UMCS

UNIWERSYTET MARII CURIE-SKŁODOWSKIEJ
W LUBLINIE
Wydział Matematyki, Fizyki i Informatyki

Kierunek: Informatyka

**Piotr Jasina**
nr albumu: 279183

# Identyfikacja inteligentnych kontraktów w sieci Ethereum

Ethereum smart contracts identification

Praca licencjacka
napisana w Zakładzie Cyberbezpieczeństwa
pod kierunkiem dr. Damiana Rusinka

**Lublin rok 2019**

# Spis treści

# Wstęp

...

# Rozdział 1

# Ethereum

## 1.1 Historia

Literatura: [2, 1]. TODO

## 1.2 Opis platformy

Literatura: [2, 1]. TODO

## 1.3 Ethereum Virtual Machine

Literatura: [2, 1]. TODO

## 1.4 Inteligentne kontrakty

Literatura: [2, 1]. TODO

# Rozdział 2

# Solidity

## 2.1 Sygnatura funkcji

Literatura: [2, 1]. TODO

## 2.2 Selektor funkcji

Literatura: [2, 1]. TODO

## 2.3 Generowanie akcesorów podczas kompilacji

Literatura: [2, 1]. TODO

# Rozdział 3

# Projekt Aplikacji

Celem mojej pracy licencjackiej było utworzenie aplikacji internetowej umożliwiającej identyfikacje inteligentnych kontraktów wykorzystywanych w sieci Ethereum. Dzięki aplikacji użytkownik po wprowadzeniu na stronie kodu bajtowego kontraktu jest w stanie otrzymać najbardziej prawdopodobną implementacje kontraktu napisana w języku Solidity bazując na bazie danych aplikacji.

Poniżej zostało opisane działanie aplikacji wraz ze szczegółowym opisem funkcjonalności, architektury oraz wykorzystanych technologi.

## 3.1 Funkcjonalność

Po wejściu na stronę główną aplikacji użytkownik zobaczy w górnej części menu, w którym ma do wyboru: identyfikację inteligentnych kontraktu, wprowadzanie plików źródłowych kontraktów do aplikacji oraz dokumentacje API aplikacji. Na stronie głównej poniżej menu znajduje się opis aplikacji wraz z aktualna liczba kodów źródłowych znajdujących się w bazie danych aplikacji.

### 3.1.1 Identyfikacja inteligentnych kontraktów

Pierwsza opcja dostępną w menu jest identyfikacja inteligentnych kontraktów. Po naciśnięciu przycisku na menu, użytkownik zostanie przekierowany na podstronę na której ma możliwość wprowadzenia kodu bajtowego w systemie szesnastkowym.

Podczas wprowadzania kodu istnieje możliwość wprowadzenia kodu z prefixem "0x" oraz bez tego prefixu. Jeśli użytkownik poda kod z prefixem to aplikacja podczas przetwarzania kodu zignoruje niepotrzebne znaki. Takie rozwiązanie zostało zastosowane w celu zapewnienia użytkownikowi większej wygody oraz komfortu w korzystaniu z aplikacji.

Pomyślne wprowadzone dane wykorzystywane do identyfikacji zatwierdzamy przyciskiem "Submit", a następnie po stronie serwerowej aplikacji rozpoczynany jest proces analizy wprowadzonego kodu bajtowego oraz wyszukiwane są najbardziej prawdopodobne implementacje. W rezultacie —jak widzimy na rysunku TUTAJ BEDZIE ZDJECIE :D— utrzymujemy listę wyszukanych implementacji posortowanych malejąco według współczynnika dopasowania. Jeśli klikniemy przyciskiem na jedną z wyświetlonych pozycji to w nowej karcie przeglądarki otworzy się podstrona z implementacją kontraktu wraz z podświetleniem składni języka Solidity.

### 3.1.2 Wprowadzanie kodu źródłowego kontraktu do aplikacji

Kolejna opcją dostępną dla użytkownika jest możliwość dodania własnego kodu źródłowego kontraktu napisanego w języku Solidity. Opcja ta umożliwia użytkownikom wsparcie aktualnej bazy danych o kolejne kody źródłowe inteligentnych kontraktów, w wyniku takiego działania wszyscy pozostali użytkownicy

maja większa szanse na precyzyjną identyfikacje kontraktu. Jak widać na rysunku –TUTAJ RYSUNEK–, ze względu na wygodę użytkowników korzystających z aplikacji, zostały utworzone dwie możliwości wprowadzania kodów źródłowych.

Pierwsza opcja umożliwia wprowadzenie lokalnego pliku zawierającego kod źródłowy z dysku komputera za przeglądarki internetowej.

Druga możliwością jest wklejenie kodu źródłowego bezpośrednio do pola tekstowego. Druga opcja została utworzona ponieważ, podczas korzystania z aplikacji użytkownik może bezpośrednio skopiować kod źródłowy, który jest w dowolnym innym źródle tekstowym i wkleić go bezpośrednio do mojej aplikacji bez konieczności tworzenia pliku tymczasowego.

### 3.1.3 Interfejs programistyczny aplikacji

## 3.2 Architektura

Literatura: [2, 1]. TODO

### 3.2.1 Wyszukiwanie sygnatur funkcji w kodzie źródłowym

Literatura: [2, 1]. TODO

### 3.2.2 Wyszukiwanie selektorów funkcji w kodzie bajtowym

Literatura: [2, 1]. TODO

### 3.2.3 Szukanie implementacji na podstawie kodu bajtowego

Literatura: [2, 1]. TODO

## 3.3 Wykorzystane technologie

Literatura: [2, 1]. TODO

# Bibliografia

[1] Bibliografia 1. `http://www.google.com`.

[2] Bibliografia 2. *Nazwa*.

# Spis tabel

# Spis rysunków

# Spis listingów