ДЗ “Проверка СмК”

Форма: код

Область: Ethereum

Цель работы: получить навыки проверки правильности исполнения кода смарт-контрактов

Результат: написаны тесты, код в github

Описание

Основы безопасности

Смарт-контракты работают с чувствительными данными, например с деньгами, и если логика обработки имеет уязвимость, то либо ценность данных снижается или эти данные утекают. Так как информацию в блокчейне невозможно изменить в одностороннем порядке, то и требования к правильности обработки данных повышаются. Самые “знаковые” события - взлом The DAO, вывод 600 млн из моста (август 2021).

Для повышения качества и надёжности работы смарт-контрактов применяются различные методики, основная из которых - тестирование.
Для тестирования смарт-контрактов применяются: unit тестирование (проверка отдельных функций), интеграционное тестирование (проверка правильности взаимодействия контрактов), формальная верификация (область активных академических исследований).

Многие угрозы безопасности проистекают из ошибок или недоработок при разработке языка описания бизнес-логики смарт-контрактов -- языка Solidity. Самая известная из которых - небезопасные математические операции связанные с переполнением целочисленного типа.

Индустрия встречала множество угроз связанных с “особенностями” языком, некоторые угрозы перерастали в атаки, но многие угрозы были решены в свежих релизах языка Solidity. Поэтому важно постоянно актуализировать версию компилятора и проверять документацию на предмет раскрытия новых особенностей языка. Список известных угроз вы найдете в Ссылках на полезные источники

Unit и интеграционное тестирование

Для написания тестов смарт-контрактов чаще всего применяются разработанные для этих целей инструменты, например набор инструментов Truffle (Waffle), или HardHat.
Чаще всего, тестирование смарт-контрактов производится в локальной EVM, которая запускается или с помощью выбранного инструмента или с помощью специализированных инструментов, например Ganache. После проведения всех испытаний, смарт-контракт проверяют на правильность работы в одной из тестовых сетей.

Формальная верификация

Формальная верификация - это методология проверки правильности работы смарт-контрактов согласно описанной ранее формальной спецификации, формальная спецификация пишется на специализированном языке описания спецификаций (пример такого языка Isabelle, Coq). Формальная верификация - это использование математических инструментов для создания и проверке доказательств корректности работы. Один из лучших способов подтверждения правильности работы смарт-контрактов.

Задание 1 (8 баллов)

1. Разработать unit тесты для смарт-контракта “Камень-ножницы-бумага”
2. Написать интеграционные тесты на взаимодействие смарт-контрактов

Задание 2 (10 баллов)

1. Раскрыть в Пояснительной записке актуальное состояние индустрии в области создания формальных спецификаций и проведения формальных проверок смарт-контрактов.

Условие

отсутствуют

Контрольные вопросы

1. Почему на этапе разработки используется локальная EVM, а не рабочая?
2. С чем связана сложность проведения формальных проверок смарт-контрактов?
3. Как передать eth при вызовах call и delegatecall?
4. В чём заключается суть Re-Entrancy атаки?

Полезные ссылки

1. Solidity Security: Comprehensive list of known attack vectors and common anti-patterns
2. DelegateCall: Calling Another Contract Function in Solidity | by zeroFruit | Coinmonks
3. Contract ABI Specification — Solidity 0.8.10 documentation
4. Ethereum Development Testing
5. Security Considerations — Solidity 0.8.10 documentation
6. Truffle | Testing Your Contracts | Documentation
7. Waffle
8. Ethereum development environment for professionals by Nomic Labs
9. Standing the Time of Test with Truffle and Ganache
10. Isabelle
11. The Coq Proof Assistant: Welcome!
12. SMTChecker and Formal Verification
13. Delegatecall