

THINK LIKE AN ATTACKER

BRINGING SMART CONTRACTS TO THEIR BREAK(ING) POINT

ABOUT US



ABOUT US

▶ Ilias

▶ Co-Founder of Neodyme

▶ Twitter: @A2nkF_

▶ Thomas

▶ Co-Founder of Neodyme

▶ Twitter: @t_lambertz



ABOUT THIS WORKSHOP

ABOUT THIS WORKSHOP

“THINKING” LIKE AN ATTACKER

ABOUT THIS WORKSHOP

“THINKING” LIKE AN ATTACKER

(LIST OF COMMON BUGS
ON [BLOG.NEODYME.IO](https://blog.neodyme.io))

IS THIS BROKEN?

~~IS THIS BROKEN?~~

~~IS THIS BROKEN?~~

HOW DO I BREAK THIS?

WHAT DOES THE
CODE DO?

TRACE USER INPUT

SOLANA BASICS

FOR ATTACKERS

ACCOUNTS

ACCOUNTS


```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```


ACCOUNTS

```
pub struct AccountInfo<'a> {  
    → pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```


ACCOUNTS


- ▶ Used to identify an account



```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```

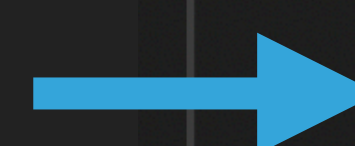

ACCOUNTS

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
    pub is_signer: bool,  
    // [...]  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
    pub owner: &'a Pubkey,  
    pub executable: bool,  
    // [...]  
}
```



ACCOUNTS

- ▶ Flag indicating if the account has signed the transaction



```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```



ACCOUNTS

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```


ACCOUNTS

- ▶ Account balance in lamports
- ▶ 1 SOL = 1_000_000_000 lamports
- ▶ Runtime asserts
total_lamports_before ==
total_lamports_after

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```




ACCOUNTS

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```


ACCOUNTS


- ▶ Up to 10MB of mutable storage
- ▶ Can only be written by the 'owner' account
- ▶ Can not be resized (currently)

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```



ACCOUNTS


```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```



ACCOUNTS

- ▶ The only account that can write to the data
- ▶ It can only be changed if the data is zero
- ▶ !!! CHECK THE OWNER !!!

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```



ACCOUNTS

- ▶ The only account that can write to the data
- ▶ It can only write if the owner is zero
- ▶ !!! CHECK THE OWNER

CHECK THE OWNER

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    pub data: Vec<RefCell<'a mut u64>>,  
    pub data2: Vec<RefCell<'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
    pub executable: bool,  
  
    // [...]  
}
```


ACCOUNTS

- 
- ▶ The o... can write
to t...
- ▶ It can...
is zero
- ▶ !!! CHECK T...
- CHECK THE OPA**

can write

```
pub struct Account {
    pub key: &'a str,
    pub is_signer: bool,
}
```

OWNER

CHECK THE OWNER!!!!!!1!!

OWNER

OWNER!!!!!!1!!

```
pub struct AccountInfo<'a> {
    pub key: &'a Pubkey,

    pub is_signer: bool,

    pub cell: Cell<&'a mut u64>>,
    pub cell: Cell<&'a mut [u8]>>,
    pub key: Pubkey,
    pub is_signer: bool,
}
```


ACCOUNTS

- ▶ The owner can write to the account
- ▶ It can be created with a balance of zero

CHECK

CHECK THE OWNER!!!!!!1!!

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    pub balance: u64,  
    pub data: RefCell<&'a mut u64>>,  
    pub metadata: RefCell<&'a mut [u8]>>,  
    pub owner: Pubkey,  
    pub is_owner: bool,  
    // ...  
}
```


ACCOUNTS

- ▶ The owner can write to the account
- ▶ It can be created with a balance of zero

CHECK THE OWNER

CHECK

THE

OWNER

!!!

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,
```

```
    pub data: RefCell<&'a mut u64>>,  
    pub data2: RefCell<&'a mut [u8]>>,  
    pub data3: Pubkey,  
    pub data4: bool,
```

```
    pub data5: bool,
```

```
    pub data6: bool,
```

```
    pub data7: bool,
```

```
    // ...  
}
```


ACCOUNTS

- ▶ The owner can write to the account
- ▶ It can be created with a balance of zero

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,
```

OWNER

CHECK

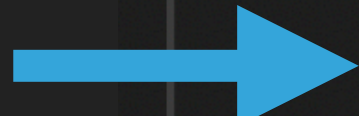
THE

OWNER

!!!1!!!

ACCOUNTS

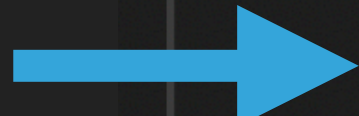
```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```



ACCOUNTS

- Indicates if this account is a smart contract or not

```
pub struct AccountInfo<'a> {  
    pub key: &'a Pubkey,  
  
    pub is_signer: bool,  
  
    // [...]  
  
    pub lamports: Rc<RefCell<&'a mut u64>>,  
  
    pub data: Rc<RefCell<&'a mut [u8]>>,  
  
    pub owner: &'a Pubkey,  
  
    pub executable: bool,  
  
    // [...]  
}
```



TRANSACTIONS

TRANSACTIONS

**CHECK THE OWNER!!!!!!
1!!**

[illegible]

Status: 0k

Status: 0k

Status: 0k

[illegible]

Status: 0k

Status: 0k

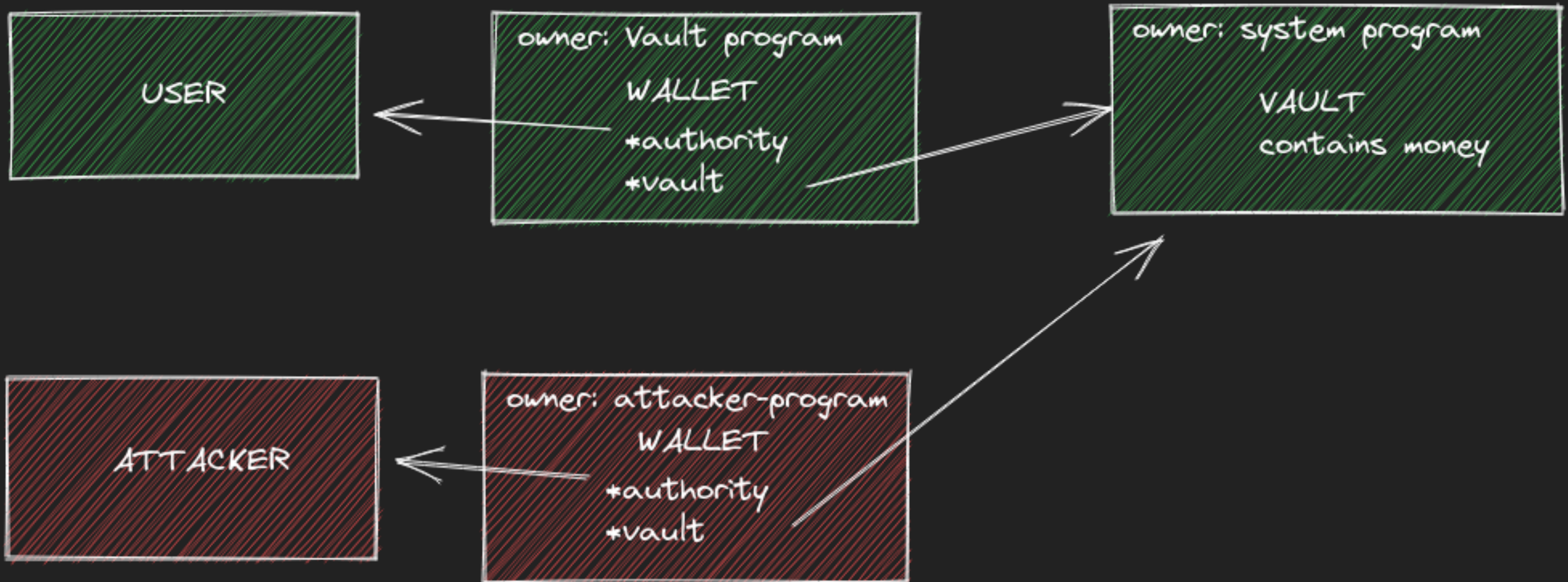
LEVELO

EXPLOIT

EXPLOIT



EXPLOIT



POC FRAMEWORK

Nd

Rust crate: <https://docs.rs/poc-framework/>

GETTING ACCESS

- ▶ Visit: <https://workshop.neodyme.io>
- ▶ We're providing remote environments but you can also use your own :)