

What Mallory did

- 1 She signs the transaction with her own key**
The Signer check only asks whether this key signed. It did.
[Check passes](#)
- 2 She passes in Alice's vault**
It really is owned by the Tip Vault program.
[Check passes](#)
- 3 The vault's type tag matches Vault**
It's a genuine vault, not a forgery.
[Check passes](#)
- 4 Nothing compares vault.authority to the signer**
Alice's lamports move to Mallory's wallet.
[Check missing](#)

Every check that ran was correct. The bug is the check that never ran.

The fix is **two constraints**

Vulnerable any signer, any vault

```
#[derive(Accounts)]
pub struct Withdraw<'info> {
    #[account(mut)]
    pub vault: Account<'info, Vault>,
    #[account(mut)]
    pub authority: Signer<'info>,
}
```

And note: the vulnerable version compiles cleanly. This is a logic bug, not a syntax error.

Fixed owner only

```
#[derive(Accounts)]
pub struct Withdraw<'info> {
    #[account(mut,
        seeds = [b"vault", authority.key().as_ref()],
        bump = vault.bump,
        has_one = authority)]
    pub vault: Account<'info, Vault>,
    #[account(mut)]
    pub authority: Signer<'info>,
}
```

has_one requires the vault's stored authority to equal the signer. **seeds** require the vault's address to be derived from the signer's key. **Either one closes this hole. Using both is defense in depth.**

Why Solana is different

	Ethereum (EVM)	Solana
Where state lives	Inside the contract's own storage	In separate accounts the program owns
Who's calling	One implicit msg.sender	Explicit Signer accounts, possibly several
What a transaction touches	Discovered as the code runs	Declared up front, so transactions run in parallel
Paying for storage	Gas per storage write	Refundable rent-exempt deposit sized to the account
Upgrades	Immutable unless you add a proxy	Upgradeable by default through an upgrade authority
Reentrancy	A classic attack class	Only direct self-recursion; call depth is capped
Typical bug	Reentrancy, unchecked external calls	Missing signer, owner or relationship checks

Reading any Solana program

- ☐ Is every authority account a Signer?
- ☐ Is every data account checked for owner and type?
- ☐ Is each account tied to the others with `has_one` or seeds?
- ☐ Are PDAs derived with the canonical bump?
- ☐ Is the target of every cross-program call pinned?
- ☐ Is all arithmetic checked?
- ☐ Can a closed account be revived or reinitialized?
- ☐ Who holds the upgrade authority?

DISCUSSION

Pick a Solana program you've used. Who holds its upgrade authority, and what could they change tomorrow?

Mallory needed no exploit code, only a crafted account list. **What does that mean for how we test these programs?**