Asset rights abstraction

a Case for Smart Contract Wallets

Josef J.  Naim Ashhab
CEO, PWN  Smart contract dev, PWN
The Why
Smart contract wallets beyond multisigs
Smart contract wallets

Proxying contract interactions via another contract (/w extra logic)

- **Pros:**
  - Enable advanced automation/call wrapper
    - (Gnosis) Safe (Multisig)
    - DeFiSaver (external actor/bot to readjust your CDP)
  - Advanced identity/key handling / asset separation
    - ERC725/735 - Decentralized Identity Standards
    - Argent (key - asset separation)

- **Cons:**
  - adds gas overhead
  - may require extra integrations (EIP 1271 etc.)
Why messing up with rights?
Case for an asset rights layer

- Creating hooks into someone-else’s wallet (if they consent)
- Doesn’t require moving actual assets around as much
  - Tackles accounting / KYC
- Self-Custody collateral enables „DeFi Mortgage“
  - Have your cake and eat it too (Use them while they backing a loan)
    - Voting rights, Gaming items/parcels, ENS names
- Asset renting
  - Allow someone else using your tokens w/o losing actual ownership
Technical details
Requirements

● Has to act as a normal contract wallet
  ○ Call arbitrary calladata on any address
    ■ transfer, approve, asset utility, etc.
● Enable tokenizing assets transfer rights (ATR)
  ○ Fungible, non-fungible, and semi-fungible assets
  ○ Enable ATR token holder to transfer asset from owners wallet
  ○ Prevent owner without an ATR token to transfer / burn its assets
    ■ Block transfer / burn calls
    ■ Block approval calls
Wallet design
Wallet design

- (Gnosis) Safe multisig contract wallet
  - Guard
    - checks before and after transaction
  - Module
    - enable to initiate transaction without owners approval

ATR Guard
- enforcing transfer rules
- tracking approved operators

ATR Module
- ATR token (ERC721)
- transferring asset via ATR token
- tracking tokenized balances

Gnosis Safe v1.3.0

Guard → Module → Default fallback handler
Challenges
Challenges

- Approval issue while minting an ATR token
- “Stalking attack”
  - Transferring a malicious asset to victims wallet
- EIP-1271
- Gas overhead
- Non-standard assets
- Not possible to use `delegatecalls`
Approval issues while minting ATR token

- Check that collection hasn’t approved address before minting ATR token
- 4 types of approval
  - ERC20 - approve(amount)
  - ERC721 - approve(id) + setApprovalForAll(address)
  - ERC1155 - setApprovalForAll(address)
- Only ERC721 - approve(id) has a getter function for approved address
-💡 All approve calls are done through wallet
  - -> wallet can track them
“Stalking Attack”

- Put victims wallet into invalid tokenized balance state
  - -> Victim cannot execute any transaction
    - would revert on Insufficient tokenized balance error
- 2 types of transfer
  - Claim
    - to ATR token holder address
  - Transfer
    - to any address, but need recipient permission
- The attack can still be executed, but much harder
  - Functions to recover from this attack
Open challenges

- Enable EIP-1271
  - Wallet needs to “pre-approve” hash
- Gas overhead
  - Minting ATR token has constant overhead
  - Transferring asset via ATR token has linear overhead depending on a number of tokenized assets in a wallet
- Non-standard assets
  - Security issue for ATR token holder when asset defines non-standard transfer or approve function
- Not possible to use *delegatecall*
Thank you!

Josef J  
CEO, PWN  
josef@pwn.xyz  
@JosefJ_  

Naim Ashhab  
Smart contract dev, PWN  
naim@pwn.xyz  
@ashhanai
BEST I CAN DO IS A POAP