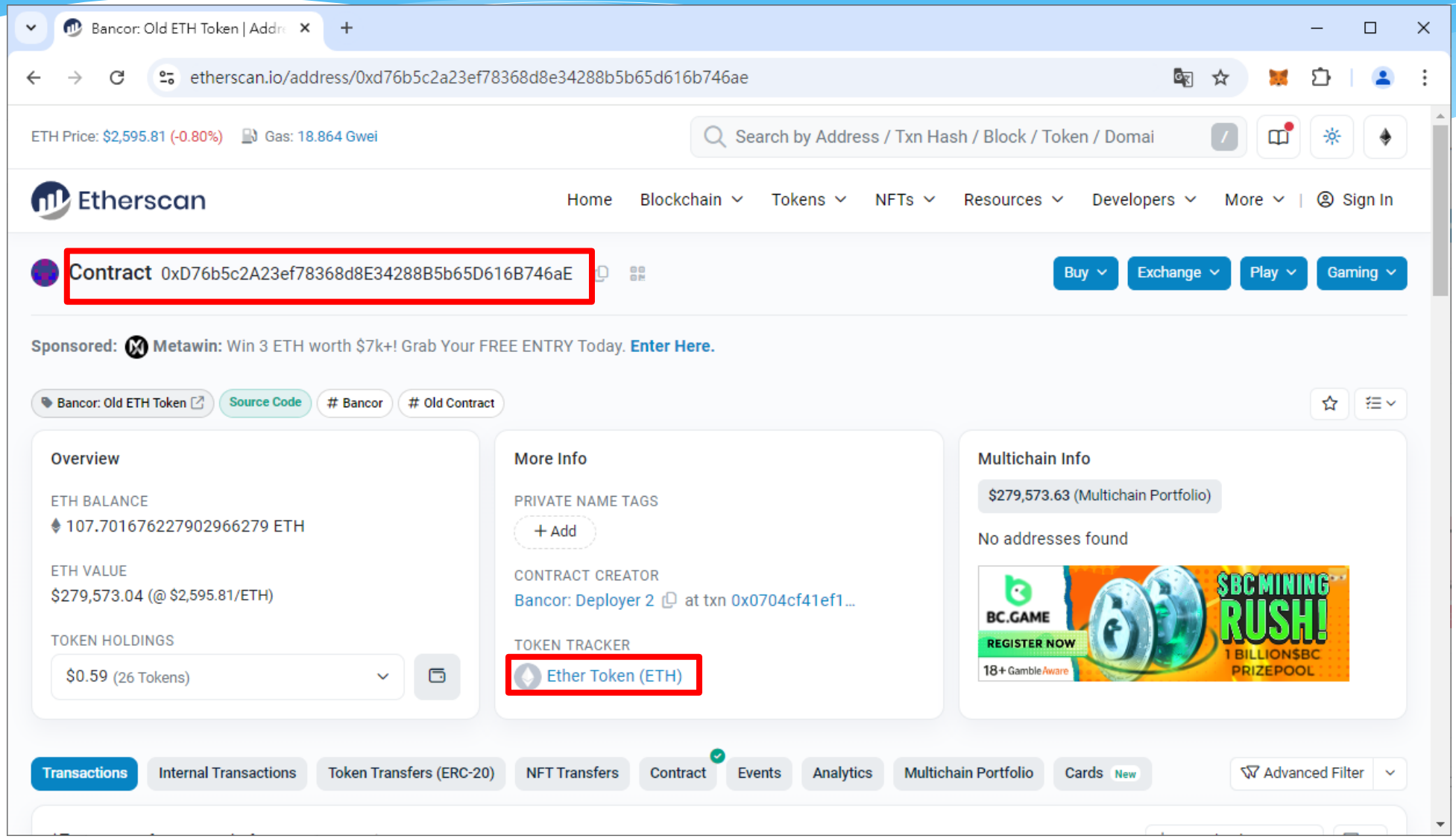
https://etherscan.io/address/0x001762430ea9c3a26e5749afdb70da5f78ddbb8c

Account Address-3



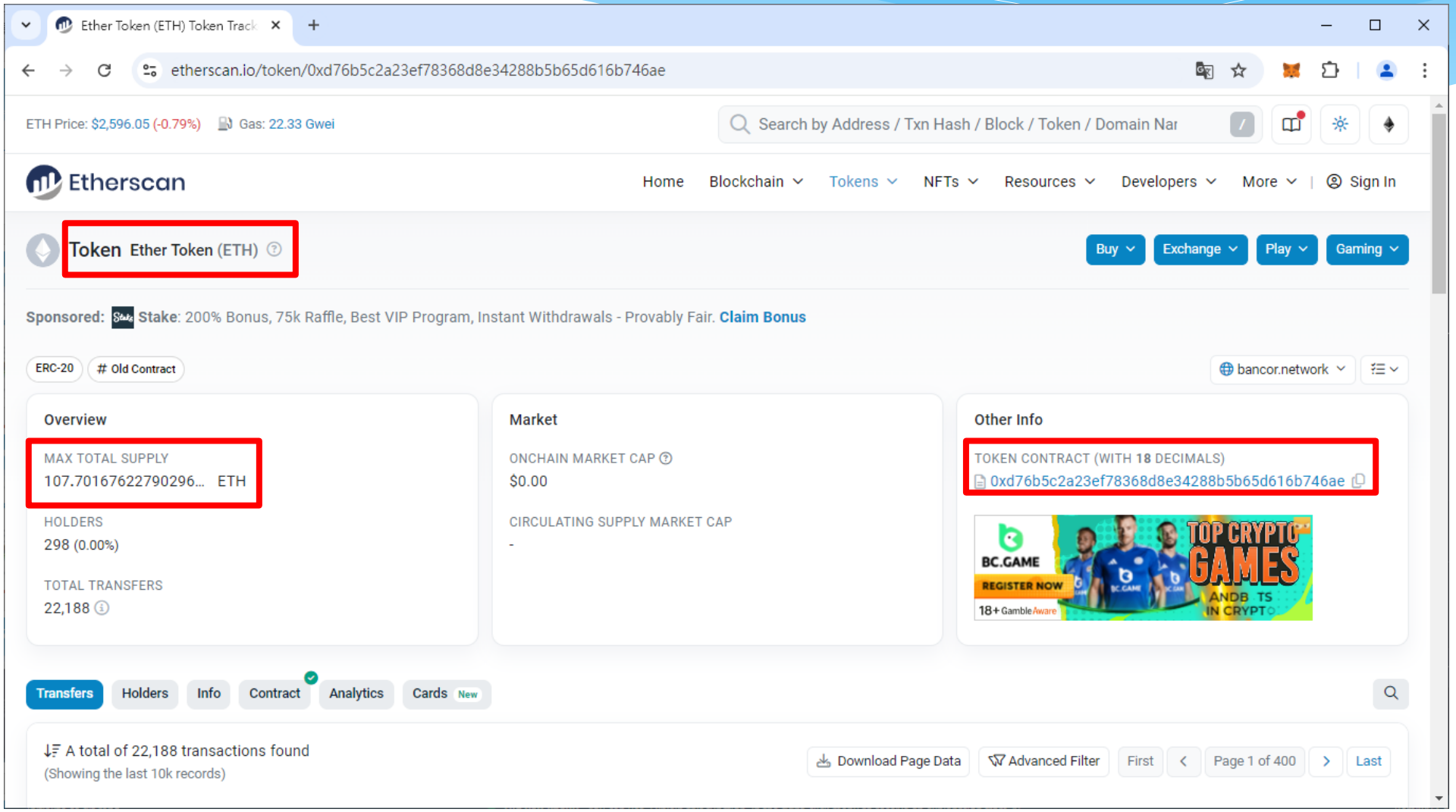
https://etherscan.io/address/0x001d14804b399c6ef80e64576f657660804fec0b

Search the Ether Token



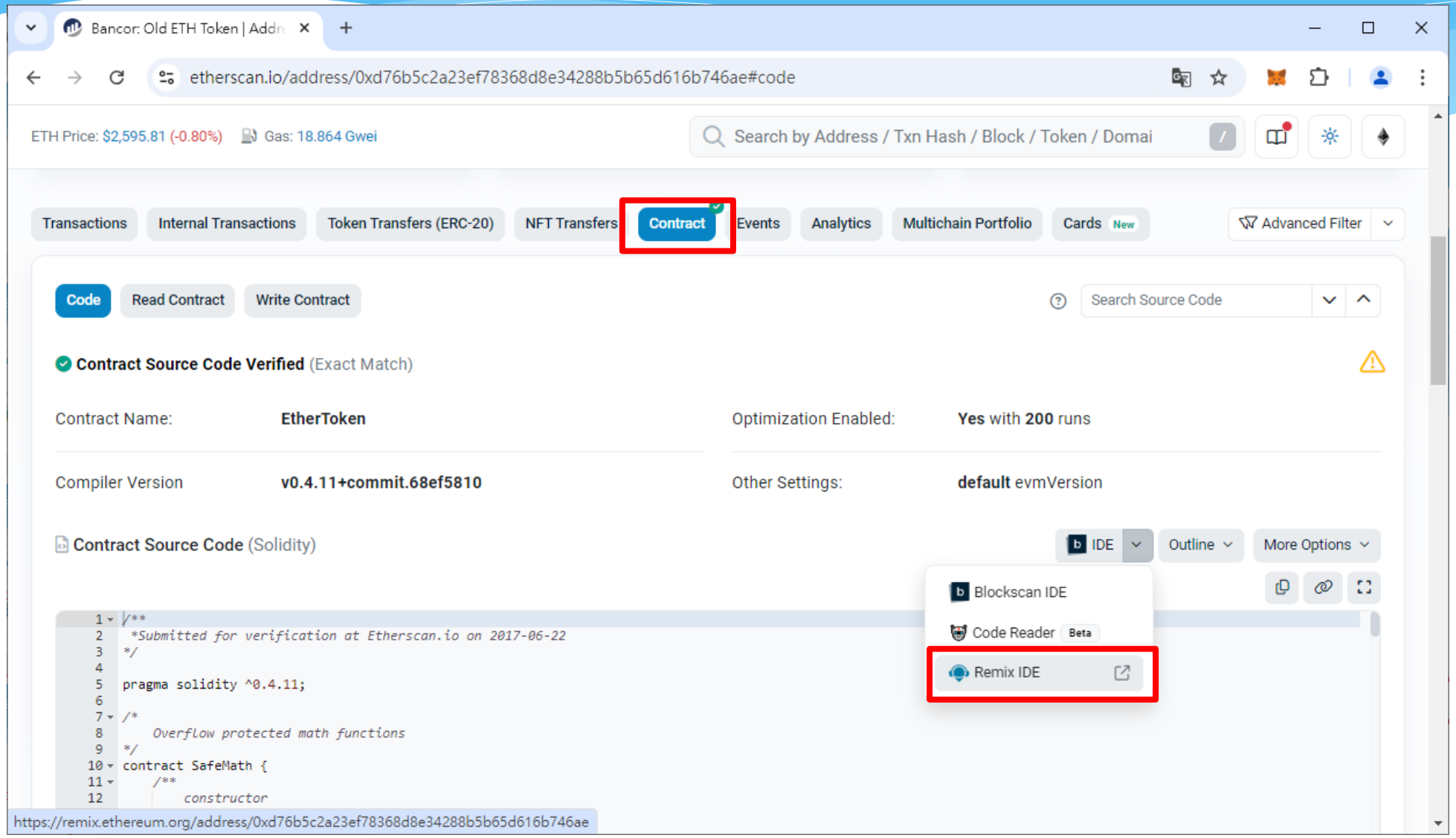
https://etherscan.io/address/0xd76b5c2a23ef78368d8e34288b5b65d616b746ae

Search the Ether Token



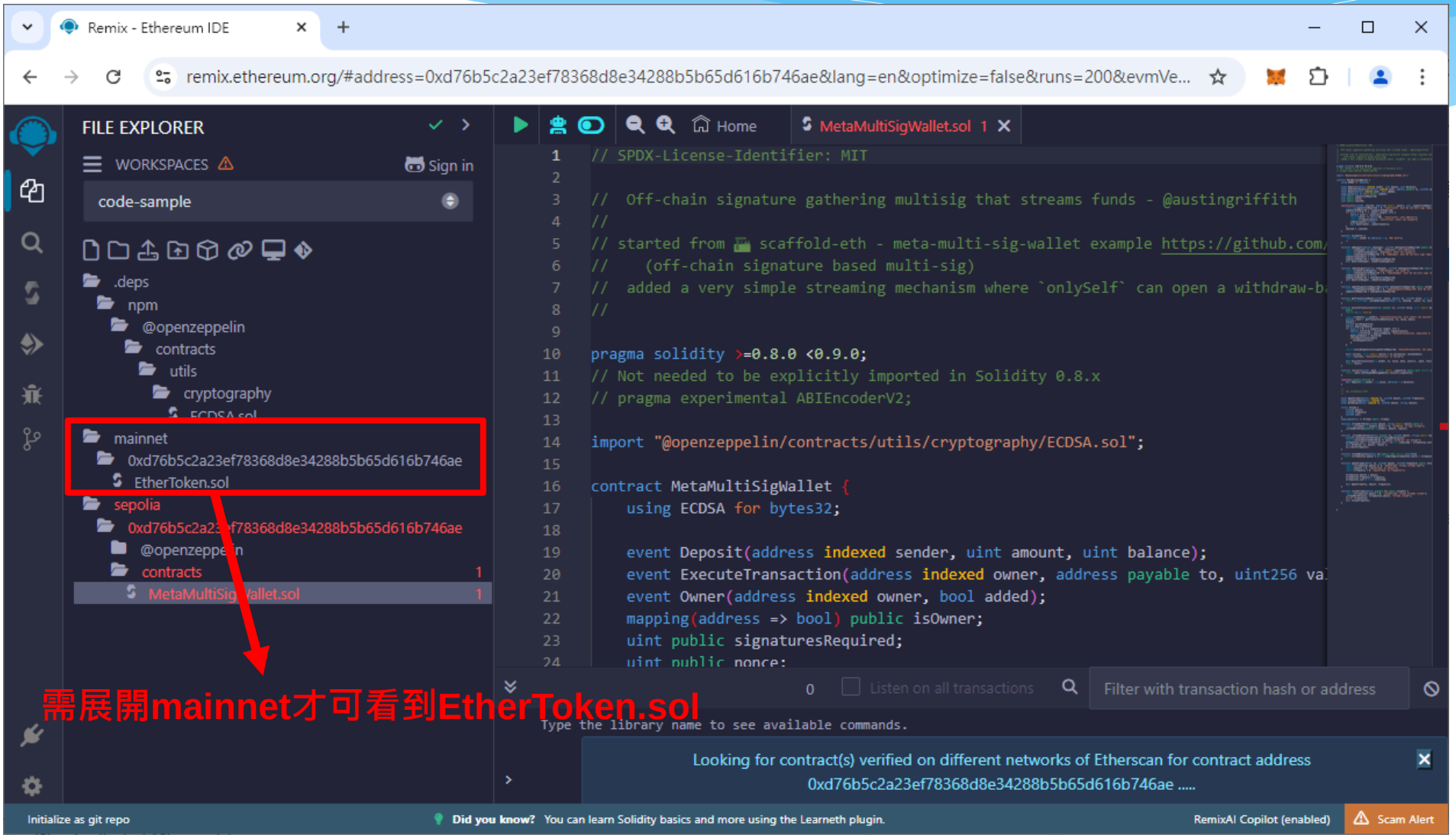
https://etherscan.io/token/0xd76b5c2a23ef78368d8e34288b5b65d616b746ae

Search the Ether Token

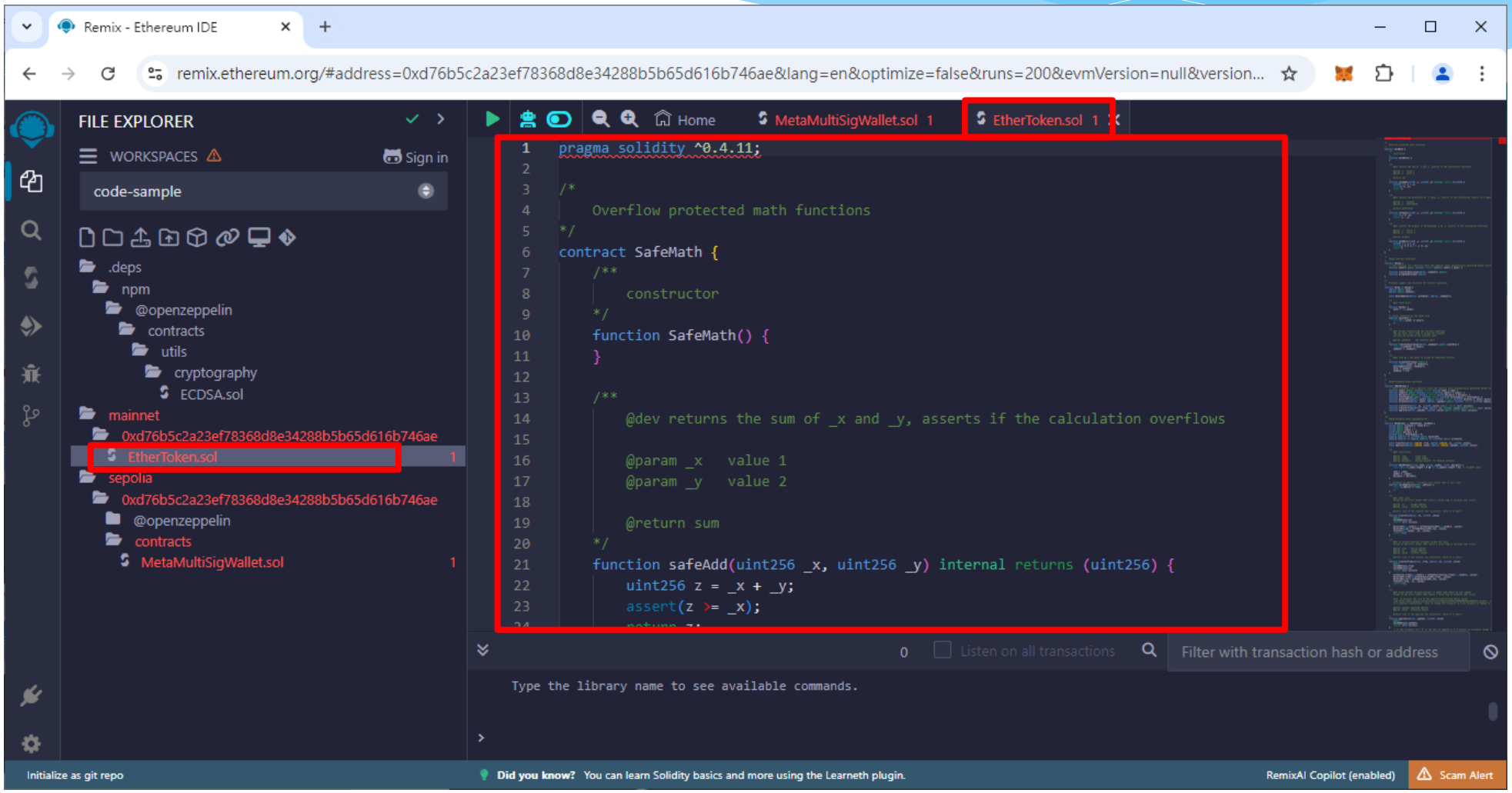


https://etherscan.io/address/0xd76b5c2a23ef78368d8e34288b5b65d616b746ae#code

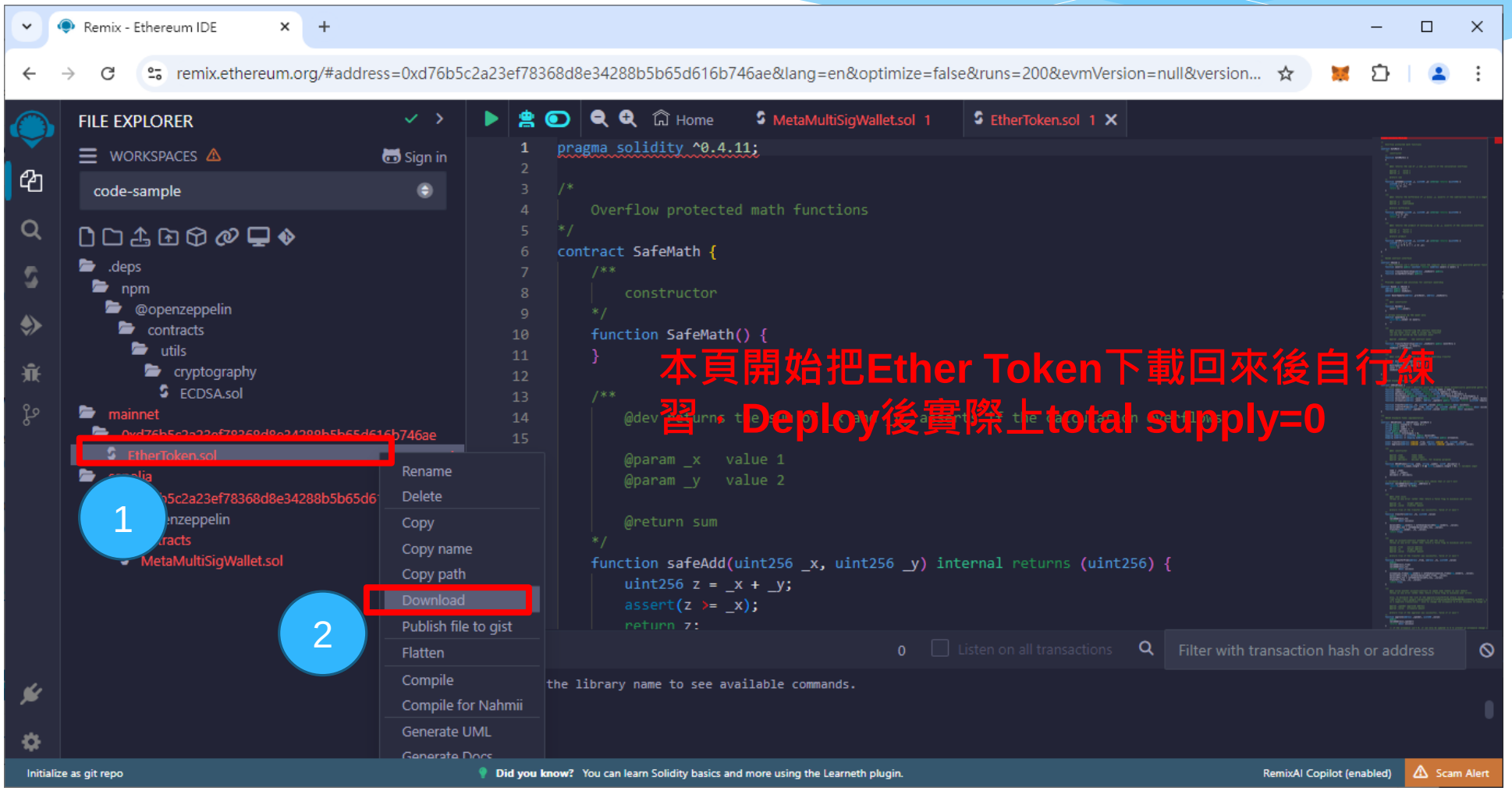
Search the Ether Token



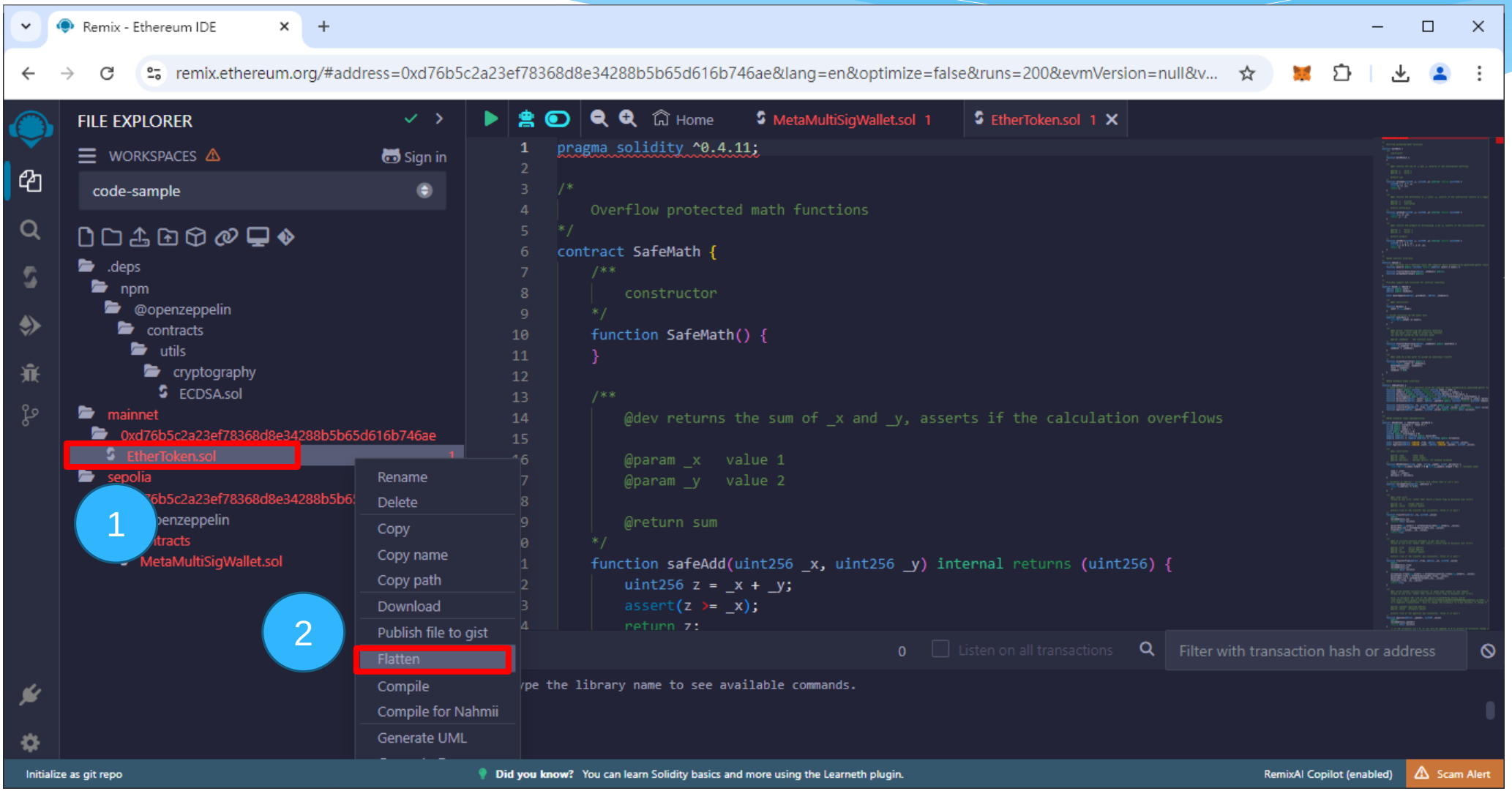
EtherToken.sol



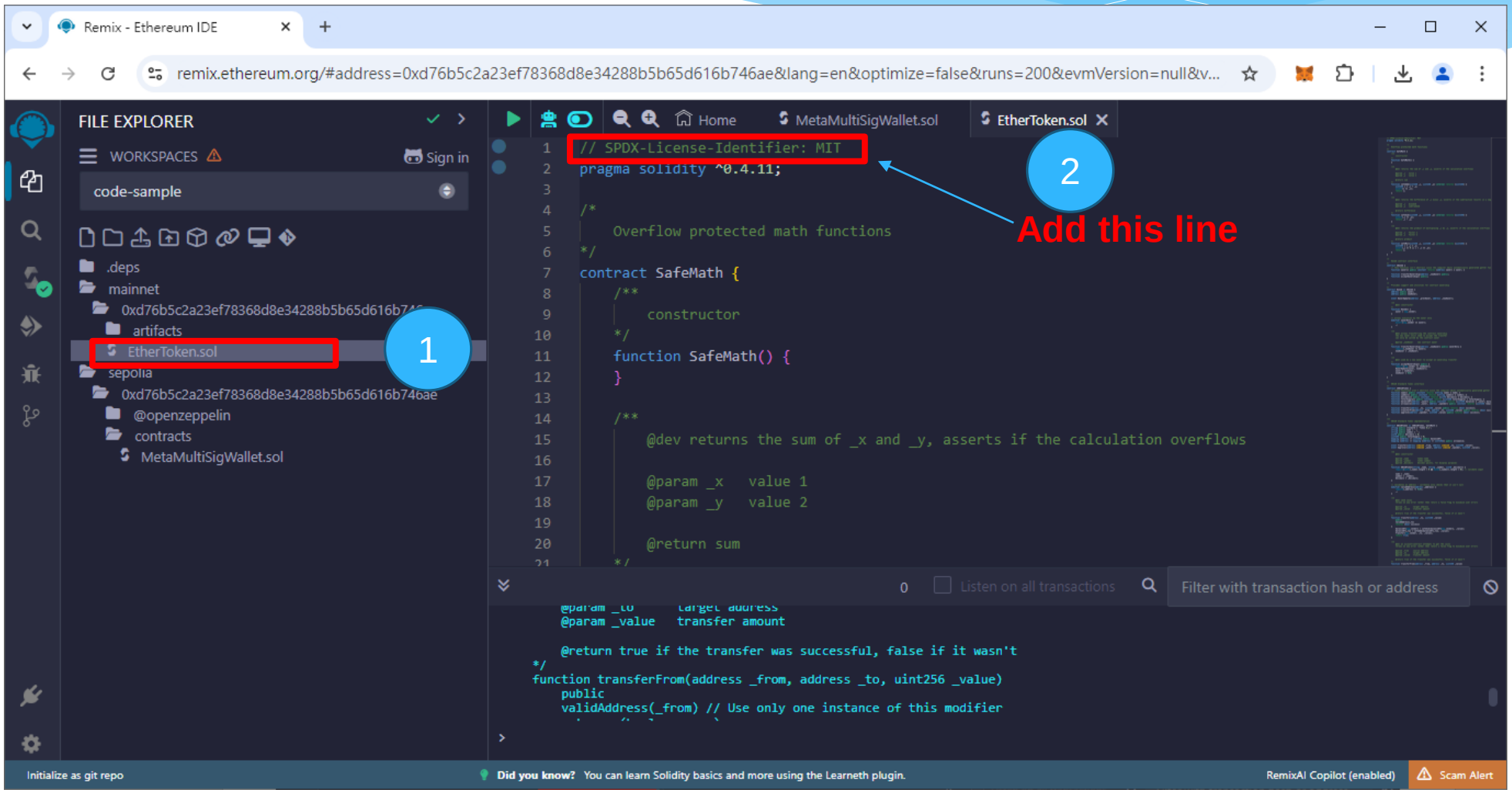
Download the EtherToken.sol



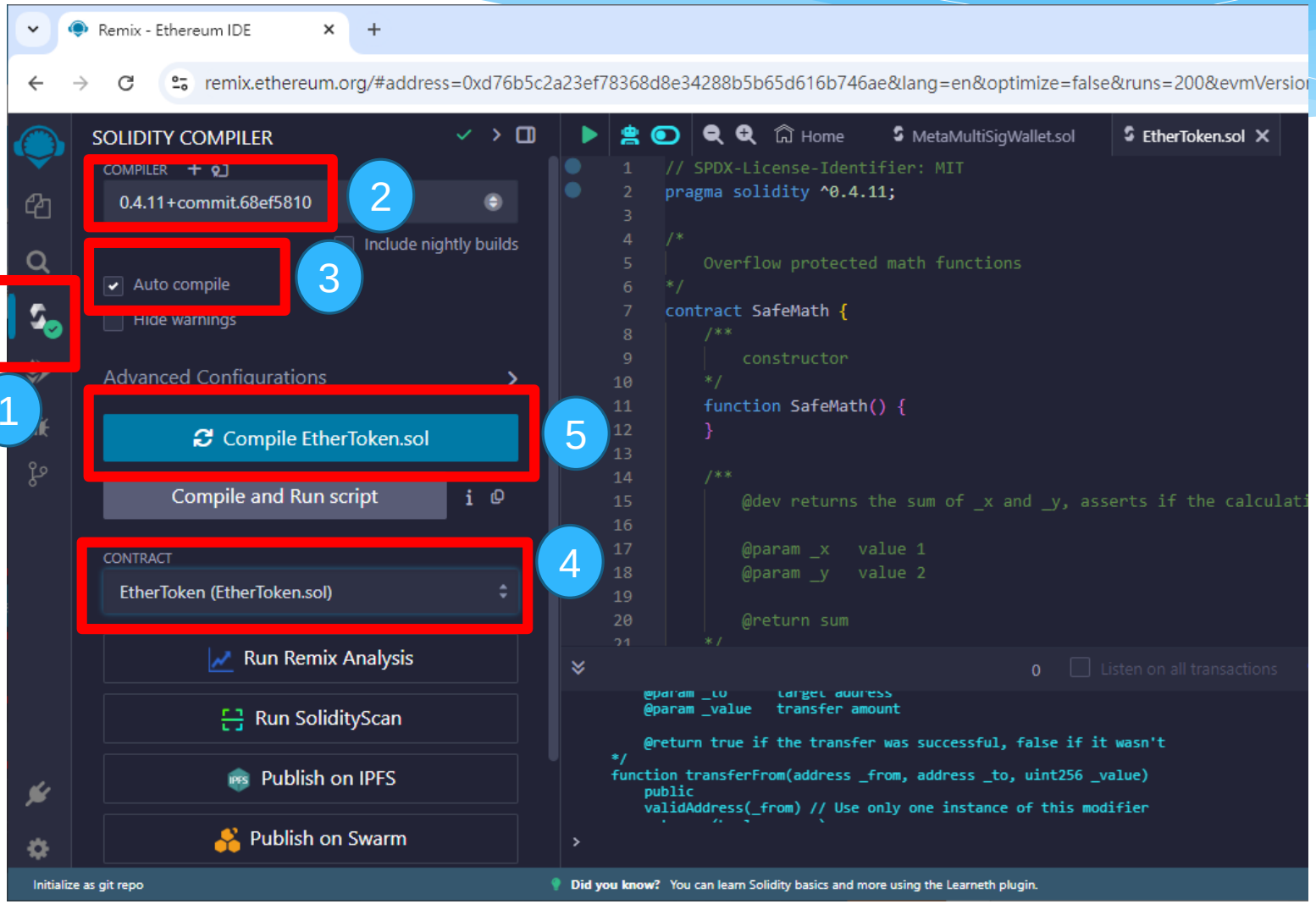
Flatten the EtherToken.sol



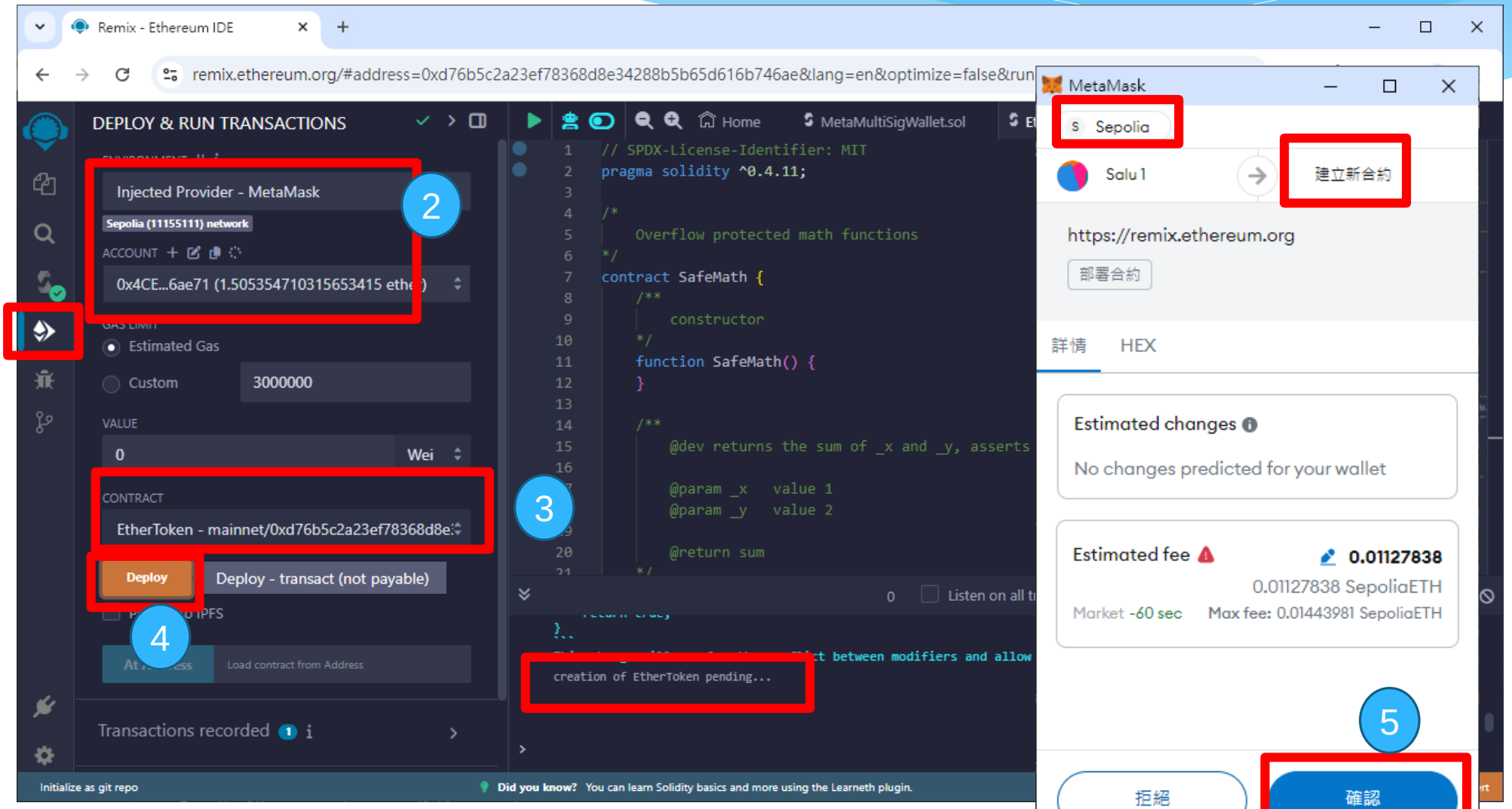
Add SPDX to EtherToken.sol



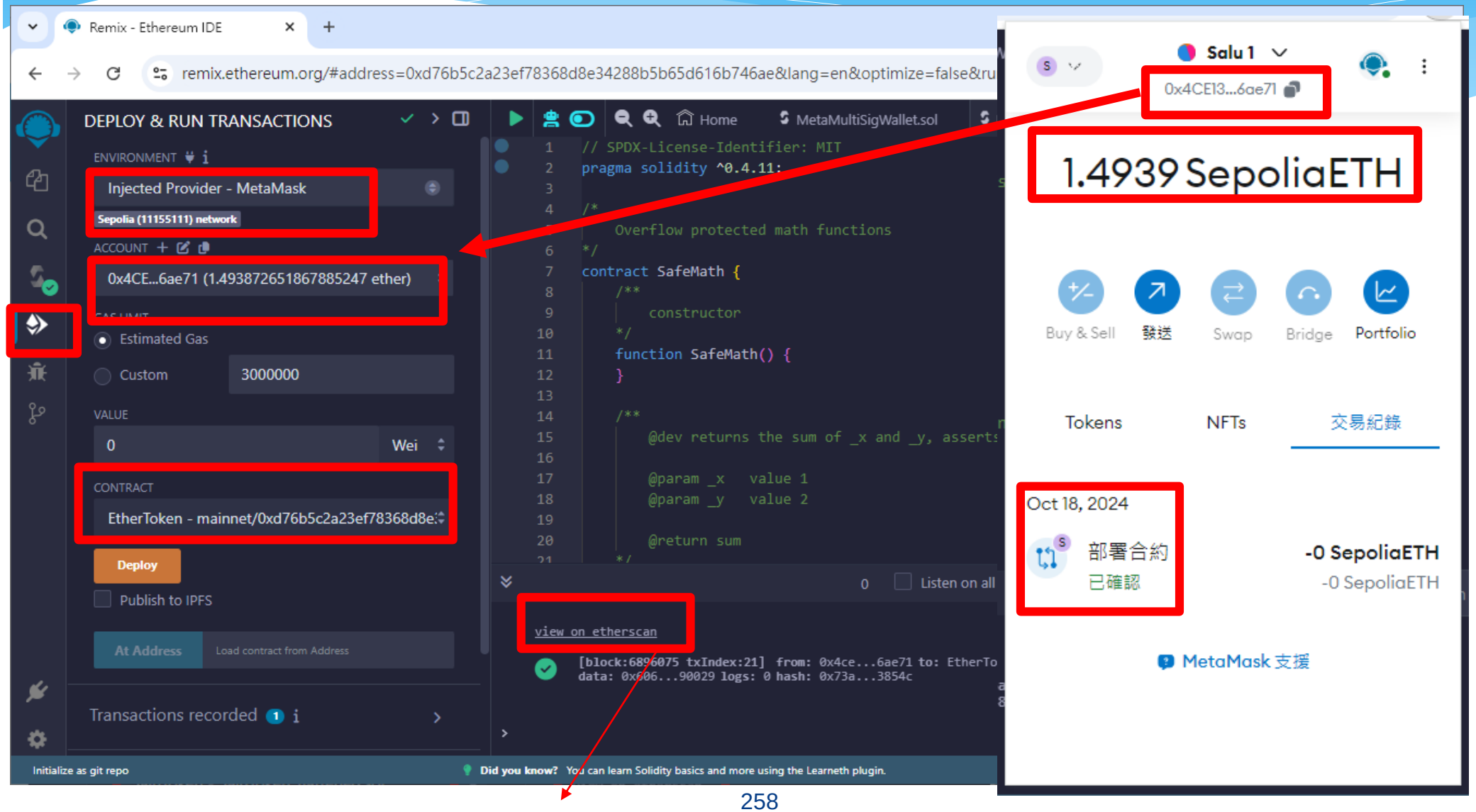
Compile Smart Contract



Deploy an ERC-20 Token

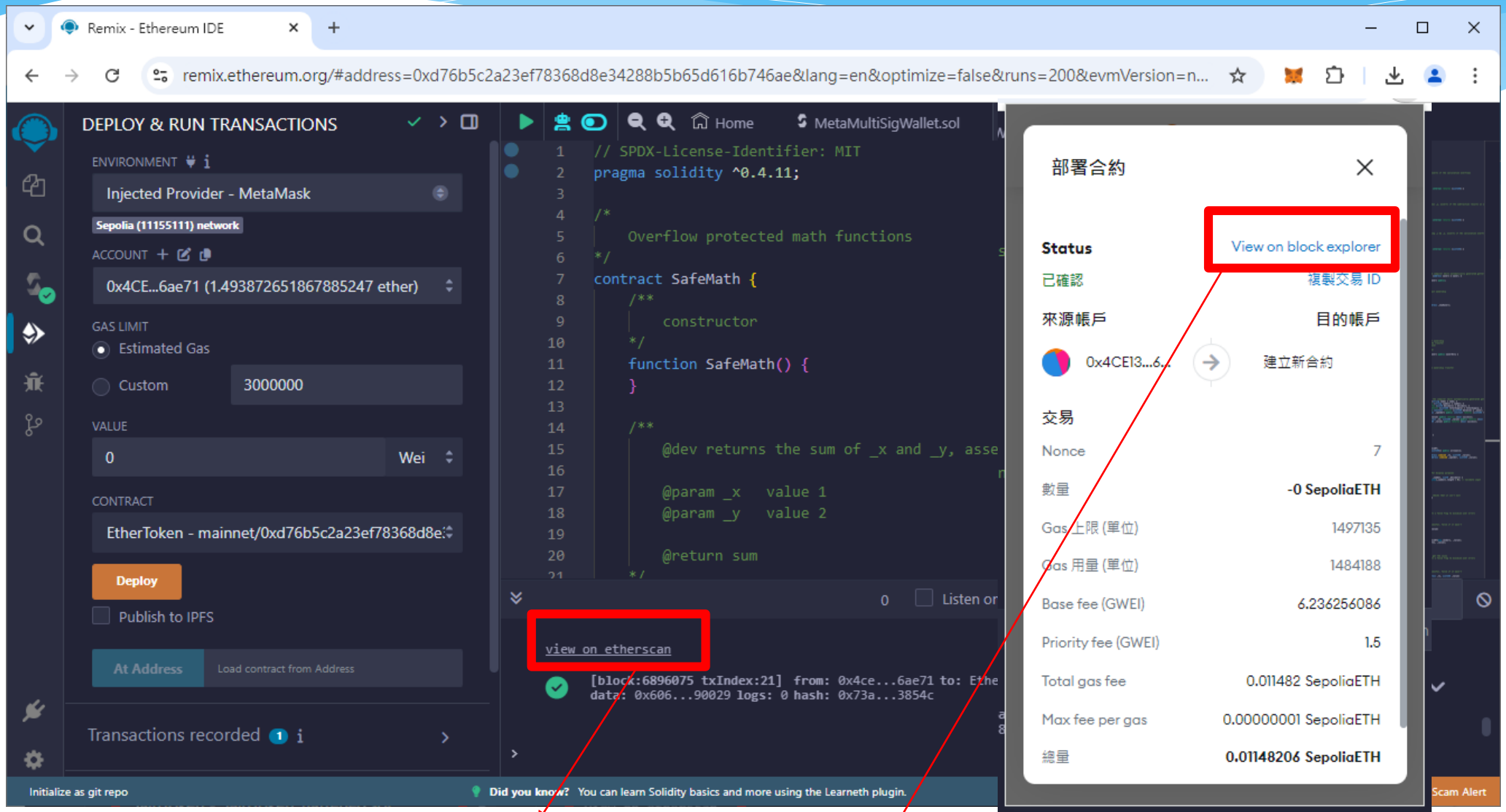


Deploy and Run Transaction



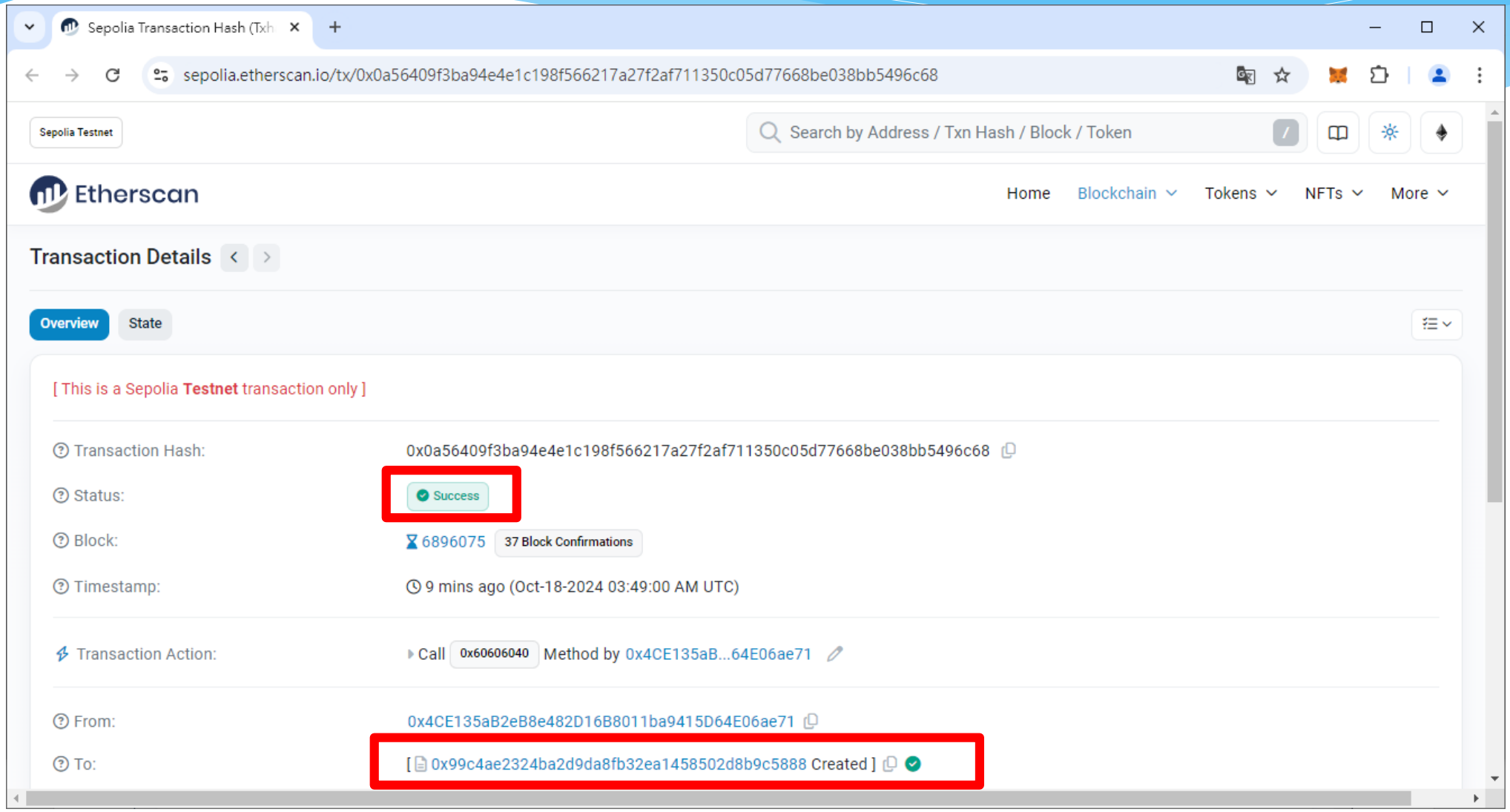
https://sepolia.etherscan.io/tx/0x0a56409f3ba94e4e1c198f566217a27f2af711350c05d77668be038bb5496c68

Deploy and Run Transaction



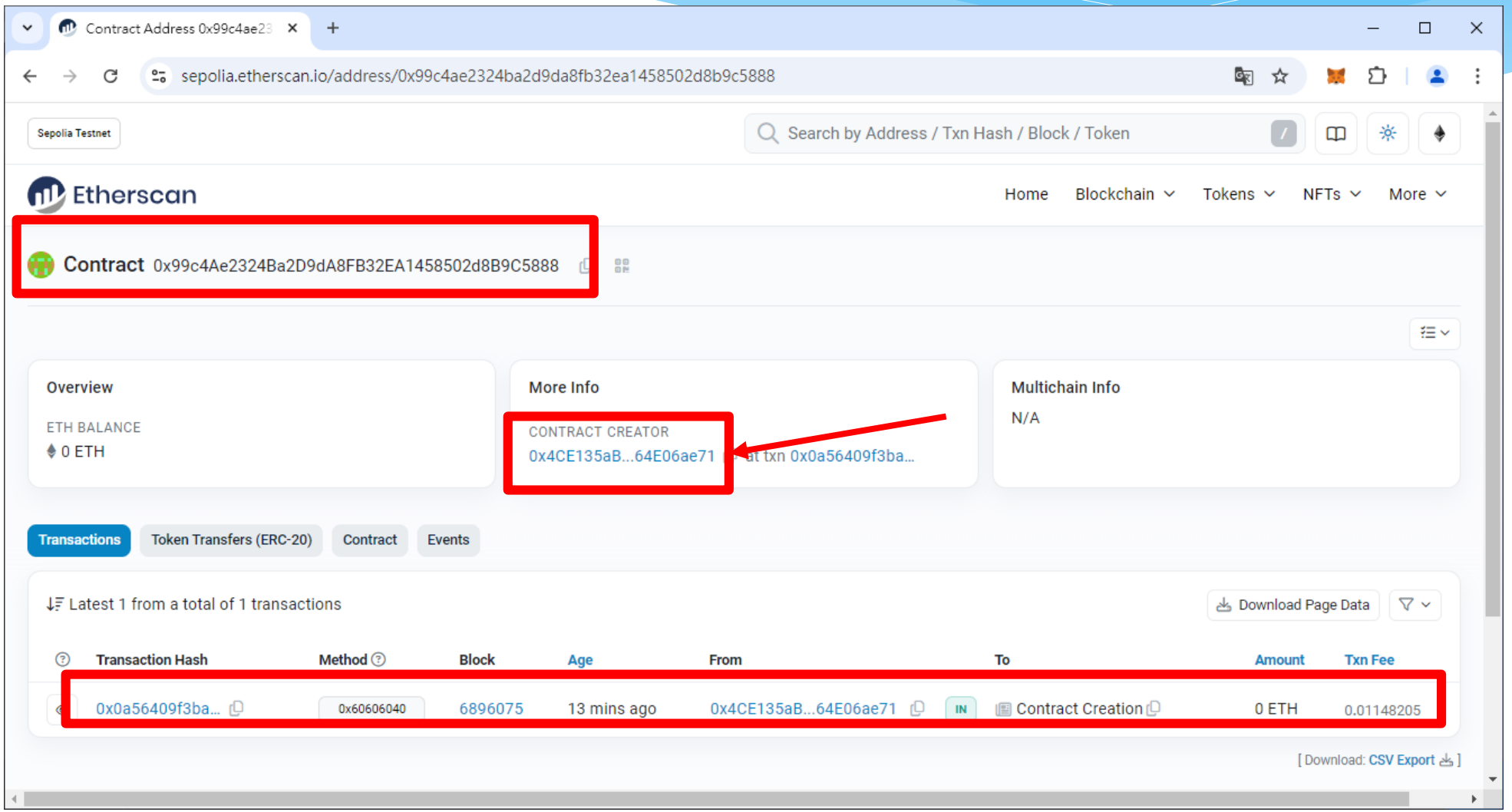
https://sepolia.etherscan.io/tx/0x0a56409f3ba94e4e1c198f566217a27f2af711350c05d77668be038bb5496c68

View Transaction on Sepolia



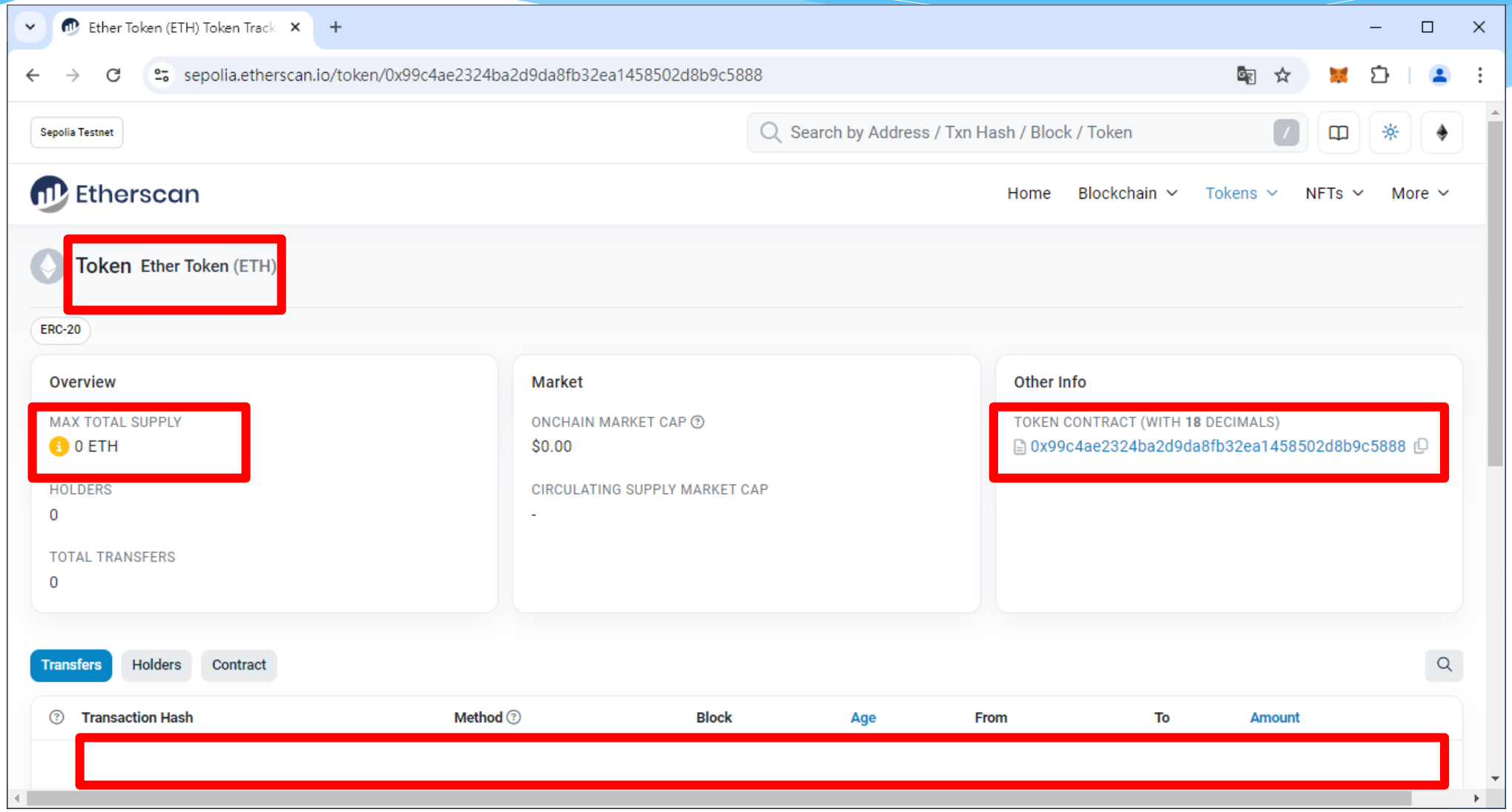
https://sepolia.etherscan.io/tx/0x0a56409f3ba94e4e1c198f566217a27f2af711350c05d77668be038bb5496c68

Smart Contract Information



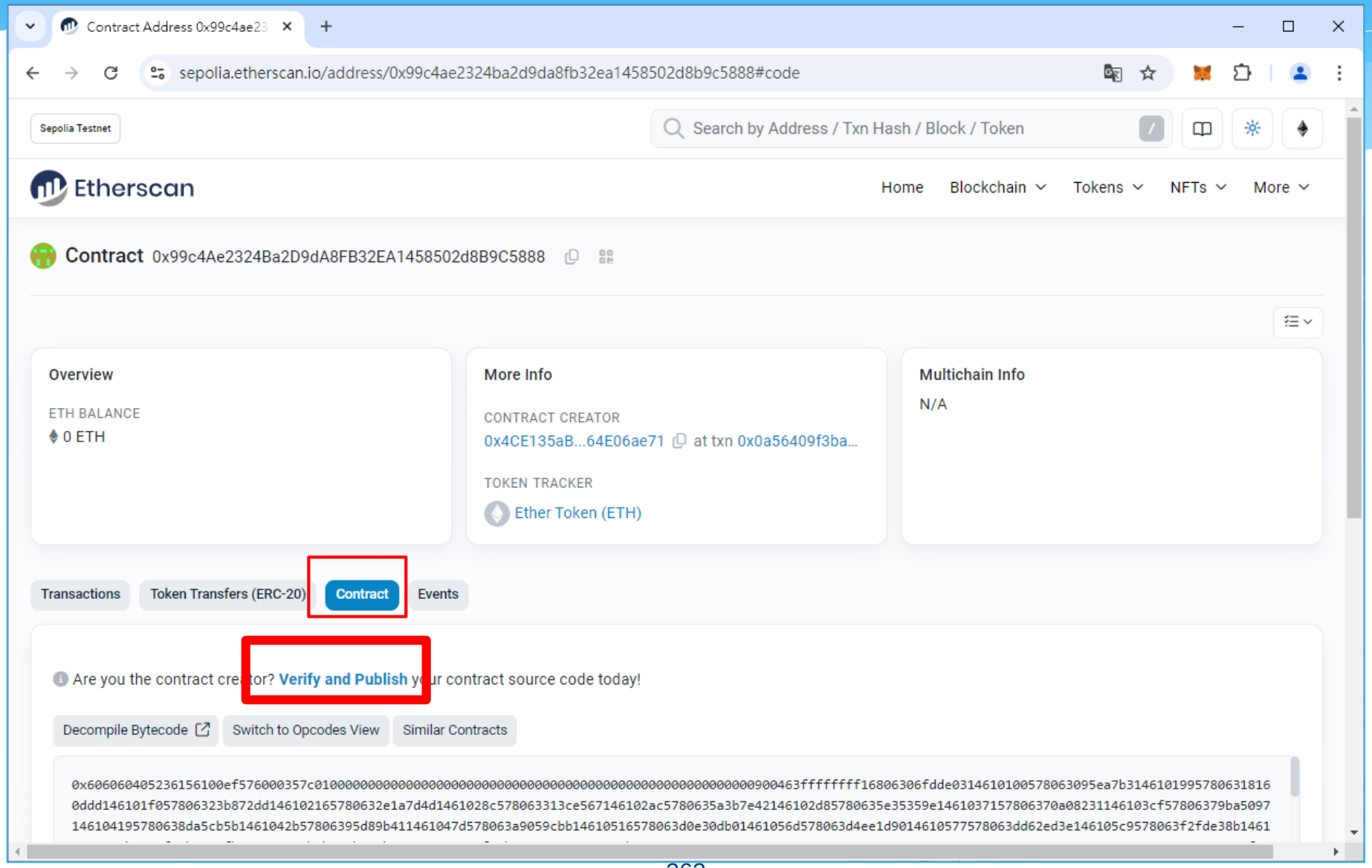
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888

Smart Contract Information



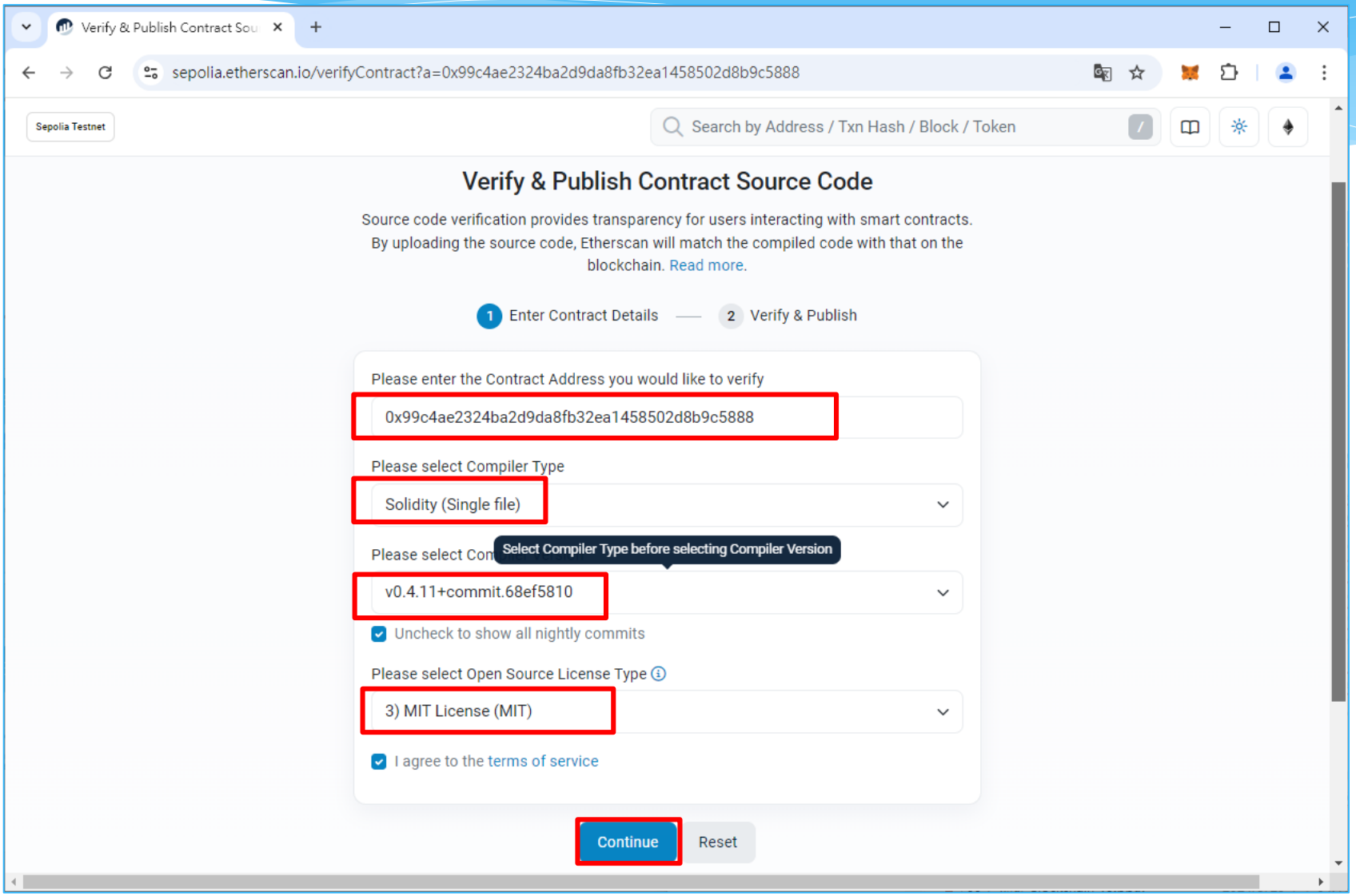
https://sepolia.etherscan.io/token/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888

View & Publish



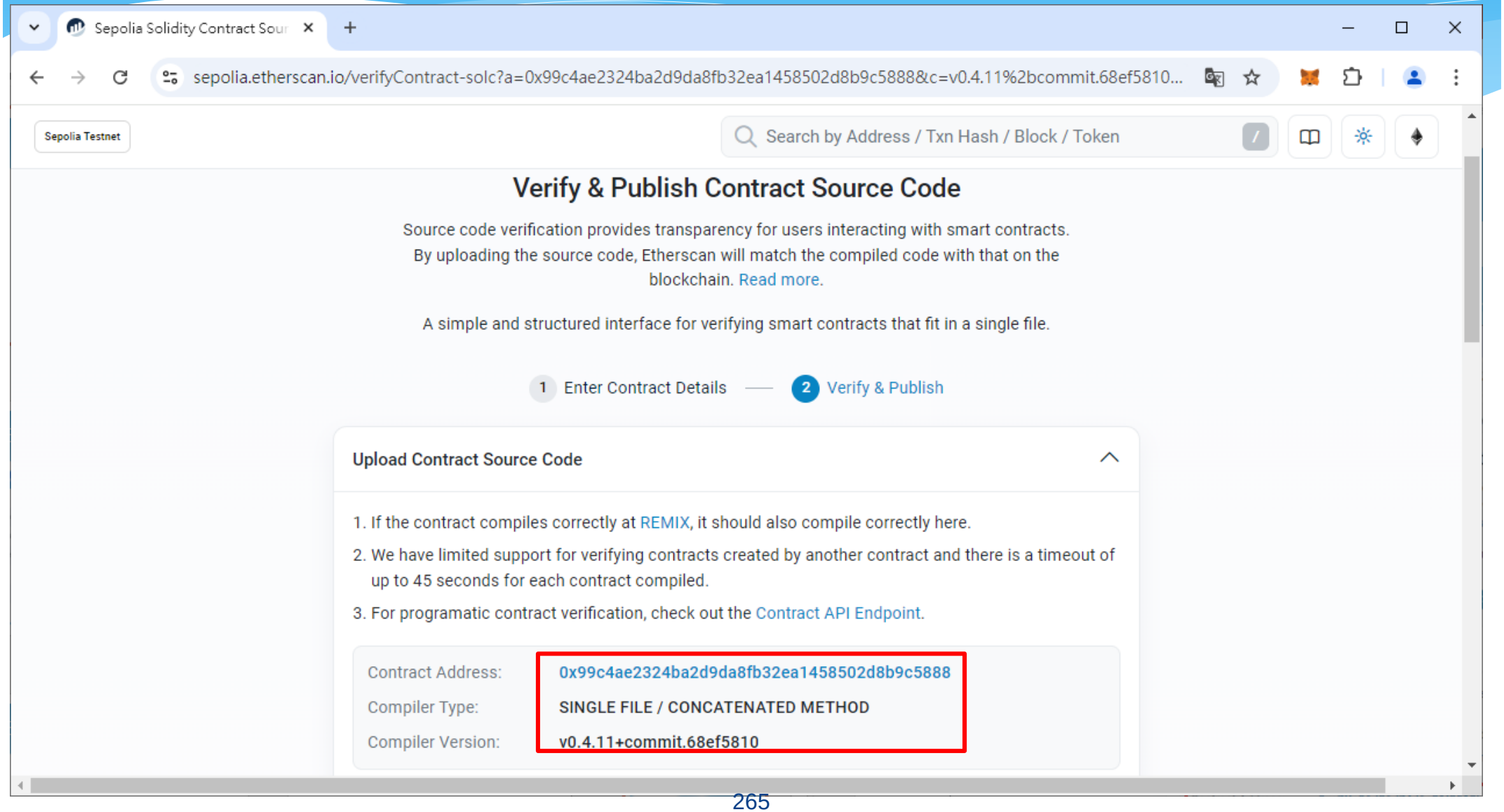
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#code

View & Publish Contract Source Code

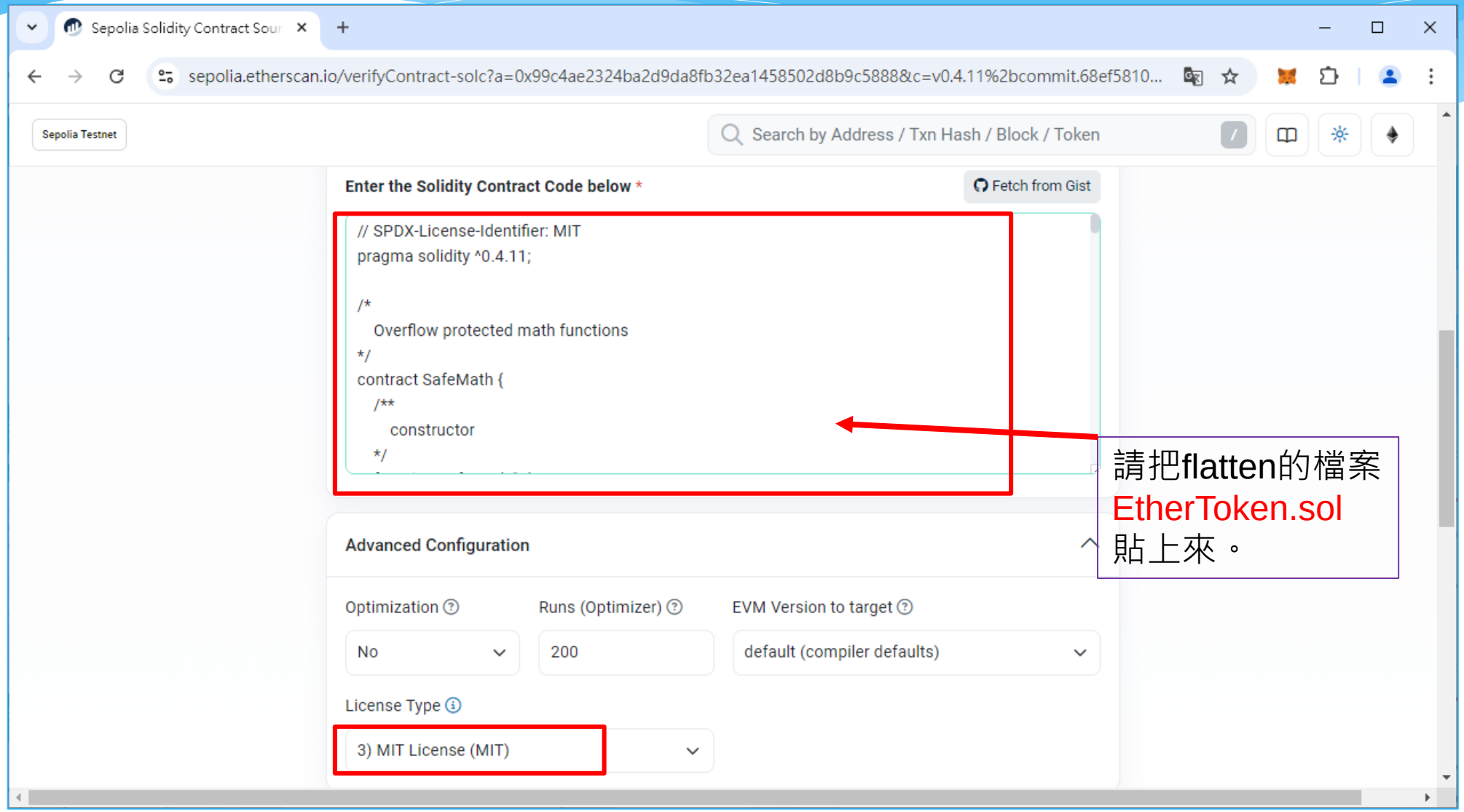


https://sepolia.etherscan.io/verifyContract?a=0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888

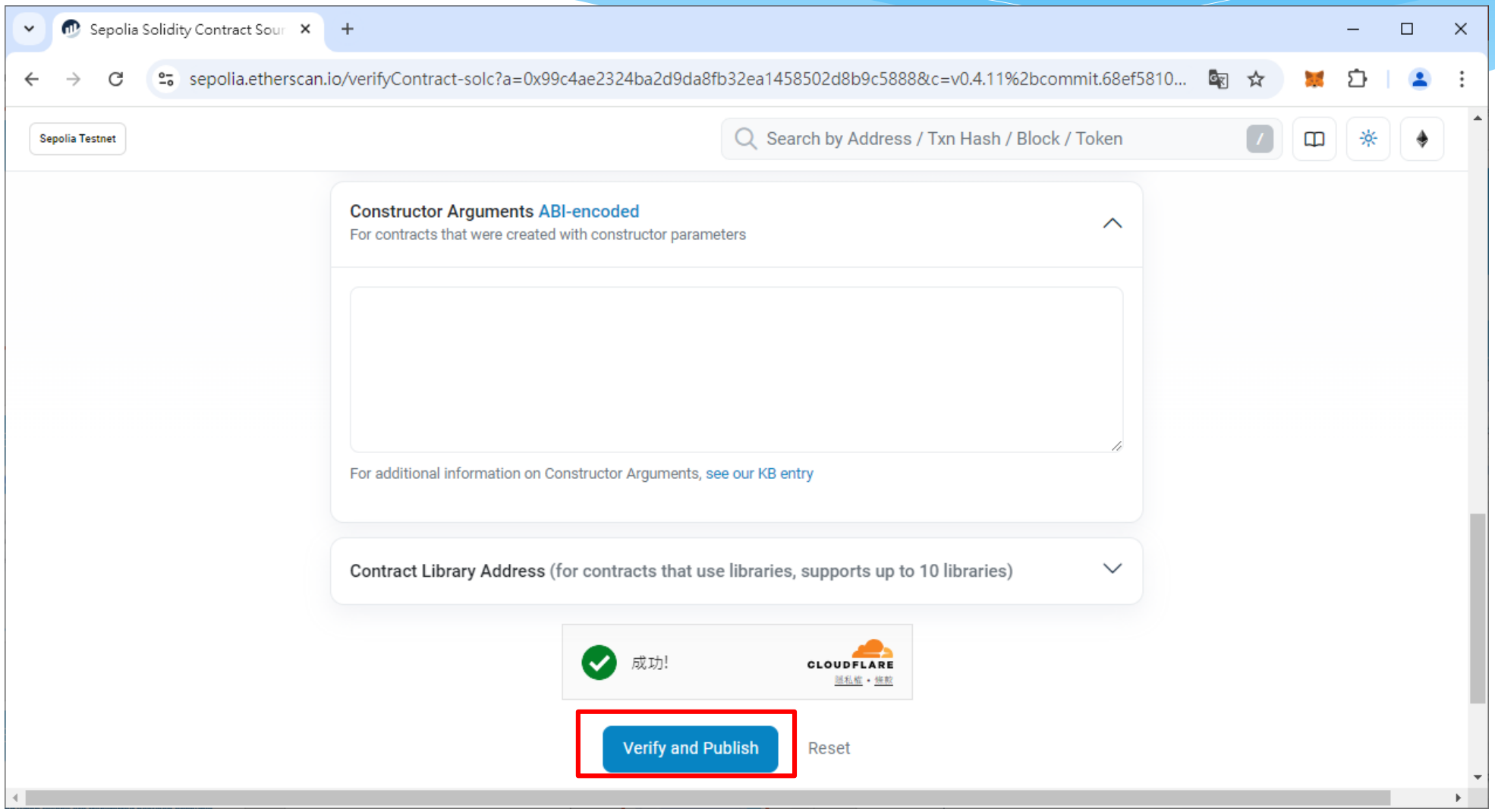
View & Publish Contract Source Code



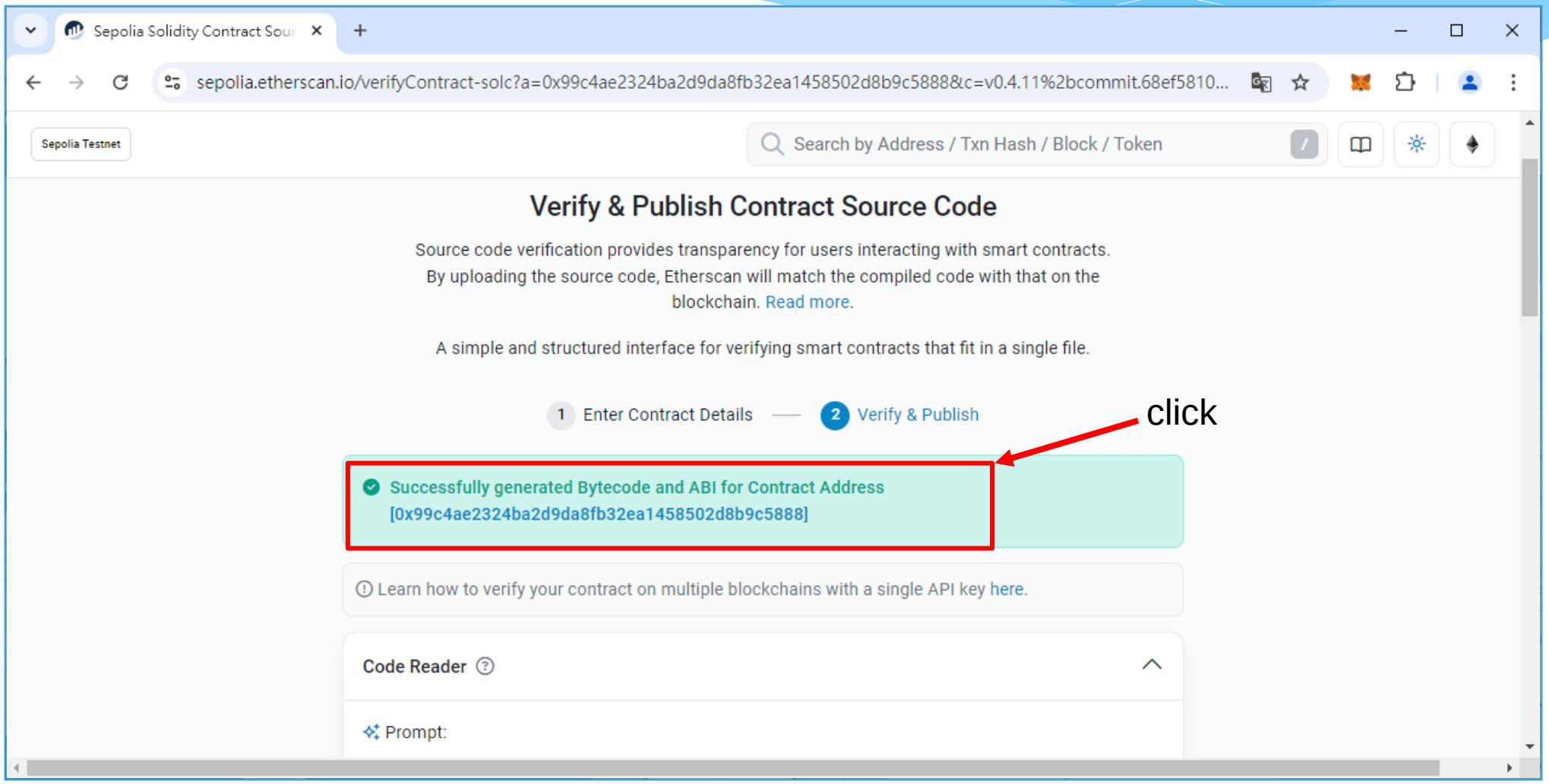
View & Publish Contract Source Code



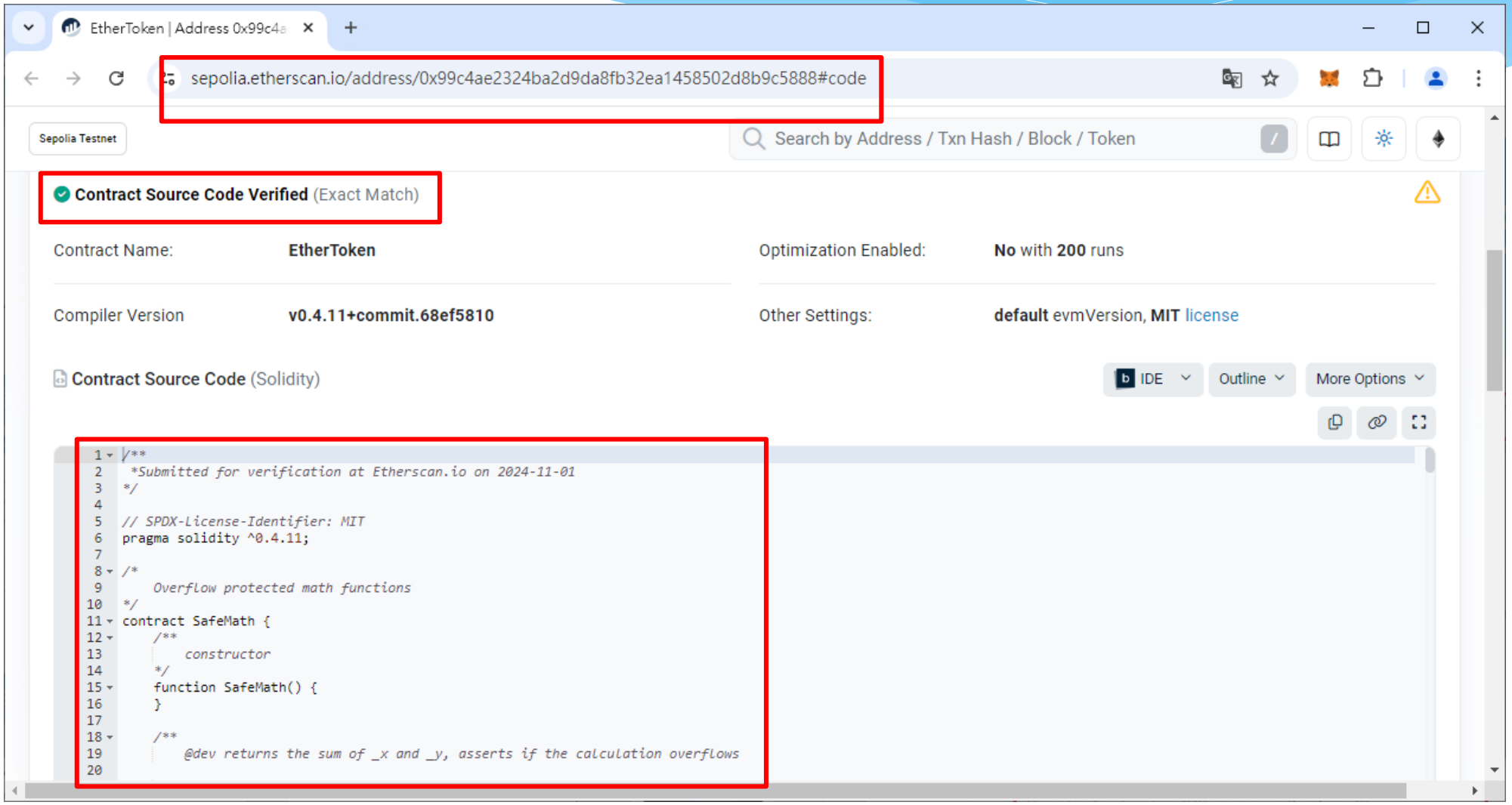
View & Publish Contract Source Code



View & Publish Contract Source Code

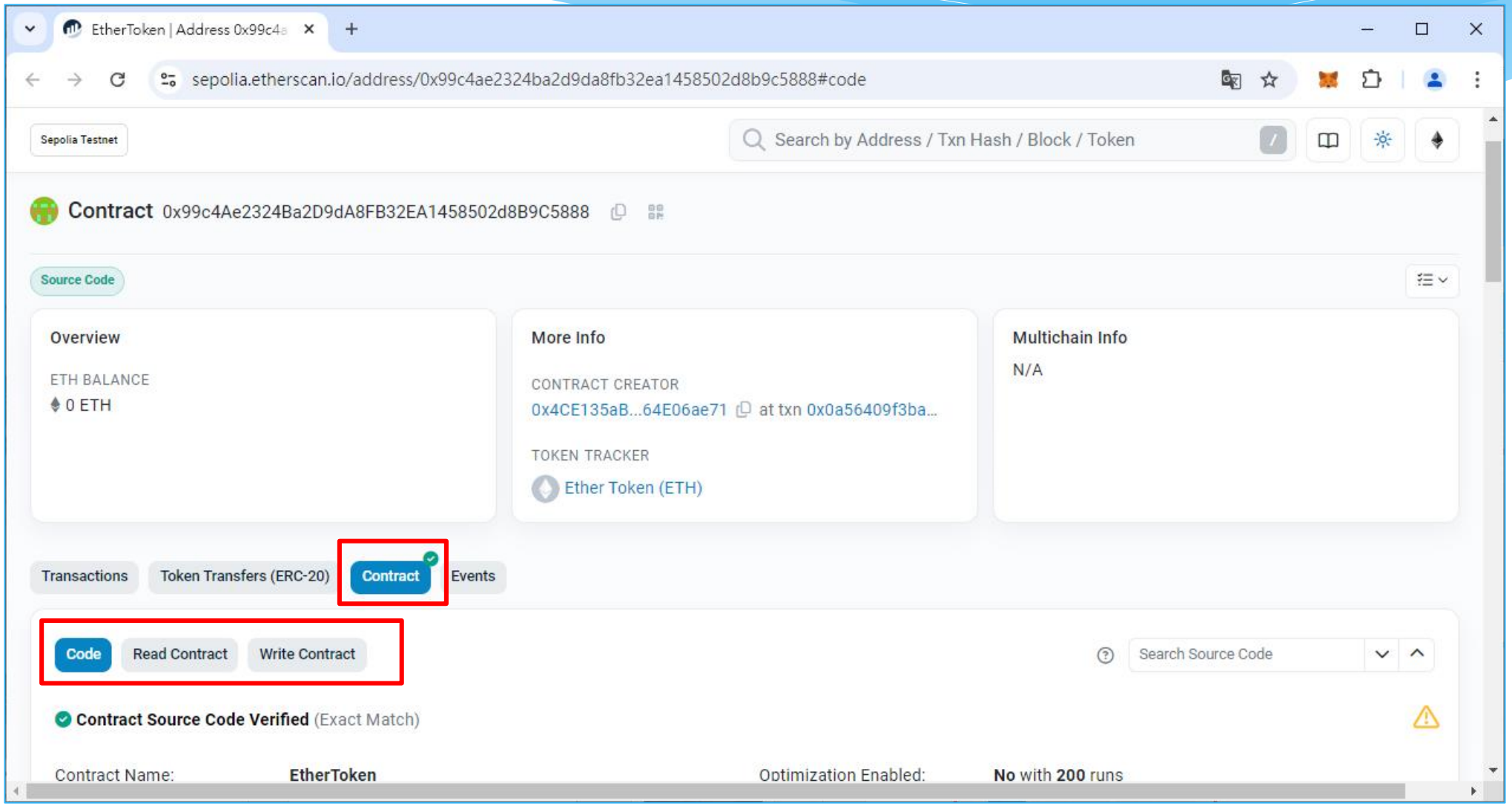


View Contract Source Code



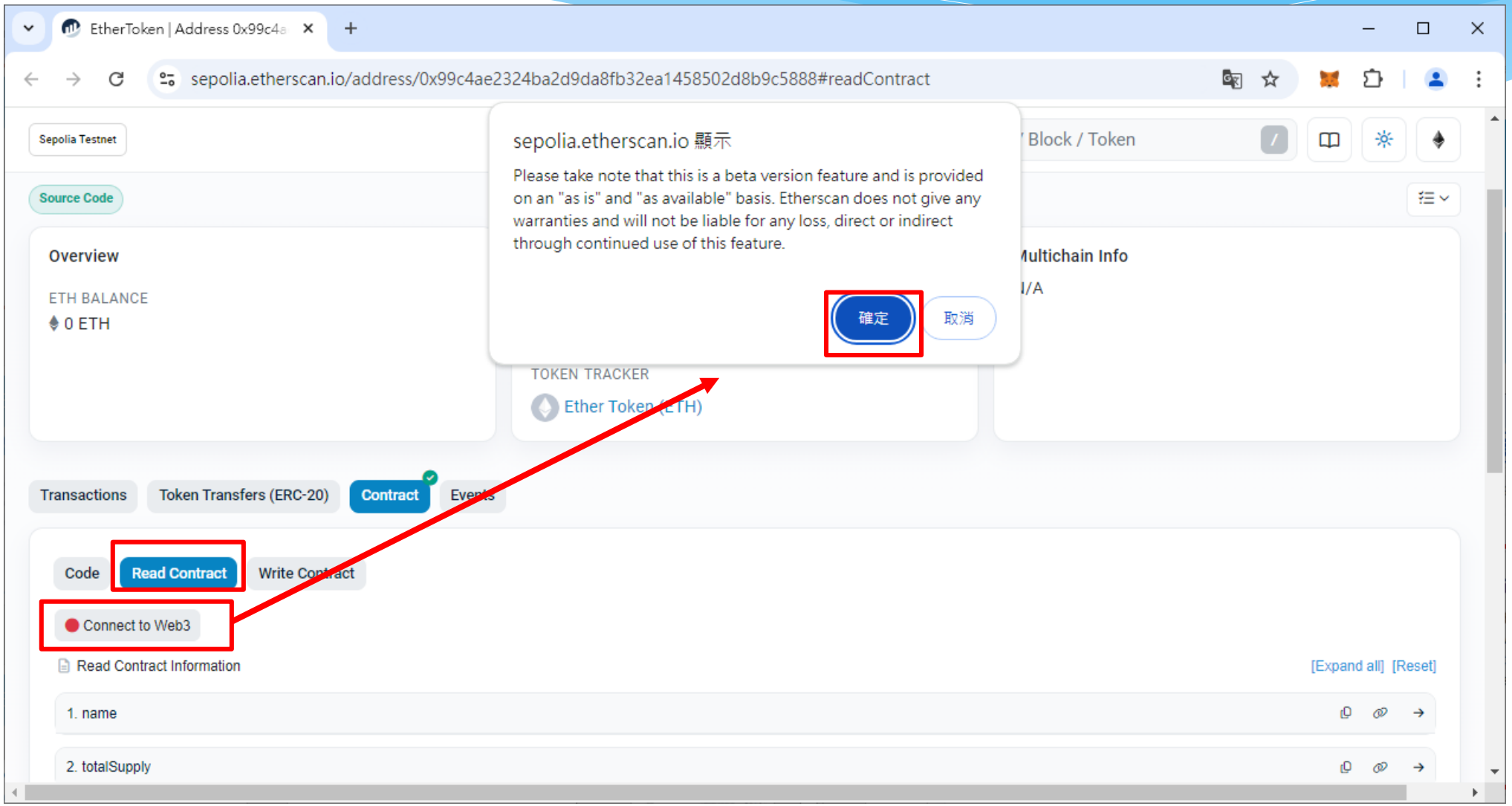
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#code

View & Publish Contract Source Code



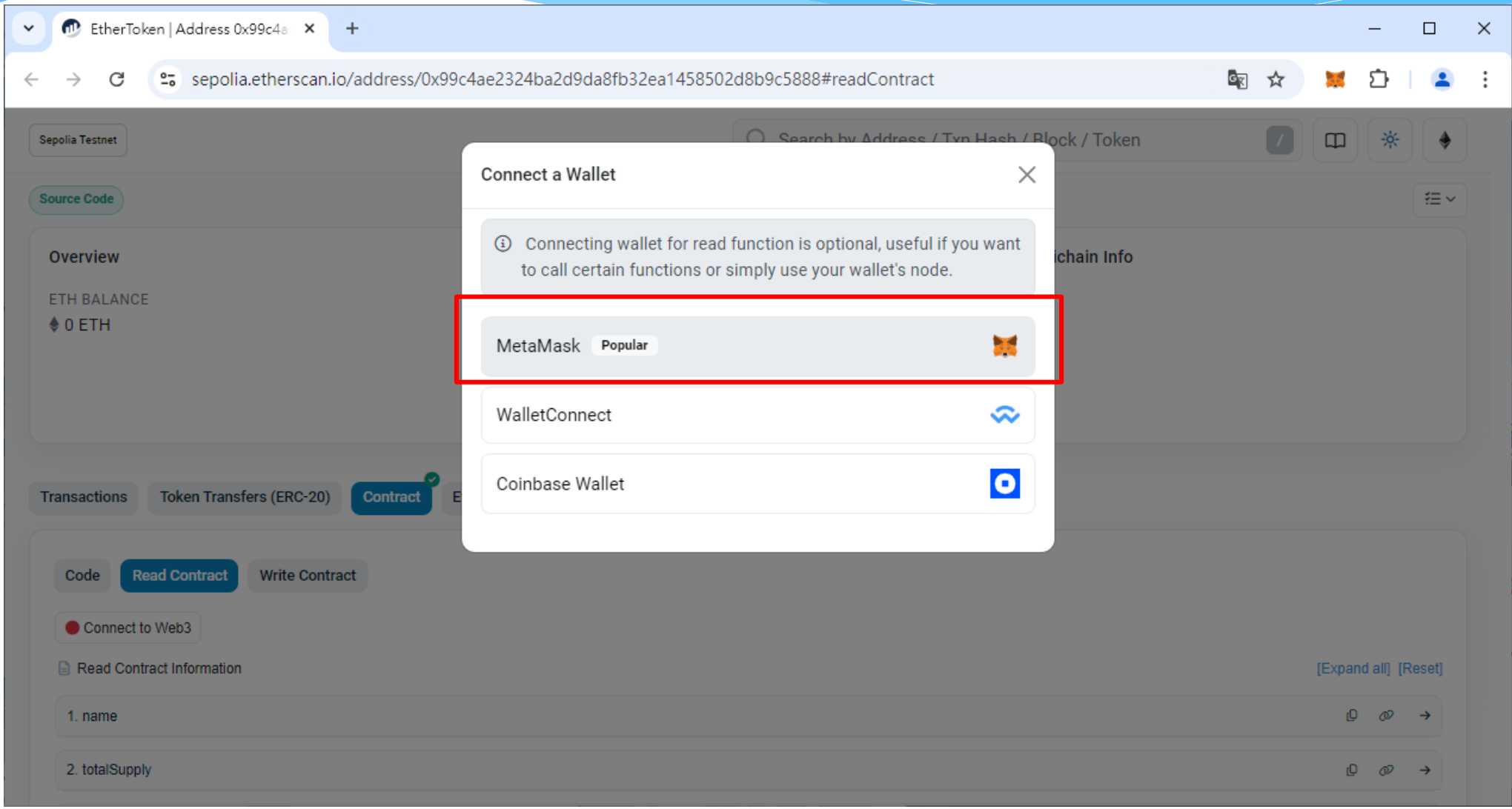
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#code

Connect to Web3



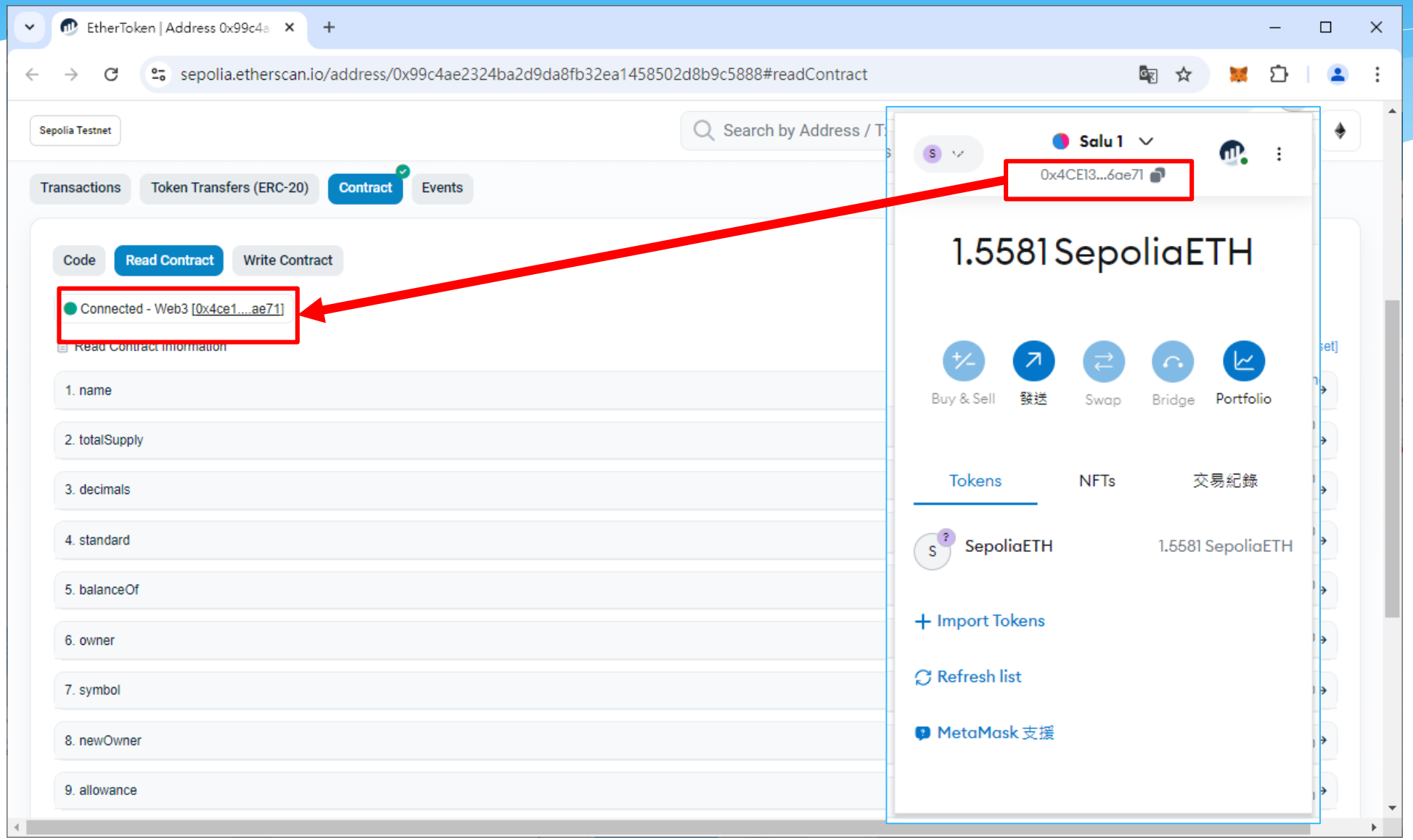
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#readContract

Connect a Wallet-MetaMask



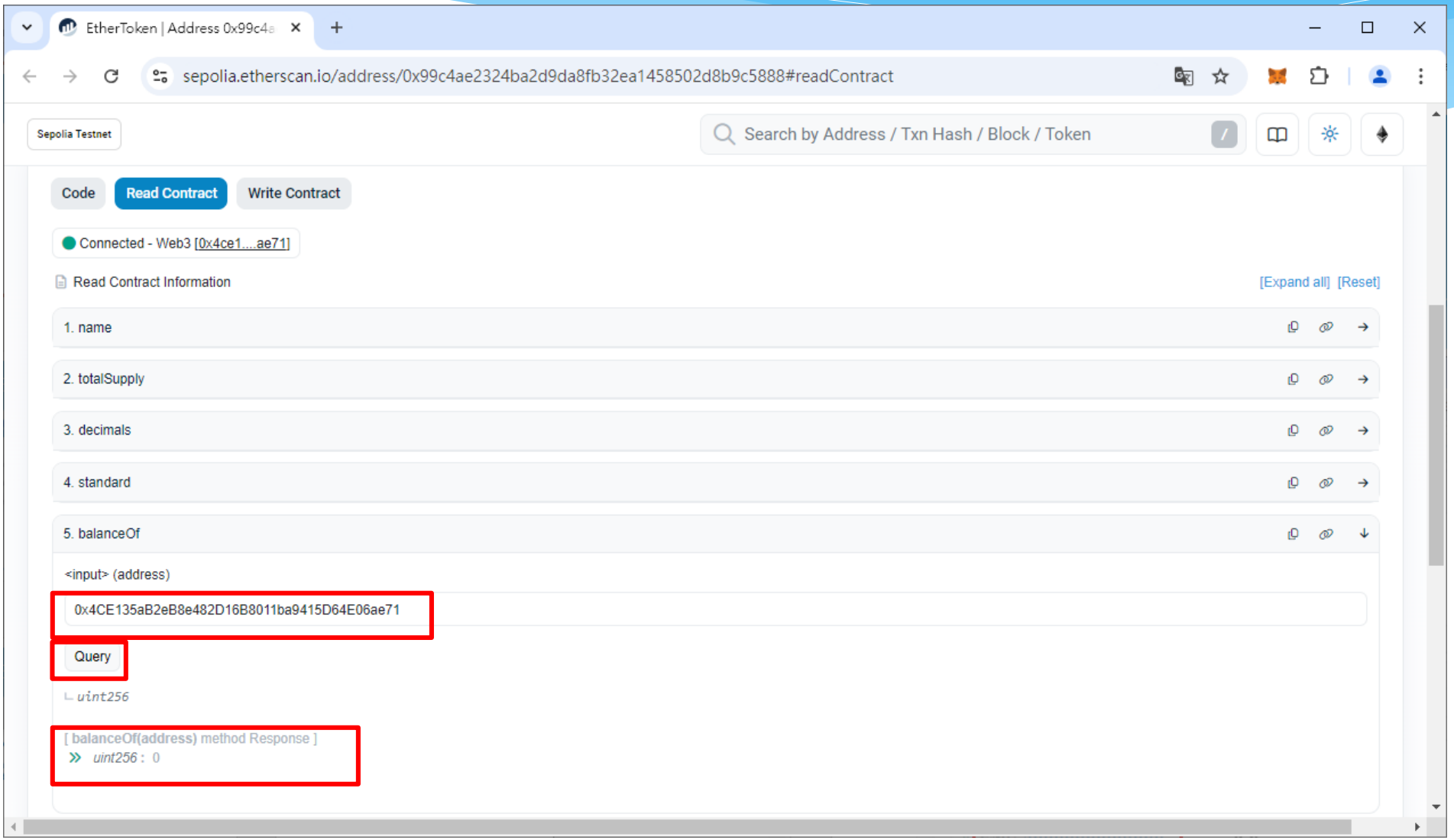
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#readContract

Connected Web3 using account



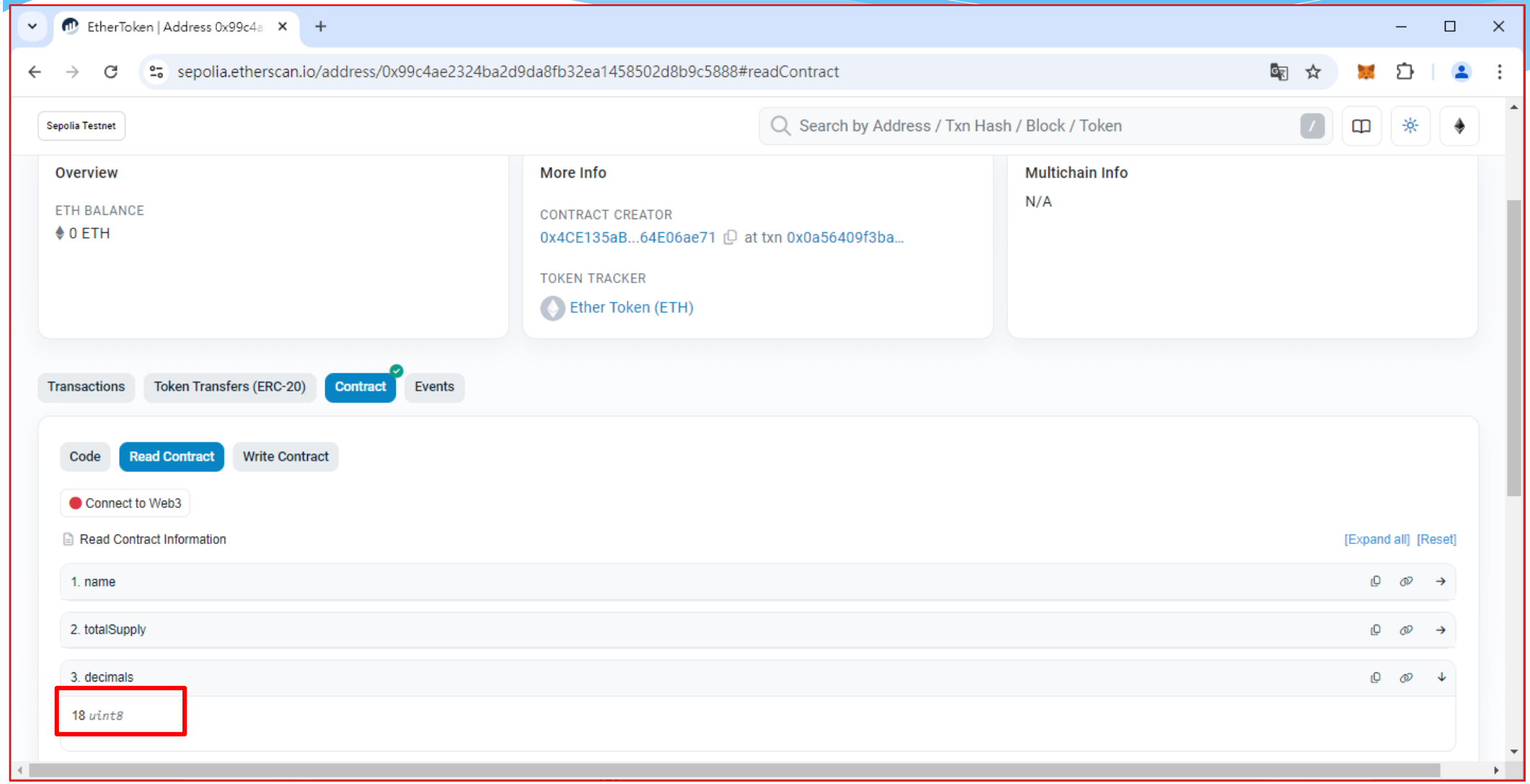
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#readContract

Read Contract - balanceOf



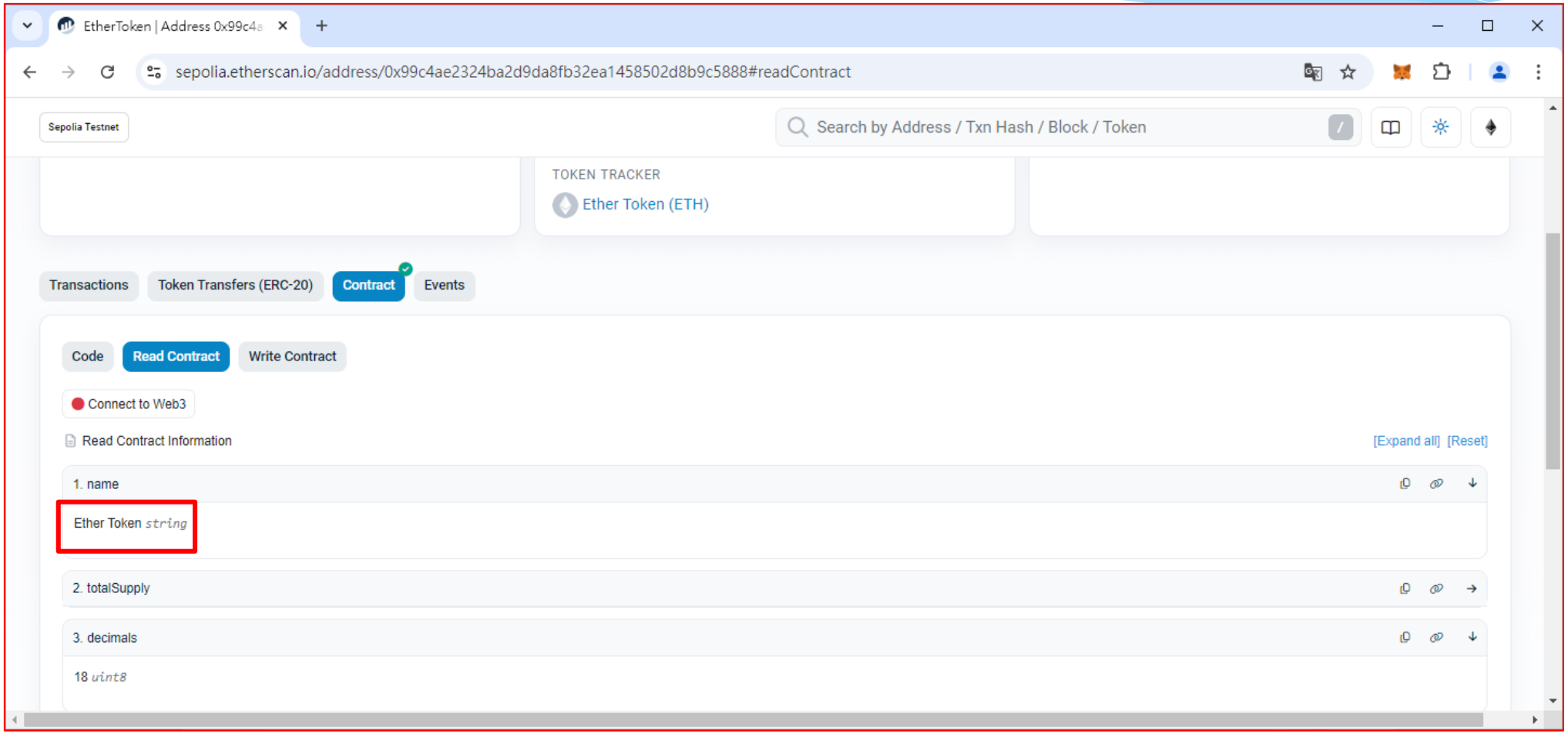
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#readContract

Read Contract-decimals



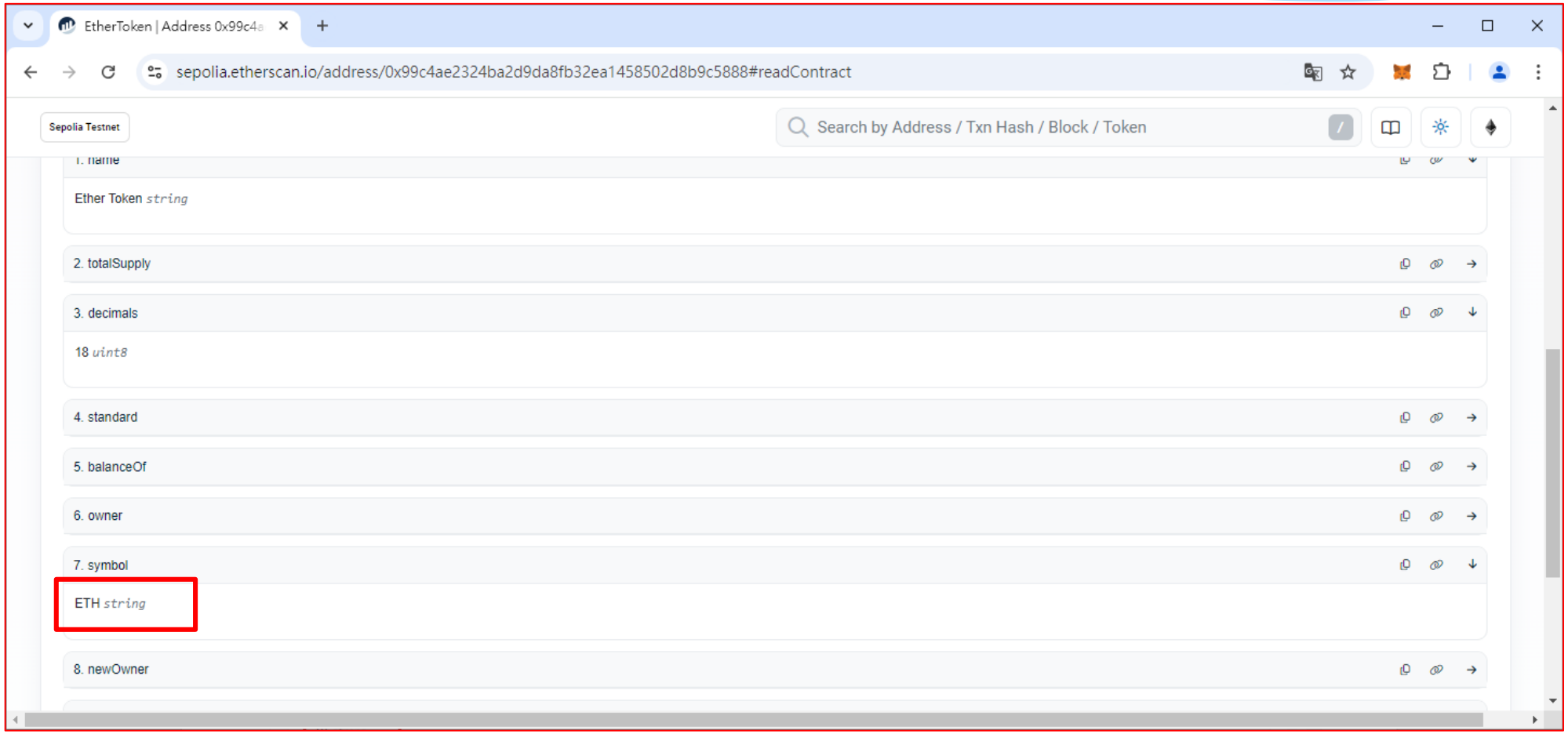
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#readContract

Read Contract-name



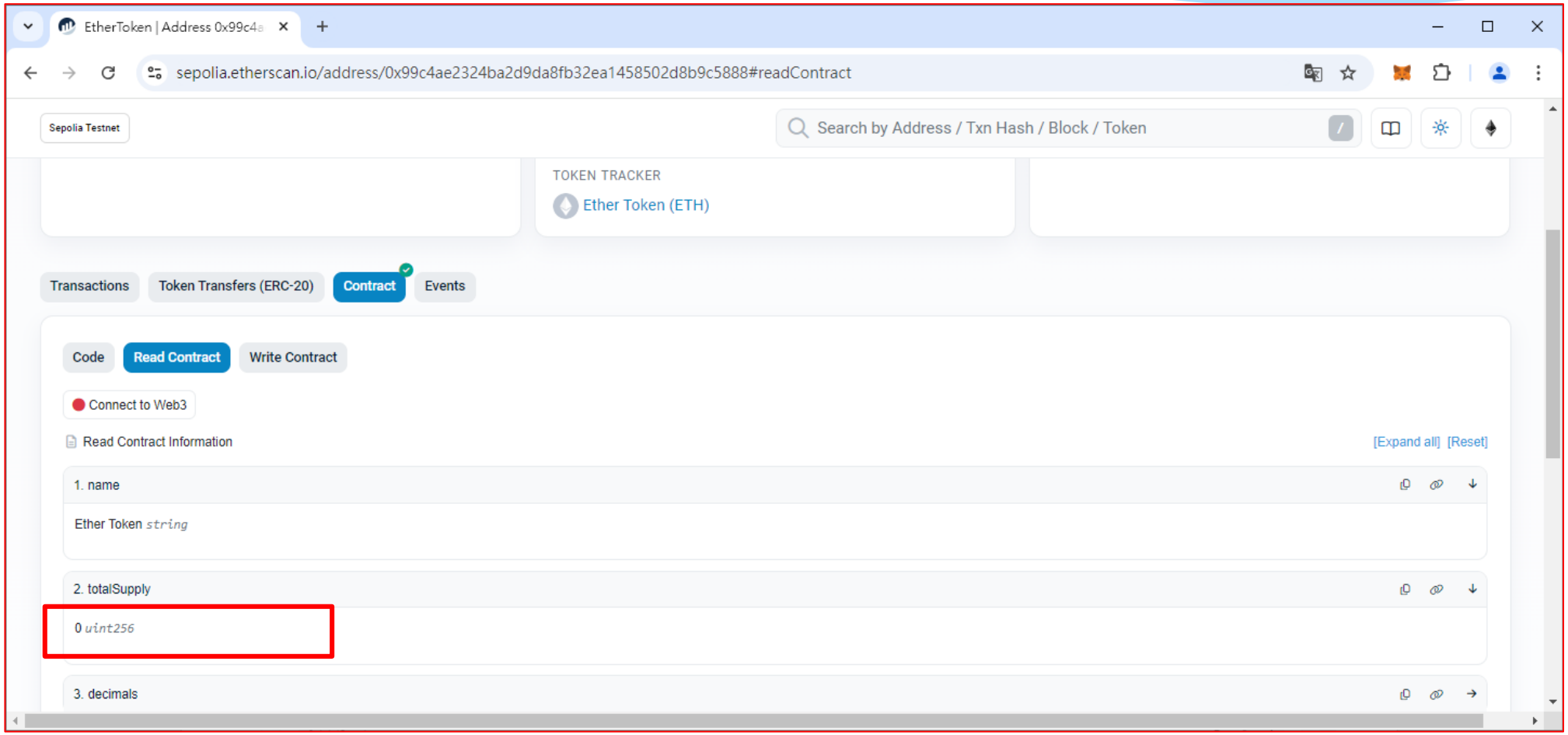
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#readContract

Read Contract-symbol



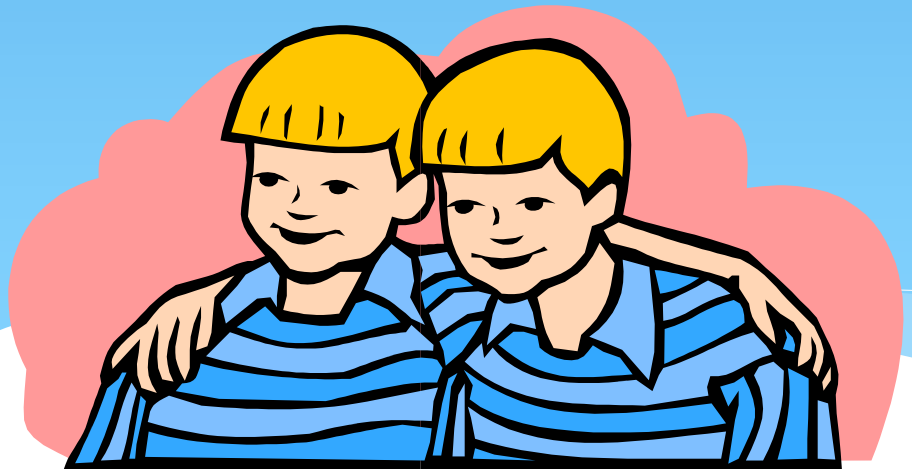
https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#readContract

Read Contract-totalSupply



https://sepolia.etherscan.io/address/0x99c4ae2324ba2d9da8fb32ea1458502d8b9c5888#readContract

Thank you ！



References

References

- MetaMask

  https://github.com/MetaMask/metamask-extension
- Remix

  https://github.com/ethereum/browser-solidity
- Openzeppelin

  https://docs.openzeppelin.com/

  https://github.com/OpenZeppelin/openzeppelin-contracts

References

- Ethereum
  - https://github.com/ethereum/
  - https://github.com/ethereum/go-ethereum
  - https://ethereum.org/en
  - https://geth.ethereum.org/docs/fundamentals/private-network
- Solidity Documentation
  - https://solidity.readthedocs.io
  - https://docs.soliditylang.org/en/v0.8.28/
- Smart Contract
  - https://www.quicknode.com/guides/tags/ethereum