МЕТОДИКА АНАЛИЗА УЯЗВИМОСТЕЙ И ОПРЕДЕЛЕНИЯ УРОВНЯ БЕЗОПАСНОСТИ СМАРТ-КОНТРАКТОВ ПРИ РАЗМЕЩЕНИИ В СИСТЕМАХ РАСПРЕДЕЛЕННЫХ РЕЕСТРОВ

Кривоногов А.А.[1], Репин М.М.[2], Федоров Н.В.[3]

Аннотация.

С каждым годом технология использования смарт-контрактов привлекает все больше внимания со стороны пользователей за счет уникальных преимуществ, которыми она обладает: автоматическое выполнение транзакций отслеживаемым и неизменным способом без авторизации третьей стороной. В то же время смарт-контракт является одним из наиболее уязвимых элементов в системах распределенных реестров, который может быть подвержен атаке со стороны злоумышленников.

Целью исследования *является разработка методики, позволяющая анализировать смарт-контракты на уязвимости информационной безопасности и определять его уровень безопасности перед размещением в системах распределенных реестров.*

Методы исследования: *для достижения поставленной цели были исследованы методы статического и динамического анализа, выявлены наиболее актуальные уязвимости информационной безопасности, а также определены параметры для расчета коэффициента критичности уязвимости и уровня безопасности смарт-контракта.*

Полученный результат: *предложена перспективная статодинамическая методика анализа уязвимостей смарт-контракта, позволяющая однозначно определять уровень безопасности смарт-контракта перед его размещением в системе распределенных реестров. Заданы её основные параметры, а также определены эталонные коэффициенты безопасности смарт-контракта. Описан полный алгоритм работы статодинамической методики анализа смарт-контракта, а также приведен пример сгенерированного документарного отчета безопасности по результатам анализа смарт-контракта.*

Ключевые слова: *статический анализ, динамический анализ, автоматизированные инструменты, информационная безопасность, критичность уязвимости, тестирование, аудит кода.*

DOI: 10.21681/2311-3456-2020-04-56-65

Введение

С каждым днем смарт-контракты (СК) становятся все более популярным механизмом, позволяющим автоматизировать многие ключевые задачи в разных отраслях. Помимо этого, СК пользуются большой популярностью и у злоумышленников, которые пытаются выявить уязвимость в коде СК и, проэксплуатировав ее, получить доступ к финансовым активам [6]. Одной из причин актуальности подобных атак является отсутствие единого набора инструментов для разработки, рекомендаций и руководств по безопасной разработке кода СК, а также шаблонов проектирования безопасных СК. Это в свою очередь может приводить к возникновению уязвимостей в коде или логике СК, некоторые из которых могут быть использованы при реализации атак на СК или на систему распределенных реестров (РР) в целом [12, 14].

На текущий момент существует большое количество инструментов статического и динамического анализа, в основе которых применяются разные методы для поиска и определения уязвимых конструкций [7].

Однако, единая методика анализа, объединяющая в себе инструменты разного типа анализа, при помощи которой можно бы было оценить безопасность СК, в настоящее время отсутствует.

Рассмотрим существующие методы анализа с целью определения особенностей, преимуществ и недостатков при исследовании кода СК.

Методы анализа уязвимостей смарт-контрактов

Как правило, разработка приложения состоит из большого количества тестов, в процессе которых осуществляется поиск и исправление следующих ошибок [9]:

1 Кривоногов Антон Алексеевич, аспирант кафедры Информационной безопасности Московского Политехнического университета, г. Москва, Россия. E-mail: aakrivonogov97@yandex.ru

2 Репин Максим Михайлович, руководитель направления Научно-исследовательского института «Восход», г. Москва, Россия. E-mail: bmstu.iu8@gmail.com

3 Федоров Николай Владимирович, кандидат технических наук, доцент, заведующий кафедрой Информационной безопасности Московского Политехнического университета, г. Москва, Россия. E-mail: fedorovnv31@mail.ru

- Логические ошибки;
- Бизнес-ошибки;
- Уязвимости информационной безопасности (ИБ).

Существует несколько подходов к аудиту кода СК.

Первый подход – это ручное рассмотрение бизнес-логики и логики разработки СК командой экспертов, которые изучают структуру и исходный код СК с целью выявления возможных уязвимостей.

Второй подход – это использование инструментов автоматизированного анализа, которые выполняют проверку СК и осуществляют поиск уязвимостей ИБ.

Анализ уязвимостей можно разделить на два основных типа:

- **Статический анализ уязвимостей:** выявляет уязвимости перед выполнением СК;
- **Динамический анализ уязвимостей:** выявляет уязвимости во время выполнения СК.

Статические анализ уязвимостей

Статический анализ уязвимостей в СК представляет собой метод отладки СК, который осуществляется путем проверки кода без его фактического выполнения. При статическом анализе осуществляется рассмотрение всех возможных вариантов исполнения кода, а также определяются уязвимые шаблоны и недостатки, которые можно ожидать во время выполнения СК [2].

Необходимо выделить следующие **преимущества** статического анализа:

- Обеспечение хорошего покрытия анализируемого СК и возможность охватывать все возможные пути исполнения;
- Обнаружение проблем с кодом, прежде чем он будет готов к интеграции и дальнейшему динамическому анализу;
- Отслеживание уязвимостей до точного местоположения в коде СК.

Несмотря на преимущества, статический анализ уязвимостей содержит следующие **недостатки:**

- Невоспроизводимость результатов: информация о том, как вызвать обнаруженные уязвимости, обычно не фиксируется, поэтому, возможно, потребуется проверить уязвимости, чтобы избежать ложных срабатываний;
- Ограниченный диапазон уязвимостей: возможность проверки только известных атак или предопределенных правил.

Наиболее популярными и эффективными инструментами статического анализа СК являются SmartCheck, Securify, Oyente.

Динамический анализ уязвимостей

Динамический анализ – это метод, который осуществляет проверку СК во время его выполнения [10].

Методики динамического анализа СК подразделяются на две основные категории [15]:

- **Конкретное выполнение:** включает в себя выполнение СК в обычном режиме против тщательно созданных тестовых примеров, предоставленных пользователем. Распространенной практикой конкретного выполнения является фаззинг, когда некорректный ввод предоставляется СК в попытке обнаружить уязвимость.
- **Символическое выполнение**: осуществляет выполнение СК в эмулируемой среде, используя символические переменные и отслеживая состояние СК во время его выполнения.

Динамический анализ имеет следующие **преимущества:**

- Выявленные уязвимости являются в высокой степени воспроизводимыми, поскольку присутствуют входные данные или условия пути, необходимые для их запуска;
- Возможность анализа СК, в которых нет доступа к реальному коду.
- Также у динамического анализа присутствуют и некоторые **недостатки:**
- Использование значительных вычислительных ресурсов при проведении анализа;
- Только один путь выполнения может быть проверен в каждый конкретный момент времени, что требует большого количества тестовых запусков для большей полноты тестирования кода СК;
- Фаззеры имеют ограниченную семантическую информацию о том, какие части входных данных вызвали уязвимость. Это затрудняет отслеживание уязвимости в точном месте в коде СК;
- При символическом выполнении количество условных путей вызывает экспоненциальное увеличение возможных исследуемых путей, ограничивая сложность СК, которые можно проанализировать.

Из наиболее популярных и эффективных инструментов для динамического анализа СК можно выделить следующие: Mythril, MAIAN, Manticore.

Статодинамическая методика анализа смарт-контрактов

Для разработчика безопасность должна являться главным приоритетом при разработке СК. Придерживаясь правил безопасной разработки, начиная с ранней стадии проектирования, можно избежать появления критических ошибок на всех последующих этапах жизненного цикла СК. Для обеспечения безопасности, прежде всего, необходимо наиболее полно подойти к анализу и тестированию СК, чтобы выявить все векторы атак [3, 5].

Анализ и тестирование СК осуществляется с целью выявления уязвимостей ИБ на уровне кода, которые прямо или косвенно могут влиять на общий уровень безопасности и устойчивости блокчейн-сети [1, 13].

Для большего охвата кода целесообразно использовать методы статического и динамического анализа, а также инструменты автоматизированного анализа, которые базируются на данных методах. Таким образом, для совместного использования инструментов разных типов анализа, а также для корректного оценивания уровня безопасности СК, была предложена статодинамическая методика анализа уязвимостей СК.

На первом этапе оценки СК необходимо определить перечень наиболее критичных и часто встречающихся

на сегодняшний день уязвимостей, эксплуатация которых может привести к нарушению логики работы СК.

Коэффициент критичности – это величина, которая определяет опасность данной уязвимости при ее наличии в коде СК.

Расчет коэффициента критичности осуществляется на основе следующих параметров:

- **возможность реализации** уязвимости ИБ – P_{TH}: может принимать значения от 1 до 3 в зависимости от того, насколько вероятной является реализация данной уязвимости, т.е. от отсутствия объективных предпосылок, статистики и мотивации для реализации уязвимости до существования объективных предпосылок, достоверной статистики и наличия мотивации у злоумышленника;
- **степень возможного ущерба от реализации** уязвимости ИБ – E_{DMG}: данный показатель рассчитывается как сумма ущерба на СК (SC_{DMG}), инфраструктуру (I_{DMG}) и активы (A_{DMG}) по следующей формуле:

$$E_{DMG} = SC_{DMG} + I_{DMG} + A_{DMG} \quad (1)$$

Каждый из показателей может принимать значение от 1 до 10 в зависимости от критичности ущерба от эксплуатации уязвимости;

- **актуальность реализации** уязвимости ИБ – R_{TH}: может принимать значения от 0 до 1 в зависимости от актуальности эксплуатации уязвимости и получении дальнейшей выгоды на сегодняшний день;
- **трудность эксплуатации** уязвимости ИБ – D_{EXP}: данный показатель рассчитывается по следующей формуле:

$$D_{EXP} = (EXP + H) * T \quad (2)$$

Каждый показатель может варьироваться от 1 до 3 в зависимости от трудности эксплуатации (EXP) уязвимости, количества используемых ресурсов (H) для успешной реализации уязвимости, а также времени (T).

Определив все показатели, необходимо перейти к расчету коэффициента критичности уязвимости по следующей формуле:

$$C_{SC} = \frac{(P_{TH} + E_{DMG}) * R_{TH}}{D_{EXP}} \quad (3)$$

На основе вышеперечисленных параметров осуществляется подсчет коэффициентов критичности для уязвимостей ИБ. Подробное описание каждой уязвимости, коэффициент критичности, а также способы устранения и смягчения последствий представлены в табл. 1 [4, 8].

По результатам определения коэффициента критичности каждой из уязвимостей ИБ, далее следует перейти к исследованию СК, используя инструменты статического и динамического анализа. В качестве инструментов статического анализа можно использовать инструменты SmartCheck [2] и Securify [9], а для динамического анализа – инструмент MAIAN [10].

Осуществив анализ СК инструментами автоматизированного анализа и обнаружив потенциальные уязвимости, далее необходимо определить коэффициенты критичности каждой найденной уязвимости ИБ в соответствии с табл. 1, а затем перейти к расчету коэффициента безопасности целевого СК.

Расчет коэффициента безопасности СК осуществляется на основе следующих параметров:

- L_{SC} – коэффициент количества строк в СК. Принимает значение в соответствии с табл. 2;

Таблица 2

Значения количества строк в коде СК

Количество строк в коде	Значение
1-20	1
21-40	2
41-60	3
61-100	4
101 и более	5

- F_{SC} – коэффициент количества функций в СК. Принимает значение в соответствии с табл. 3;
-

Таблица 3

Значения количества функций в коде СК

Количество функций в коде	Значение
1-3	1
4-6	2
7 и более	3

- C_{SC} – коэффициент критичности обнаруженной уязвимости ИБ в СК. Данный показатель рассчитывается заранее по каждой уязвимости и берется из табл. 1;
- V_{PR} – коэффициент наличия уязвимости ИБ в коде СК. Может принимать значения 0 или 1 в зависимости от того, присутствует ли уязвимость из табл. 1 в коде СК. Значение параметра определяется после исследования СК статическими и динамическим инструментами анализа.

Коэффициент безопасности СК определяется по следующей формуле:

$$SL_{SC} = \frac{F_{SC}}{L_{SC}} * \sum_{i=1}^{n} C_{SC(i)} * V_{PR(i)} \quad (4)$$

, где n – число уязвимостей ИБ в коде СК.

DOI:10.21681/2311-3456-2020-04-56-65

Таблица 1

Сводная таблица коэффициентов критичности наиболее актуальных уязвимостей ИБ в СК

№	SWC	Название уязвимости	Краткое описание	Оценка критичности	CWE	Мера смягчения
1	107	Reentrancy (повторный вход)	Одна из главных опасностей при вызове внешних контрактов заключается в том, что они могут взять на себя управление потоком. При атаке с повторным входом (например, атака рекурсивного вызова) СК злоумышленника осуществляет обратный вызов вызывающего СК до того, как первый вызов функции будет завершен. Это может привести к нежелательному взаимодействию различных вызовов функции и дальнейшей потере криптоактивов.	3	CWE-841: ненадлежащее соблюдение поведенческого рабочего процесса	Использовать transfer() вместо contract.call() для передачи Ether на ненадежные адреса. При использовании низкоуровневых вызовов необходимо убедиться, что все внутренние изменения состояния исполнены до выполнения вызова.
2	104	Unchecked call (непроверенное возвращаемое значение вызова)	Возвращаемое значение вызова (call) не проверяется. Выполнение возобновится, даже если вызываемый СК создает исключение. Если вызов дает сбой случайно или злоумышленник вынуждает вызов дать сбой, это может привести к непредвиденному поведению в последующей логике работы СК.	2,63	CWE-252: непроверенное возвращаемое значение	При использовании низкоуровневых методов вызова (call) необходимо убедиться, что также происходит обработка возможности сбоя вызова. Необходимо проверять возвращаемое значение.
3	116	Timestamp Dependence (зависимость от метки времени)	СК часто требуется доступ к текущей временной метке, чтобы инициировать зависящие от времени события. Поскольку Ether децентрализован, узлы могут синхронизировать время только до некоторой степени. Более того, злоумышленники могут изменять временную метку своих блоков, особенно если они могут получить преимущества от этого. В связи с этим, разработчики не могут полагаться на точность предоставленной метки времени.	2,1	CWE-829: включение функциональности из ненадежной сферы управления	Рекомендуется разрабатывать СК с учетом того, что временная метка блока и реальная временная метка могут варьироваться до полуминуты. Кроме того, рекомендуется использовать номер блока или внешний источник метки времени через оракулов.
4	115	Tx.origin (авторизация через tx.origin)	Tx.origin является глобальной переменной в Solidity, которая возвращает адрес учетной записи, которая отправила транзакцию. Использование этой переменной для авторизации может сделать СК уязвимым, если авторизованная учетная запись осуществит вызов вредоносного СК. Вызов может быть сделан к уязвимому СК, который проходит проверку авторизации, поскольку tx.origin возвращает первоначального отправителя транзакции, который в этом случае является авторизованным аккаунтом.	3,6	CWE-477: использование устаревшей функции	Рекомендуется не использовать переменную tx.origin для авторизации. Вместо этого необходимо использовать переменную msg.sender. Помимо этого, следует использовать address.transfer() вместо address.call.value(amount)().

№	SWC	Название уязвимости	Краткое описание	Оценка критичности	CWE	Мера смягчения
5	101	Integer Overflow / Underflow (целочисленное переполнение / опустошение)	Целочисленное переполнение / опустошение происходит, когда арифметическая операция пытается создать числовое значение, выходящее за пределы диапазона, который может быть представлен заданным числом битов – либо больше максимального, либо меньше минимального представимого значения.	2,6	CWE-682: неверный расчет	Рекомендуется использовать проверенные безопасные математические библиотеки для арифметических операций, такие как SafeMath.
6	100	Visibility (модификатор доступа по умолчанию)	Функции, для которых не указан модификатор доступа, по умолчанию являются public. Это может привести к уязвимости, если разработчик забыл установить модификатор, а злоумышленник может внести изменения состояния СК.	4	CWE-710: ненадлежащее соблюдение стандартов кодирования	Функции могут быть определены как external, public, internal или private. Рекомендуется явно определять тип модификатора доступа для функции. Это может значительно уменьшить поверхность атаки.
7	114	Transaction Order Dependence (зависимость порядка транзакций)	Майнеры просматривают транзакции в списке неподтвержденных транзакций (pending pool) и выбирают, какие транзакции включить в блок, основываясь на том, кто заплатил достаточно высокую цену (gas) для включения. Поскольку каждый может просматривать список, у злоумышленника есть время на то, чтобы проанализировать и сформировать собственную транзакцию, которая учитывает уже полученную информацию.	2,3	CWE-362: одновременное выполнение с использованием общего ресурса с неправильной синхронизацией («состояние гонки»)	Рекомендуется использовать хеширование для защиты от данного вида атаки. Т.е. сначала необходимо отправлять хеш от передаваемых данных, а после того, как первая транзакция будет подтверждена, необходимо отправлять вторую транзакцию, которая будет содержать в себе сами данные. Если хеш совпадает со СК, то пользователь получает вознаграждение.
8	105	Prodigal contract (расточительный контракт)	Из-за отсутствия или недостаточного контроля доступа злоумышленники могут вывести Ether из учетной записи СК. Эта ошибка также вызывается непреднамеренным раскрытием функций инициализации.	3,75	CWE-284: неправильный контроль доступа	Внедрить меры контроля, чтобы снятие активов могло быть инициировано только уполномоченными сторонами или в соответствии со спецификациями системы СК.
9	106	Self-destruct (самоуничтожение)	Из-за отсутствия или недостаточного контроля доступа злоумышленники могут самостоятельно уничтожить СК, осуществив вызов функции SELFDESTRUCT.	3,83	CWE-284: неправильный контроль доступа	Рекомендуется не использовать функцию самоуничтожения, если она не является абсолютно необходимой. При этом, если все же необходимо использовать данную функцию, то рекомендуется реализовать многозначную схему, чтобы несколько сторон могли одобрить действие самоуничтожения.

После вычисления коэффициента безопасности, необходимо сопоставить полученное значение с эталонными значениями из табл. 4 и определить уровень безопасности СК.

Таблица 4

Эталонная таблица уровней безопасности СК

Коэффициент безопасности	Уровень безопасности
0-2	Высокий уровень
2,1-6	Средний уровень
6,1-10	Низкий уровень
10,1 и более	Критический уровень

Определив уровень безопасности СК, разработчик должен сделать вывод в соответствии с табл. 5, чтобы принять решение относительно публикации СК в системах РР.

дики анализа исходного кода СК, поиска уязвимых конструкций и определения уровня безопасности осуществляется последующему сценарию:

1. Подготовить исходный код Solidity к анализу;
2. Осуществить статический анализ исследуемого кода при помощи инструмента SmartCheck и Securify на перечень уязвимостей ИБ (табл. 1);
3. Проверить каждую уязвимость ИБ из списка на присутствие в коде СК;

3.1. Если уязвимость в коде не обнаружена:

3.1.1. Присвоить уязвимости оценку 0.

3.2. Если уязвимость присутствует в коде:

3.2.1. Присвоить уязвимости оценку 1.

4. После проверки СК статическими инструментами анализа следует преобразовать высокоуровневый код Solidity в байт-код EVM при помощи компилятора Solc;
5. Запустить тестовую приватную блокчейн-сеть Go-Ethereum для выполнения динамического анализа СК;

Таблица 5

Описания уровней безопасности СК

Уровень безопасности СК	Описание
Высокий уровень	Возможность эксплуатации СК злоумышленником с высоким уровнем безопасности сводится к нулю. Неточности, которые могут быть обнаружены в процессе тестирования, не представляют угрозы для работоспособности СК.
Средний уровень	Риск эксплуатации СК со средним уровнем безопасности относительно невелик, однако данный СК не может быть размещен в системе РР. Уязвимости, выявленные в ходе тестирования СК, не являются критическими.
Низкий уровень	СК с низким уровнем безопасности ставит под угрозу конфиденциальность информации отдельных пользователей. При этом эксплуатация уязвимостей может нанести ущерб репутации клиента или привести к умеренным финансовым последствиям.
Критический уровень	СК с критическим уровнем безопасности ставит под угрозу конфиденциальность информации пользователей. При этом, эксплуатация уязвимостей может привести к катастрофическим последствиям для репутации клиента или серьезным финансовым последствиям для пользователей.

По результатам определения уровня безопасности СК следует сгенерировать детальный документарный отчет по анализу кода целевого СК, в котором должна быть представлена следующая информация:

- Уровень безопасности СК в соответствии с табл. 5;
- Список обнаруженных уязвимостей ИБ в исходном СК;
- Подробное описание уязвимостей, обнаруженных в исходном коде СК;
- Конкретное расположение уязвимостей в коде СК;
- Возможные меры по устранению найденных уязвимостей.

Пример потенциального отчета безопасности СК изображен на рис. 1.

Полный алгоритм работы статодинамической мето-

6. Осуществить динамический анализ исследуемого кода при помощи инструмента MAIAN на перечень уязвимостей ИБ (табл. 1);
7. Проверить каждую уязвимость из списка на присутствие в коде СК;

7.1. Если уязвимость в коде не обнаружена:

7.1.1. Присвоить уязвимости оценку 0.

7.2. Если уязвимость присутствует в коде:

7.2.1. Присвоить уязвимости оценку 1.

8. Определить коэффициент критичности найденных актуальных уязвимостей ИБ (табл. 1);
9. Определить коэффициенты количества строк (табл. 2) и функций (табл. 3) в коде исследуемого СК;
10. Осуществить по формуле расчет коэффициента безопасности СК, опираясь на коэффициент критичности, коэффициент наличия уязвимостей, а также на количество строк и функций в коде иссле-

дуемого СК, значения которых были определены в ходе анализа;

11. Сравнить полученный результат с эталонными значениями (табл. 4);

12. Сгенерировать документарный отчет и сделать вывод относительно уровня безопасности СК (табл. 5);

12.1. Если СК имеет уровень безопасности средний, низкий или критический:

12.1.1. Устранить выявленные уязвимости в коде СК;

12.1.2. Осуществить повторное тестирование и анализ доработанного кода СК на потенциальные уязвимости, начиная с шага 1.

12.2. Если СК имеет высокий уровень безопасности:

12.2.1. Завершить анализ СК.

13. Опубликовать безопасный СК в системе РР.

Результаты работы каждого этапа используются при выполнении работ следующего этапа. Общая последовательность работы статодинамической методики анализа СК представлена на рис. 2.

Отчет безопасности

1. Уровень безопасности смарт-контракта **example.sol: КРИТИЧЕСКИЙ**

2. Обнаруженные уязвимости

2.1. Статический анализ уязвимостей смарт-контракта

Название	Описание	Возможные меры по устранению
Reentrancy (Повторный вход)	При **атаке с повторным входом** СК злоумышленника осуществляет обратный вызов вызывающего СК до того, как первый вызов функции будет завершен.	Использовать **transfer()** вместо **contract.call()** для передачи Ether на ненадежные адреса.
Visibility (модификатор доступа по умолчанию)	**Неуказанный модификатор доступа** для метода может привести к уязвимости и непреднамеренному изменению состояния СК.	Каждая функция должна быть явно определена как **external, public, internal** или **private.**

2.2. Динамический анализ уязвимостей смарт-контракта

Название	Описание	Возможные меры по устранению
Prodigal contract (расточительный контракт)	**Из-за отсутствия** или **недостаточного контроля доступа** злоумышленники могут самостоятельно уничтожить СК, осуществив вызов функции **SELFDESTRUCT.**	Внедрить **меры контроля,** чтобы снятие активов могло быть инициировано только сторонами, которые уполномочены на это, или в соответствии со спецификациями системы СК.

3. Расположение уязвимостей в коде контракта

```
contract Example {
  address owner;

  // called by the constructor
  function initWallet(address _owner) {
    owner = _owner; // any user can change owner
    // more setup
  }
```

Заключение: смарт-контракт **example.sol** содержит потенциальные уязвимости. Необходимо внести изменения в код и осуществить повторный анализ смарт-контракта на безопасность.

ПРОВЕРЕНО

Рис. 1. Пример документарного отчета СК example.sol

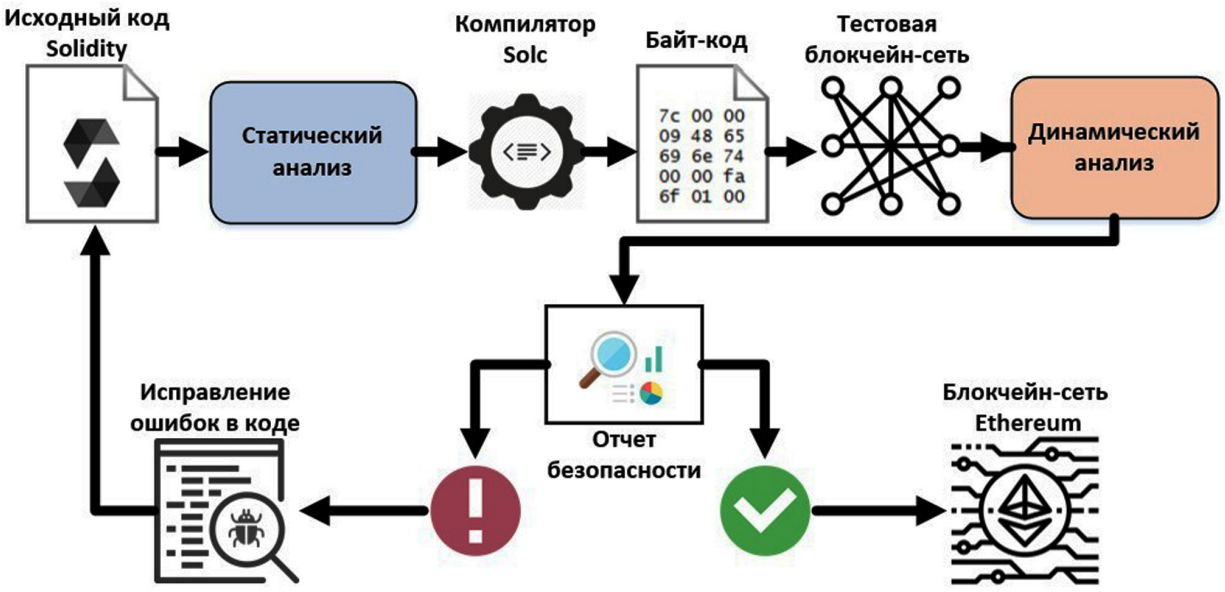

Рис. 2. Архитектура статодинамической методики анализа

Отдельно стоит отметить, что тестирование безопасности СК должно проводиться специалистами, в т. ч. разработчиками, имеющими опыт анализа защищенности СК [11].

При выявлении проблем в процессе тестирования и анализа исходного кода, СК передается на доработку, при этом, исправленный СК должен обязательно пройти повторное тестирование и анализ, прежде чем он будет опубликован в системе РР для дальнейшего выполнения заложенных в него функций.

Повторное тестирование и анализ проводится с целью контроля устранения выявленных недостатков и отсутствия вредоносных конструкций, которые могут привести к краже финансовых активов и нарушению логики работы СК [15].

Выводы

В рамках данной работы рассмотрены основные методы анализа СК. Проанализированы их преимущества и недостатки, а также специфика применения. Сформирован вывод о необходимости применения комплекса методик и инструментов для проведения всестороннего анализа кода СК.

Произведен анализ актуальных уязвимостей ИБ, определены ключевые параметры для расчета коэффициента критичности уязвимости ИБ и уровня безопасности СК, а также описан полный алгоритм поиска уязвимых конструкций в коде СК.

Результатом данной работы является статодинамическая методика анализа уязвимостей СК, которая объединяет в себе инструменты статического и динамического анализа, тем самым позволяя однозначно определять уровень безопасности СК. Полученный результат поможет разработчику принять решение относительно размещения СК в системе РР.

Литература

1. Bishwas C Gupta. Analysis of Ethereum Smart Contracts – A Security Perspective [Текст] / Department of Computer Science and Engineering. Indian Institute of Technology Kanpur, 2019, 70 с.
2. E. Marchenko and Y. Alexandrov. Smartcheck: Static analysis of ethereum smart contracts / 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, ser. WETSEB '18. ACM, 2018. С. 9-16.
3. Репин М.М., Пшехотская Е.А., Простов И.А., Амфитеатрова С.С. Использование платформы на основе распределенных реестров Ripple в банковских платежных системах // Системный администратор № 3(196), 2019. С. 86-89.
4. Atzei N., Bartoletti M., Cimoli T. A survey of attacks on Ethereum smart contracts / Principles of Security and Trust. M.: Springer, Berlin, Heidelberg, 2017. С. 164-186.
5. Фролов А.В. Создание смарт-контрактов Solidity для блокчейна Ethereum. Практическое руководство / Александр Фролов. М.: ЛитРес: Самиздат, 2019, 240 с.
6. Антон Вашкевич. Смарт-контракты: что, зачем и как. М.: Симплоер, 2018, 98 с.
7. Федоров Н.В. Математическое и имитационное моделирование сложных систем. Учебное пособие. М.: МГИУ, 2014, 252 с.
8. A. Mense and M. Flatscher. Security vulnerabilities in ethereum smart contracts. In Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services. ACM, 2018. С. 375-380.

DOI:10.21681/2311-3456-2020-04-56-65

9. Petar Tsankov, Andrei Dan, Dana Drachsler-Cohen, Arthur Gervais, Florian Bünzli, and Martin Vechev. Securify: Practical Security Analysis of Smart Contracts. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018. С. 67-82.
10. Nikolic, Ivica & Kolluri, Aashish & Sergey, Ilya & Saxena, Prateek & Hobor, Aquinas. Finding the Greedy, Prodigal, and Suicidal Contracts at Scale. ACSAC, 2018. С. 653-663.
11. G. Bigi, A. Bracciali, G. Meacci, and E. Tuosto. Validation of decentralised smart contracts through game theory and formal methods. In Programming Languages with Applications to Biology and Security. Springer, 2015. С. 142-161.
12. Репин М.М., Пшехотская Е.А. Обеспечение информационной безопасности смарт-контрактов в системах на основе технологии распределенных реестров // Системный администратор № 5(198), 2019. С. 70-73.
13. B. Marino and A. Juels. Setting standards for altering and undoing smart contract. In International Symposium on Rules and Rule Markup Languages for the Semantic Web. Springer, 2016. С. 151-166.
14. Пителинский К.В., Александрова А.В. Структура и принцип работы смарт-контрактов // Сборник научных статей по итогам работы круглого стола с международным участием. 15-16 января 2020 г. Часть 2, 2020, С. 101-103.
15. Ilya Grishchenko, Matteo Maffei, and Clara Schneidewind. A Semantic Framework for the Security Analysis of Ethereum Smart Contracts. In Principles of Security and Trust, Lujo Bauer and Ralf Küsters (Eds.). Springer International Publishing, Cham, 2018. С. 243-269.

Рецензент: *Цирлов Валентин Леонидович, кандидат технических наук, доцент кафедры ИУ-8 «Информационная безопасность» МГТУ им. Н.Э. Баумана, г. Москва, Россия. E-mail: v.tsirlov@bmstu.ru*

METHODOLOGY FOR ANALYZING VULNERABILITIES AND DETERMINING THE SECURITY LEVEL OF A SMART CONTRACT WHEN PLACED IN DISTRIBUTED LEDGER SYSTEMS

Krivonogov A.A.[4], Repin M.M.[5], Fedorov N.V.[6]

Every year, the technology of using smart contracts is attracting more and more attention from users due to the unique advantages that it possesses: automatic execution of transactions in a traceable and unchanging way without third party authorization. At the same time, a smart contract is one of the most vulnerable elements in distributed ledger systems, which can be susceptible to attack by intruders.

***The aim of the research** is to develop a methodology that allows analyzing a smart contract for information security vulnerabilities and determining the security level of a smart contract before placing it in distributed ledger systems.*

***Research methods:** to achieve this goal, methods of static and dynamic analysis were studied, the most relevant information security vulnerabilities were identified, and parameters for calculating the criticality factor of vulnerability and the security level of a smart contract were determined.*

***Result:** a promising static-dynamic method for analyzing the vulnerabilities of a smart contract is proposed, which makes it possible to unambiguously determine the security level of a smart contract before its placement in the distributed ledger system. Its main parameters are set, and the reference security factors of a smart contract are determined. The complete algorithm of the static-dynamic method of analyzing a smart contract is described, and an example of a generated documentary security report based on the results of analyzing a smart contract is given.*

***Keywords:** static analysis, dynamic analysis, automated tools, information security, vulnerability criticality, testing, code audit.*

References

1. Bishwas C Gupta. Analysis of Ethereum Smart Contracts – A Security Perspective / Department of Computer Science and Engineering. Indian Institute of Technology Kanpur, 2019, 70 p.
2. E. Marchenko and Y. Alexandrov. Smartcheck: Static analysis of ethereum smart contracts / 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, ser. WETSEB '18. ACM, 2018, pp. 9-16.

4 Anton Krivonogov, postgraduate student of the Department of Information Security, Moscow Polytechnic University, Moscow, Russia. E-mail: aakrivonogov97@yandex.ru

5 Maxim Repin, Head of the direction of the Research Institute «Voskhod», Moscow, Russia. E-mail: bmstu.iu8@gmail.com

6 Nikolay Fedorov, Ph.D. (Engineering), Associate Professor, Head of the Department of Information Security, Moscow Polytechnic University, Moscow, Russia. E-mail: fedorovnv31@mail.ru

3. Repin M.M., Pshehotskaya E.A., Prostov I.A., Amfiteatrova S.S. Ispol'zovanie platformy na osnove raspredelennykh reestrov Ripple v bankovskikh platezhnykh sistemakh // Sistemnyy Administrator № 3 (196), 2019, pp. 86-89.
4. Atzei N., Bartoletti M., Cimoli T. A survey of attacks on Ethereum smart contracts / Principles of Security and Trust. – M.: Springer, Berlin, Heidelberg, 2017, pp. 164-186.
5. Alexander Frolov. Sozdanie smart-kontraktov Solidity dlya blokcheyna Ethereum. Prakticheskoe rukovodstvo / Alexander Frolov. – M.: Litres: Samizdat, 2019, 240 p.
6. Anton Vashkevich. Smart-kontrakty: chto, zachem i kak. – M.: Simploer, 2018, 98 p.
7. Fedorov N.V. Matematicheskoe i imitatsionnoe modelirovanie slozhnykh sistem. Uchebnoe posobie. – M.: MGIU, 2014, 252 p.
8. A. Mense and M. Flatscher. Security vulnerabilities in ethereum smart contracts. In Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services. ACM, 2018, pp. 375-380.
9. Petar Tsankov, Andrei Dan, Dana Drachsler-Cohen, Arthur Gervais, Florian Bünzli, and Martin Vechev. Securify: Practical Security Analysis of Smart Contracts. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018, pp. 67-82.
10. Nikolic, Ivica & Kolluri, Aashish & Sergey, Ilya & Saxena, Prateek & Hobor, Aquinas. Finding The Greedy, Prodigal, and Suicidal Contracts at Scale. ACSAC, 2018, pp. 653-663.
11. G. Bigi, A. Bracciali, G. Meacci, and E. Tuosto. Validation of decentralised smart contracts through game theory and formal methods. In Programming Languages with Applications to Biology and Security. Springer, 2015, pp. 142-161.
12. Repin M.M., Pshehotskaya E.A. Obespechenie informatsionnoy bezopasnosti smart-kontraktov v sistemakh na osnove tekhnologii raspredelennykh reestrov // Sistemnyy Administrator № 5 (198), 2019, pp. 70-73.
13. B. Marino and A. Juels. Setting standards for altering and undoing smart contracts. In International Symposium on Rules and Rule Markup Languages for the Semantic Web. Springer, 2016, pp. 151-166.
14. Pitelinsky K.V., Alexandrova A.V. Struktura i printsip raboty smart-kontraktov // Sbornik nauchnykh statey po itogam raboty kruglogo stola s mezhdunarodnym uchastiem. 15-16 yanvarya 2020 g. Chast' 2, 2020, pp. 101-103.
15. Ilya Grishchenko, Matteo Maffei, and Clara Schneidewind. A Semantic Framework for the Security Analysis of Ethereum Smart Contracts. In Principles of Security and Trust, Lujo Bauer and Ralf Küsters (Eds.). Springer International Publishing, Cham, 2018, pp. 243-269.