Tópicos em Segurança da Informação Solidity e Ethereum

Ivan Sendin

FACOM - Universidade Federal de Uberlandia
ivansendin@yahoo.com,sendin@ufu.br

1 de outubro de 2024

- Bitcoin, Currencies, and Fragility
  - E uma critica a critica...
- Why bitcoin is worse than a Madoff-style Ponzi scheme

- Contrato self-executing e trustless
- 3 Bugs
  Exercicios....aguardem
- import, struct, address payable
- mapping
- memory (storage)
- this
- block.number
- msg.sender, msg.value
- balance

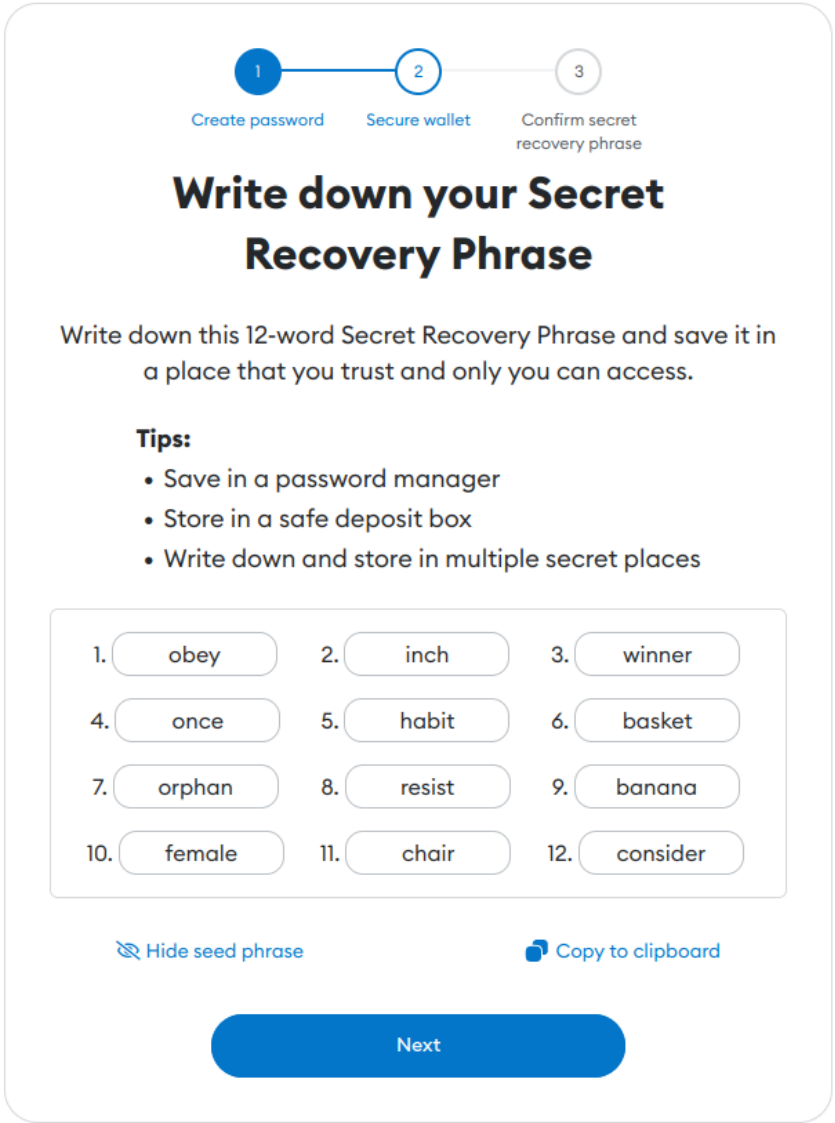
- Endereços de contratos
  Escondidos em objetos Python
- Externally owned addresses
  Usuários

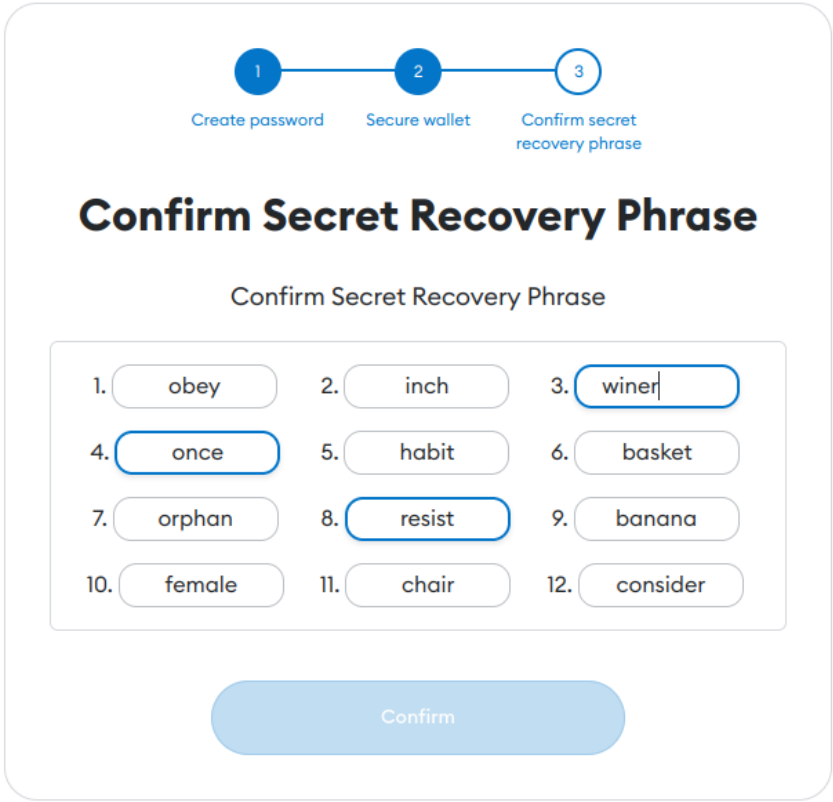
Carteiras

- Gerenciamento de EOA
- Controles SK
- Hierarquicas, palavras chave
- Comunicação com as redes Ethereum, similares e **testes**

- MetaMask
- My Ether Wallet
- Sempre existe a possibilidade de “sites falsos”
- Paper Wallet

- MetaMask
- Add on para a maioria dos navegadores
- Seleciona rede (CUIDADO!!), contas,...
- Aviso que nao existe recuperação de senhas


1
2
3
Create password
Secure wallet
Confirm secret recovery phrase
Write down your Secret Recovery Phrase
Write down this 12-word Secret Recovery Phrase and save it in a place that you trust and only you can access.
Tips:
• Save in a password manager
• Store in a safe deposit box
• Write down and store in multiple secret places
1. obey
2. inch
3. winner
4. once
5. habit
6. basket
7. orphan
8. resist
9. banana
10. female
11. chair
12. consider
Hide seed phrase
Copy to clipboard
Next


1
2
3
Create password
Secure wallet
Confirm secret recovery phrase
Confirm Secret Recovery Phrase
Confirm Secret Recovery Phrase
1. obey
2. inch
3. winer
4. once
5. habit
6. basket
7. orphan
8. resist
9. banana
10. female
11. chair
12. consider
Confirm

- My Ether Wallet - MEW
- Permite acesso a contratos
- Endereço e ABI
- “Segredo” da sua carteira!
- Não é seguro (eles mesmos falam!)
- Muito útil para testes!
- Conversa com o MetaMask
- (export da SK)

- Dos desenvolvedore do MEW
- enkrypt
- Ethereum, Polkadot and AVAX
- Também tem um sistema de palavras

Faucet

- Bica, fonte d'agua, torneira publica,...
- Conseguir moedas de graça
- Redes de testes
- Rede SEPOLIA
- Sepolia Faucet
  Cadastro no Alchemy (Nao precisa colocar os dados para cobrança!!)
  0.5 SepoliaEthe a cada 24 horas

Gas

- O seu SC roda nos computadores espalhados pelo mundo
  De forma redundante
  Diferente de uma nuvem
- Alguem precisa pagar pelo serviço
- Gas
- Cotação Gas X Ether (X US$)
- Esta moeda "separada" (cotação separada) permite uma dissociação

Gas

- Cada instrução da EVM tem um custo
  Boas praticas de programação
- Computação off chain
- Cada byte armazenado também
  storage/memory
- A "carteira" cuida do pagamento
- A carteira paga para rodar o contrato A, que chama um contrato B, que chama...tudo por conta de quem iniciou a transação
- Define um Gas Máximo para a transação
  Por enquanto tudo escondido/default
- O gas não é revertido no rollback(require)

Gas

- Pode pagar mais...(tente no MetaMask)
- pressa
- Ataques!!
  frontrunning

Mapping

- Estrutura chave/valor
- Todas as chaves estão inicializadas com "zero"
- Pode exigir algum esforço de programação
  `mapping (address=>uint) public balances;`
- Sem iterator!!!
- Iterate em mapping: aqui e aqui

Modifiers

```
function XYZ(<param types>)
{internal|external} [constant] [payable]
[returns(<ret types>)]
```

- Existem outros...
- Podem ser definidos pelos usuários...

```
contract meuContrato {
    ...
    address owner;

modifier onlyOwner() {
  require (msg.sender==owner,"Only Woner");
  _;
}


constructor(...) public payable {
  ...
  owner = msg.sender;
...
}

function addClient(address c) public onlyOwner {
  ...
  }

 function changeOwner(address o) public onlyOwner {
   owner=o;
  }
```

```
contract exemplo1 {

   struct jurorInfo {
     bool exists;
     ..
   }

   mapping(address=>jurorInfo) jury;
   address[] jurorIter;

  modifier onlyJuror() {
   require (jury[msg.sender].exists,"Only Juror!");
   _;
 }

  ....

   function addJuror(address j) public onlyOwner {
      require (jury[j].exists == false,"Please, I need a new juror!");
      jurorInfo storage ji = jury[j];
      ji.exists = true;
      jury[j] = ji;
      jurorIter.push(j);
  }
```

LUPA

- Lowest-Unmatched Price Auctions
- Menor lance único
- 1,2,5,1,2,3,7
- O 3 é o menor lance único
- "Mais para jogo"
  - Procuradoria investiga leilões na TV (FSP 2007)
  - Jusbrasil
- O lance deveria ser **secreto**...

LUPA

- Lances em ether
  É uma aposta...
  Lance fica para o leiloeiro!
- Artigo
- Premio em ether
- (nesta implementação: sem token, sem colateral, sem **punicoes**,...)

```
  contract LUPA {

enum LUPAStates { Bid, Payment, Finished}

struct BidValue {
  uint value;
  bool isUnmatched;
  address payable[] bidders;
}

mapping (uint =>  BidValue) bids;
uint blocklimit;
LUPAStates myState;
uint prizeValue;
address payable owner;

modifier onlyOwner {
  require (msg.sender == owner,
          "Sorry!");
  _;
}
```

```
constructor(uint time) payable public {
    ....
}

 function bid() public payable {
   ....
   trata o msg.value
 }
```