

Confronta il testo
Trova la differenza tra
due file di testo



Diffchecker Desktop

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

[Get Desktop](#)

The DNS course
for developers



I've spent 2 years learning DNS while building NSLookup.io. Now, I'm teaching everything I know.

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

- 21 removals

123 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
36     let (vault_address, vault_seed) = Pubkey::find_program_address(
37         &[&authority_info.key.to_bytes(), &"VAULT".as_bytes()],
38         program_id,
39     );
40
41     let rent = Rent::from_account_info(rent_info)?;

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
58     invoke_signed(
59         &system_instruction::create_account(
60             &authority_info.key,
61             &vault_address,
62             rent.minimum_balance(0),
63             0,
64             &program_id,
65         ),
66         &[authority_info.clone(), vault_info.clone()],
67         &[&[
68             &authority_info.key.to_bytes(),
69             &"VAULT".as_bytes(),
70             &[vault_seed],
71         ]],
72     );
73     let wallet = Wallet {

```

Text moved to lines 59-63

```

35     let (wallet_address, wallet_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
58     invoke_signed(
59         &system_instruction::create_account(
60             &authority_info.key,
61             &vault_address,
62             rent.minimum_balance(0),
63             0,
64             &program_id,
65         ),
66         &[authority_info.clone(), vault_info.clone()],
67         &[&[
68             &authority_info.key.to_bytes(),
69             &"VAULT".as_bytes(),
70             &[vault_seed],
71         ]],
72     );
73     let wallet = Wallet {

```

+ 43 additions

143 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
34 // Utilizziamo lo stesso account_info per wallet e vault
35 let wallet = wallet_info.clone();
36 let vault = vault_info.clone();
37
38 // Creazione dell'account wallet
39 let (wallet_address, wallet_seed) =
40     Pubkey::find_program_address(&[&authority_info.key.to_bytes()], program_i
d);

41
42
43     let rent = Rent::from_account_info(rent_info)?;
44
45     assert_eq!(*wallet.key, wallet_address);
46     assert!(wallet.data_is_empty());

47
48     invoke_signed(
49         &system_instruction::create_account(
50             &authority_info.key,
51             &wallet_address,
52             rent.minimum_balance(WALLET_LEN as usize),
53             WALLET_LEN,
54             &program_id,
55         ),
56         &[authority_info.clone(), wallet.clone()],
57         &[&[&authority_info.key.to_bytes(), &[wallet_seed]]],
58     );
59
60
61     // Creazione dell'account vault
62     Text moved from lines 35-39
63     let (vault_address, vault_seed) = Pubkey::find_program_address(
64         &[&authority_info.key.to_bytes(), &"VAULT".as_bytes()],
65         program_id,
66     );
67
68     assert_eq!(*vault.key, vault_address);
69     assert!(vault.data_is_empty());

70
71     invoke_signed(
72         &system_instruction::create_account(
73             &authority_info.key,
74             &vault_address,
75             rent.minimum_balance(0),
76             0,
77             &program_id,
78         ),
79         &[authority_info.clone(), vault.clone()],
80         &[&[
81             &authority_info.key.to_bytes(),
82             &"VAULT".as_bytes(),
83             &[vault_seed],
84         ]],
85     );
86
87     let wallet_data = Wallet {

```

Testo originale

Testo originale

Your saved diffs will appear here.

```
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 }
123
```

↑ Open file

Testo modificato

```
132     assert_eq!(wallet_data.vault, *vault.key);
133
134     if amount > **vault.lamports.borrow_mut() {
135         return Err(ProgramError::InsufficientFunds);
136     }
137
138     **vault.lamports.borrow_mut() -= amount;
139     *destination_info.lamports.borrow_mut() += amount;
140
141     Ok(())
142 }
```



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

Build Software Fast



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

[Check it out](#)

Trovare la differenza