Sensitive On Chain Data: Exercise 1 - Mastering CAST

Intro

In this exercise you will learn how to work with `cast` tool.

Installations & Guidelines

Download and install Foundry

```
curl —L https://foundry.paradigm.xyz | bash
foundryup
```

```
For the next tasks, use this Binance RPC url:
https://rpc.ankr.com/bsc

Important: Use ONLY "cast" CLI to solve the following exercises, in your
answer provide the command that you used.
```

Useful Links

[Foundy Installation Guide](#)

[Cast Overview](#)

[Cast Commands](#)

Tasks

Task 1 - Retreiving General Information

1. Retreive the chainID.

```
Chain ID: 56
Command: cast chain-id --rpc-url https://rpc.ankr.com/bsc
```

2. Retreive the last validated block number.

```
Block Number: 26350700
Command: cast block-number --rpc-url https://rpc.ankr.com/bsc
```

Task 2 - Retreiving Transaction and Block Information

1. Get the transaction info for this tx hash
   0x3f6da406747a55797a7f84173cbb243f4fd929d57326fdcfcf8d7ca55b75fe99.

```
Command: cast tx
0x3f6da406747a55797a7f84173cbb243f4fd929d57326fdcfcf8d7ca55b75fe99 --rpc-
url https://rpc.ankr.com/bsc
```

2. Get the block timestamp and the miner address who validated the block of the transaction from the
   previous question.

```
Block Timestamp: 1675173198
Miner Address: 0x2465176C461AfB316ebc773C61fAEe85A6515DAA
Command: cast block 25263862 --rpc-url https://rpc.ankr.com/bsc
```

3. For the same transaction, get the transaction input data and contract address that was called.

```
Input Data:
0x88303dbd0000000000000000000000000000000000000000000000000000000000031c
00000000000000000000000000000000000000000000000000000000400000000000
0000000000000000000000000000000000000000000000010000000000000000000000
000000000000000000000000000000000000000000010974d
Contract: 0x5aF6D33DE2ccEC94efb1bDF8f92Bd58085432d2c
Command: cast tx
0x3f6da406747a55797a7f84173cbb243f4fd929d57326fdcfcf8d7ca55b75fe99 --rpc-
url https://rpc.ankr.com/bsc
```

Task 3 - Transaction Analysis

1. Using the data that you got from the previous question, find the function name and parameters types that was called.

```
Sighash: 88303dbd
Function Name & Params: buyTickets(uint256,uint32[])
Command: cast 4byte 88303dbd
```

2. Decode the input data.

```
Decoded input data:
796
[1087309]
Command: cast 4byte-decode
0x88303dbd0000000000000000000000000000000000000000000000000000000000000031c
0000000000000000000000000000000000000000000000000000000000040000000000
0000000000000000000000000000000000000000000000000010000000000000000000
00000000000000000000000000000000000010974d

Another Option: cast --calldata-decode "buyTickets(uint256,uint32[])"
0x88303dbd0000000000000000000000000000000000000000000000000000000000000031c
0000000000000000000000000000000000000000000000000000000000040000000000
0000000000000000000000000000000000000000000000000010000000000000000000
00000000000000000000000000000000000010974d
```

Task 4 - Smart Contract Storage Analysis

1. Get the previous task's smart contract's bytecodes.

```
Command: cast code 0x5aF6D33DE2ccEC94efb1bDF8f92Bd58085432d2c --rpc-url
https://rpc.ankr.com/bsc
```

2. Get the contract storage slots 0, 1, and 2.

```
Command 1: cast storage 0x5aF6D33DE2ccEC94efb1bDF8f92Bd58085432d2c 0 --
rpc-url https://rpc.ankr.com/bsc
Result 1:
0x0000000000000000000000000000000000000000000000000000000000000001

Command 2: cast storage 0x5aF6D33DE2ccEC94efb1bDF8f92Bd58085432d2c 1 --
rpc-url https://rpc.ankr.com/bsc
Result 2:
0x00000000000000000000000021835332cbdf1b3530fae9f6cd66feb9477dfc02

Command 3: cast storage 0x5aF6D33DE2ccEC94efb1bDF8f92Bd58085432d2c 2 --
```

```
rpc-url https://rpc.ankr.com/bsc
Result 3:
0x0000000000000000000000000000566a7e38b300e903de71389c2b801acdba5268db
```

NOTE: The state of the contract was slightly changed since I solved it and recorded the video, so you might see a bit different results.

1. Get the smart contract source-code FROM HERE, and copy it to a new file in VSCODE.

Explain which are the state variables that match storage values that you found in the previous question?

```
Slot 0: The `_status` variable from the `ReentrancyGuard` contract that is
inherited by the `PancakeSwapLottery` contract, 1 is the default state
(`NOT_ENTERED`)

Slot 1: The `_owner` variable form the `Ownable` contract that is
inherited by the `PancakeSwapLottery` contract,
`0x21835332cbdf1b3530fae9f6cd66feb9477dfc02` is the contract owner
address.

Slot 2: The `injectorAddress` from the `PancakeSwapLottery` contract
(`0x566a7e38b300e903de71389c2b801acdba5268db`).
```