

Confronta il testo
Trova la differenza tra
due file di testo

D Diffchecker Desktop

The most secure way to run Diffchecker. Get the Diffchecker Desktop app:
your diffs never leave your computer!

[Get Desktop](#)

Accelerate l'elaborazione dei tuoi dati in un ambiente sicuro e con una fatturazione
trasparente. 200€ di credito gratuito

ADS VIA CARBON

Real-time diff
Unified diff
Collapse lines
HIGHLIGHT CHANGE
Parola Lettera
SYNTAX HIGHLIGHTING
Choose syntax
STRUMENTI
Convertire in minuscolo
Ordinare le righe
Sostituire le interruzioni
di riga con spazi
Tagliare gli spazi bianchi

Confrontare e unire

Esportare su formato
PDF

Editor

Untitled diff

- 0 removals

123 lines [Copia tutti](#)

+ 4 additions

124 lines [Copia tutti](#)

```

1 use borsh::{BorshDeserialize, BorshSerialize};
2 use solana_program::*;
3 account_info:::{next_account_info, AccountInfo},
4 entrypoint::ProgramResult,
5 program:::{invoke, invoke_signed},
6 program_error::ProgramError,
7 pubkey::Pubkey,
8 rent::Rent,
9 system_instruction,
10 sysvar::Sysvar,
11 };
12
13 use crate::{Wallet, WalletInstruction, WALLET_LEN};
14
15 pub fn process_instruction(
16     program_id: &Pubkey,
17     accounts: &[AccountInfo],
18     mut instruction_data: &[u8],
19 ) -> ProgramResult {
20     match WalletInstruction::deserialize(&mut instruction_data)? {
21         WalletInstruction::Initialize => initialize(program_id, accounts),
22         WalletInstruction::Deposit { amount } => deposit(program_id, accounts, amo
unt),
23         WalletInstruction::Withdraw { amount } => withdraw(program_id, accounts, a
mount),
24     }
25 }
26
27 fn initialize(program_id: &Pubkey, accounts: &[AccountInfo]) -> ProgramResult {
28     let account_info_iter = &mut accounts.iter();
29     let wallet_info = next_account_info(account_info_iter)?;
30     let vault_info = next_account_info(account_info_iter)?;
31     let authority_info = next_account_info(account_info_iter)?;
32     let rent_info = next_account_info(account_info_iter)?;
33     let (wallet_address, wallet_seed) =
34         Pubkey::find_program_address(&[authority_info.key.to_bytes()], program_i
d);
35     let (vault_address, vault_seed) = Pubkey::find_program_address(
36         &[authority_info.key.to_bytes(), &"VAULT".as_bytes()],
37         program_id,
38     );
39
40     let rent = Rent::from_account_info(rent_info)?;
41
42     assert_eq!(*wallet_info.key, wallet_address);
43     assert!(!wallet_info.data.is_empty());
44
45     invoke_signed(
46         &system_instruction::create_account(
47             &authority_info.key,
48             &wallet_address,
49             rent.minimum_balance(WALLET_LEN as usize),
50             WALLET_LEN,
51             &program_id,
52         ),
53         &[authority_info.clone(), wallet_info.clone()],
54         &[&[authority_info.key.to_bytes(), &[wallet_seed]]],
55     )?;
56
57     invoke_signed(
58         &system_instruction::create_account(
59             &authority_info.key,
60             &vault_address,
61             rent.minimum_balance(0),
62             0,
63             &program_id,
64         ),
65         &[authority_info.clone(), vault_info.clone()],
66         &[{
67             &authority_info.key.to_bytes(),
68             &"VAULT".as_bytes(),
69             &[vault_seed],
70         }],
71     )?;
72
73     let wallet = Wallet {
74         authority: *authority_info.key,
75         vault: vault_address,
76     };
77
78     wallet
79         .serialize(&mut &mut (*wallet_info.data).borrow_mut()[])
80         .unwrap();
81
82     Ok(())
83 }
84
85 fn deposit(_program_id: &Pubkey, accounts: &[AccountInfo], amount: u64) -> Program
Result {
86     let account_info_iter = &mut accounts.iter();
87     let wallet_info = next_account_info(account_info_iter)?;
88     let vault_info = next_account_info(account_info_iter)?;
89     let source_info = next_account_info(account_info_iter)?;
90     let wallet = Wallet::deserialize(&mut &(*wallet_info.data).borrow_mut()[])?;
91
92     assert_eq!(wallet.vault, *vault_info.key);
93
94     invoke(
95         &system_instruction::transfer(&source_info.kev, &vault_info.kev, amount).

```



Accelerate l'elaborazione dei tuoi dati in un ambiente sicuro e con una fatturazione
trasparente. 200€ di credito gratuito

ADS VIA CARBON

[Eliminare](#)

[Salvare](#)

[Condividere](#)

```

96     &[vault_info.clone(), source_info.clone()], // ...
97 )?;
98
99 Ok(())
100 }
101
102 fn withdraw(_program_id: &Pubkey, accounts: &[AccountInfo], amount: u64) -> ProgramResult {
103     let account_info_iter = &mut accounts.iter();
104     let wallet_info = next_account_info(account_info_iter)?;
105     let vault_info = next_account_info(account_info_iter)?;
106     let authority_info = next_account_info(account_info_iter)?;
107     let destination_info = next_account_info(account_info_iter)?;
108     let wallet = Wallet::deserialize(&mut &(*wallet_info.data).borrow_mut()[])?;
109
110     assert!(authority_info.is_signer);
111     assert_eq!(wallet.authority, *authority_info.key);
112
113     assert_eq!(wallet.vault, *vault_info.key);
114
115     if amount > **vault_info.lamports.borrow_mut() {
116         return Err(ProgramError::InsufficientFunds);
117     }
118
119     **vault_info.lamports.borrow_mut() -= amount;
120     **destination_info.lamports.borrow_mut() += amount;
121
122     Ok(())
123 }

```

```

96     &[vault_info.clone(), source_info.clone()], // ...
97 )?;
98
99 Ok(())
100 }
101
102 fn withdraw(_program_id: &Pubkey, accounts: &[AccountInfo], amount: u64) -> ProgramResult {
103     let account_info_iter = &mut accounts.iter();
104     let wallet_info = next_account_info(account_info_iter)?;
105     let vault_info = next_account_info(account_info_iter)?;
106     let authority_info = next_account_info(account_info_iter)?;
107     let destination_info = next_account_info(account_info_iter)?;
108     let wallet = Wallet::deserialize(&mut &(*wallet_info.data).borrow_mut()[])?;
109
110     // Vulnerabilità: rimosso il controllo della chiave dell'autorità
111     // assert!(authority_info.is_signer);
112     // assert_eq!(wallet.authority, *authority_info.key);
113
114     assert_eq!(wallet.vault, *vault_info.key);
115
116     if amount > **vault_info.lamports.borrow_mut() {
117         return Err(ProgramError::InsufficientFunds);
118     }
119
120     **vault_info.lamports.borrow_mut() -= amount;
121     **destination_info.lamports.borrow_mut() += amount;
122
123     Ok(())
124 }

```

Diff salvati < Testo originale ↑ Open file Testo modificato < Open file

Your saved diffs will appear here.

```

111     assert_eq!(wallet.authority, *authority_info.key);
112     assert_eq!(wallet.vault, *vault_info.key);
113
114     if amount > **vault_info.lamports.borrow_mut() {
115         return Err(ProgramError::InsufficientFunds);
116     }
117
118     **vault_info.lamports.borrow_mut() -= amount;
119     **destination_info.lamports.borrow_mut() += amount;
120
121     Ok(())
122 }

```

```

93
94     invoke(
95         &system_instruction::transfer(&source_info.key, &vault_info.key, amount),
96         &[vault_info.clone(), source_info.clone()],
97     )?;
98
99     Ok(())
100 }
101
102 fn withdraw(_program_id: &Pubkey, accounts: &[AccountInfo], amount: u64) -> ProgramResult {
103     let account_info_iter = &mut accounts.iter();
104     let wallet_info = next_account_info(account_info_iter)?;

```



AD Deliver software faster
GitLab is the most comprehensive AI-powered DevSecOps Platform.
Software. Faster.

Build Software Fast



Bibcitations
A free online tool to generate citations, reference lists, and bibliographies. APA, MLA, Chicago, and more.

Check it out

Trovare la differenza