Slovenská Technická Univerzita v Bratislave Fakulta informatiky a informačných technológií

Digitálne meny a Blockchain

Zadanie 2 – Smart kontraktový systém pre námorné bitky

Písané otázky

- ***Predpokladajme, že hráč 1 umiestni na hraciu plochu menej ako 10 lodí, ale nikdy neklame o zásahoch alebo minutiach. Môže hráč 2 dostať svoje peniaze späť? Prečo áno, prečo nie?***
  Pokiaľ hráč 1 zničí všetky lode hráča 2, tak neprejde kontrolou pri vyplácaní výhry. Takže hráč 2 má možnosť postupne zničiť všetky lode hráča 1 a dostať sa tak k výhre.

- ***Prečo neobmedzujeme hráčov v umiestňovaní viac ako 10 lodí na ich dosku?***
  Bolo by to pre hráča ťažšie vyhrať. Ale pokiaľ by hráč zničil viac ako 10 lodí, tak by neprešiel kontrolou pri vyplácaní výhry.

- ***Nemáme mechanizmus, ktorý by hráča obviňoval z umiestnenia menej ako 10 lodí na hraciu plochu. Ako by ste ho vedeli implementovať?***
  Implementoval by som ho ako obmedzenie pred začatím samotnej hry, kedy by som vedel skontrolovať presný požadovaný počet lodí na umiestnenie.

- ***Napadajú vám scenáre útoku alebo konkrétne zraniteľné miesta v niektorom z uvedených kódov, proti ktorým by ste sa nedokázali ubrániť?***
  V kóde nie je ošetrený brute force útok na rýchle vyklikanie všetkých políčok. Takže v prípade automatizovaného útoku je možné hru vyhrať ešte pred tým, ako druhý hráč spraví nejaký ťah.

Popis doimplementovaných častí kódu v contact_starter.sol

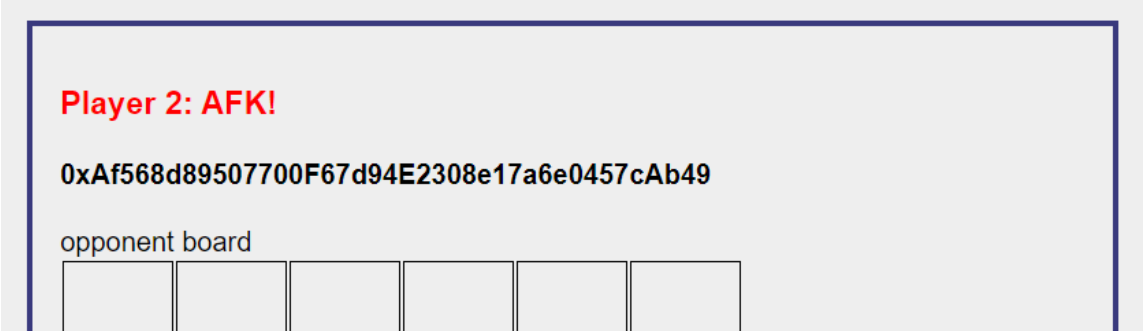
Hra využíva 2 stavy (rozohraná/ukončená) reprezentované typom **enum GameStates { RUNNING a FINISHED }**

Kontrakt má ďalej zadefinovaný jeden event **AFK**, ktorý slúži na oznámenie a zistenie, či je hráč prítomný v hre.

- **store_bid()** – uloženie adries a stávok jednotlivých hráčov
- **clear_state()** – vynulovanie stavu hry
- **store_board_commitment()** – uloženie "merkle rootu" pre konkrétneho hráča
- **check_one_ship()** – overenie, či hráč v danom kroku nepodvádzal + kontrola zásahu protihráčovej lode
- **claim_win()** – vyplatenie výhry hráčovi, ktorý splnil všetky podmienky na výhru (trafil všetky nepriateľské lode a nepodvádzal)
- **forfeit()** – vzdanie hry a vyplatenie výhry protihráčovi
- **accuse_cheating()** – obvinenie a následná kontrola, či bol posledný protihráčov ťah poctivý, resp. či neklamal
- **claim_opponent_left()** – obvinenie protihráča z neprítomnosti a následné spustenie časovača, do ktorého uplynutia musí protihráč zareagovať.
- **handle_timeout()** – reakcia na obvinenie z neprítomnosti, ktorá zruší obvinenie a zastaví časovač
- **claim_timeout_winnings()** – ak hráč obvinený z neprítomnosti nezareagoval do uplynutia časovača, tak výhru získa jeho protihráč
- **is_game_over()** – overenie, či je hra ukončená

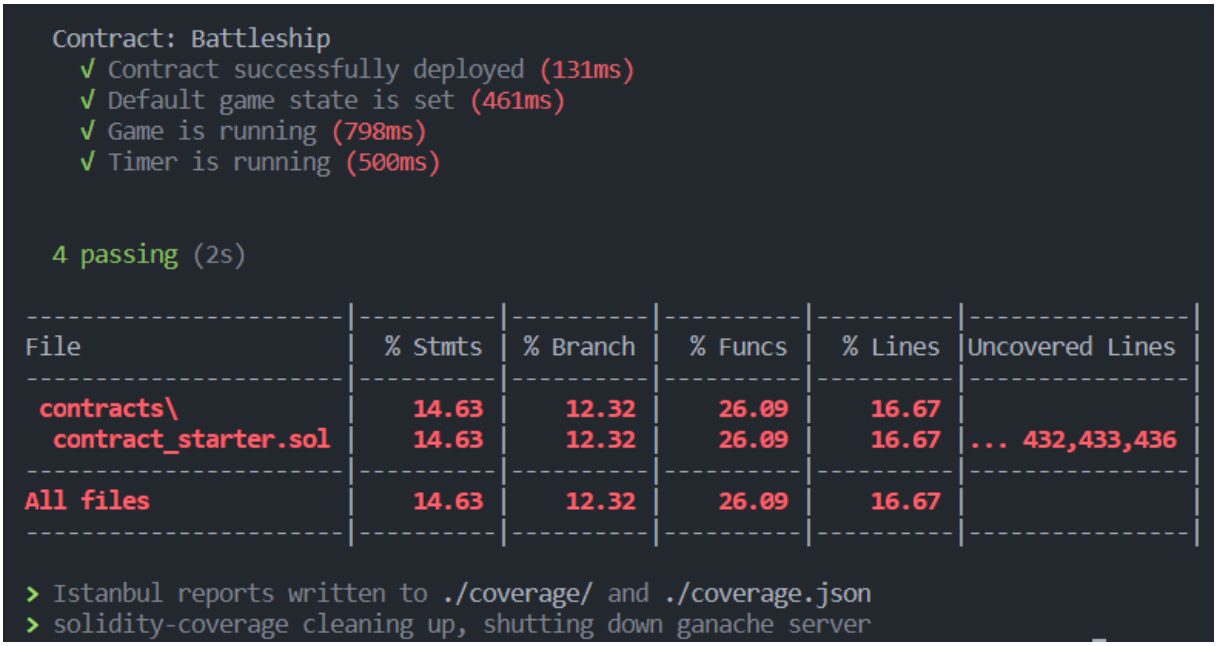
Doplnková funkcionalita

- Upozornenie hráča na obvinenie z neprítomnosti v podobe zmeny textu a farby



Testovanie

Testovanie prebehlo pomocou solidity-coverage https://github.com/sc-forks/solidity-coverage



Všeobecné informácie

- Vývojové prostredie
  - *Visual Studio Code, Remix IDE*
- Programovací jazyk
  - *JavaScript, Solidity*