СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc72315409)

[ВВЕДЕНИЕ 2](#_Toc72315410)

[СЛОВАРЬ 4](#_Toc72315411)

[Глава 1. Исследование существующих методов анализа графов 5](#_Toc72315412)

[1.1 Роль аудита в информационной экосистеме 5](#_Toc72315413)

[1.2 Проблема ограниченности времени и ресурсов 11](#_Toc72315414)

[1.3 Графы в статическом анализе 13](#_Toc72315415)

[Глава 2. Концепция метода решения задачи 14](#_Toc72315416)

[2.1 Необходимые для применения Метода условия 14](#_Toc72315417)

[2.2 Общее описание метода 15](#_Toc72315418)

[2.3 Применение Метода на различной глубине анализа 17](#_Toc72315419)

[Глава 3. Разработка алгоритма имплементации метода 21](#_Toc72315420)

[3.1 Логическая структура алгоритма 21](#_Toc72315421)

[3.2 Используемые ресурсы 25](#_Toc72315422)

[3.3 Примеры имплементации 29](#_Toc72315423)

[3.4 Выводы на основе результатов алгоритма 41](#_Toc72315424)

[Глава 4. Моделирование проведения аудита информационной безопасности с использованием Метода 42](#_Toc72315425)

[4.1 Описание предприятия 42](#_Toc72315426)

[4.2 Оценка Метода 44](#_Toc72315427)

[ЗАКЛЮЧЕНИЕ 53](#_Toc72315428)

[СПИСОК ЛИТЕРАТУРЫ 54](#_Toc72315429)

[Приложение А 56](#_Toc72315430)

ВВЕДЕНИЕ

В современном мире человечество серьезно положилось на информационные технологии. Они ежегодно развиваются, медленно заменяя новым техническим оборудованием старое и быстро обрастая всё усложняющимся программным обеспечением (далее — ПО).

Развитие ПО обусловлено не только структурной оптимизацией программ, но и увеличением доступного функционала и подходов к решению задач (например машинное обучение, нейросети и т.д.). Благодаря увеличению мощностей стала возможной обработка больших массивов разнообразных данных: картинок, литературного текста, устной речи, повсеместных отчетов и логов. Однако для реализации работы с такими массивами требуются серьезные информационные экосистемы.

По природе своей информационные экосистемы сложно структурированы, распределены на разных устройствах и используют разные подходы для работы с данными (в основном структурированные базы данных, языки разметки, различные языки программирования и конечные программные продукты). Такие структуры неизбежно содержат в себе уязвимости.

Рано или поздно уязвимости в таких системах подвергаются эксплуатации и это приводит к репутационным и финансовым издержкам. Статистика явно показывает тенденцию к увеличению количества эксплуатаций уязвимостей, что создает спрос на защиту экосистем, систем реагирования, качественные аудиты.

В целях предотвращения утечек с помощью эксплуатации уязвимостей проводится аудит информационной безопасности, в основе которого лежат методы анализа информационных систем.

Методы анализа систем вынуждены поспевать за быстрорастущими по сложности и размерам экосистемами. Эксперты по информационной безопасности применяют старые методы поиска уязвимостей, которые не масштабируются, поэтому требуют огромных временных и интеллектуальных затрат для негарантированного результата.

В таких условиях необходимы новые подходы к поиску уязвимостей. В своей работе я предлагаю масштабируемый интеллектуальный метод, применяемый в рамках аудита для повышения эффективности экспертного статического анализа.

СЛОВАРЬ

Информационная Экосистема (Далее ИЭ) - информационная система (далее — ИС) предусматривающая сбор, хранение, обработку и поиск информации, включающая в себя учет бизнес-процессов, юридического документооборота, бизнес-аналитики, циркуляции информации между модулями, из которых состоит ИЭ.

Функциональный модуль — ИС с узконаправленным функционалом для работы с единицами информации, интегрированная в ИЭ и взаимодействующая с другими модулями, состоящая из программных файлов.

Единица информации (далее ЕИ) - файл любого формата, кроме программных, хранящий информацию любого типа.

Программный файл — файл с кодом программ на любом языке программирования, любого расширения и структуры используемыми для идентификации языка.

Уязвимость — недостаток в информационной системе, эксплуатируя который, возможно нарушить её целостность, конфиденциальность и\или доступность данных и/или вызвать неправильную работу системы.

Временная бомба — заранее спланированная атака, использующая различные уязвимости в системе, реализуемая на протяжении долгого времени.

Граф — математическое представление объектов, их некоторых свойств и взаимодействий между этими объектами.

Узел — объект в графе, обладающий или не обладающий свойствами.

Грань — представление взаимодействия между объектами в графе, обладающее или не обладающее свойствами.

1. Исследование существующих методов анализа графов
   1. Роль аудита в информационной экосистеме

Профессиональный аудит есть неотъемлемая часть любого исправно функционирующего предприятия. Подходы к проведению аудитов могут разниться, от индивидуально составленных экспертами методов до принятых международных стандартов информационной безопасности.

Полноценный аудит проводится перед введением системы в рабочее состояние и затем регулярно. Суть аудита состоит в оценке текущего состояния системы, а оценка в свою очередь опирается на методы анализа систем. С помощью аудита можно предотвратить эксплуатацию уязвимостей до атаки или во время неё, а также обнаружить потенциальные утечку информации.

Разделяют два основных вида аудитов: внешний и внутренний. Внутренний подразумевает проведение исследования ИЭ предприятия штатными сотрудниками предприятия, специализирующимися на информационной безопасности. Внешний, в свою очередь, проводится экспертами информационной безопасности, не работающими на предприятии непосредственно, однако работающими на компанию, заключившую договор о предоставлении услуги “Проведение аудита с предприятием”.

Однако проведение любого аудита подразумевает определенный набор базовых действий описанных ниже.

* + 1. Подготовка проведения аудита

Целью первого этапа аудита является определение требований к аудиту, организация его проведения. Требования к аудиту зависят от ближайших целей предприятия связанных с ИЭ. Такими целями могут быть:

* Непосредственный запуск предприятия, соответственно, требуется провести полноценный единоразовый внешний аудит;
* Внедрение в ИЭ нового функционального модуля, потребуется внешний аудит внедряемого модуля;
* Расследование инцидента информационной безопасности на предприятии, также потребуется полный внешний аудит;
* Регулярная проверка ИЭ предприятия, для такой цели подойдет внутренний аудит.

Организация проведения аудита требует приготовления всех подразделений предприятия, работающих с ИЭ, для взаимодействия с аудиторами, юридическое документирование проведения аудита.

* + 1. Сбор информации об ИЭ

Информация об информационной экосистеме разнообразна, а для качественного аудита требуется как можно больше информации. Её сбор — длительный и сложный процесс. Источниками информации служат:

* документация ИЭ;
* правила внутри предприятия, особенно правила информационной безопасности;
* логи;
* права и полномочия всех работающих с ИЭ;
* инструменты анализа информационных систем и сетей;
* интервью с сотрудниками.

Собранную информацию также необходимо структурировать и организовать, чтобы её возможно было использовать для анализа.

* + 1. Анализ данных аудита

Информационная экосистема состоит из систем разных природ:

Социально-трудовая система, регламентирующая обязанности, правила поведения сотрудников

Юридическая система, регламентирующая правовую деятельность предприятия и документооборот

Информационная система, автоматизирующая и реализующая необходимый для работы предприятия функционал

Социально-трудовая система представляет интерес для аудита и может предоставить полезные знания об ИЭ. В первую очередь, правила информационной безопасности вкупе с записями об их невыполнении сотрудниками указывают на недостатки данной системы. С другой стороны, знания о полномочиях и возможностях сотрудников могут указать на потенциальные уязвимости в ИЭ.

Юридическая система, в частности, документооборот предприятия представляет интерес для аудита, в особенности утилизация и хранение документов. Утилизированные бумажные документы могут стать источником утечки. Электронный документооборот непосредственно реализован в информационной системе и тесно связан с социально-трудовой системой, поэтому информационная система представляет особый интерес при исследовании.

Изучение информационной системы является основным интересом исследования. Информационные системы — это комплексные структуры, состоящие из физической и программных частей. Под физическими частями подразумеваются компьютеры, сервера и другие электронно-вычислительные машины. Анализом побочного электромагнитного излучения и наводок занимаются эксперты соответствующей специализации. В данной работе пэмин не представляет интереса.

Программная составляющая информационной системы — это обширная область экспертизы. Технологии написания программ, языки программирования, управление памятью компьютеров серьезно развились за последние 20 лет. Большинство уязвимостей и проблем, связанных с памятью компьютера, были решены аккуратным проектированием операционных систем. Однако даже самые лучшие решения имеют свои недостатки, проблемы и уязвимости, поэтому чтобы их митигировать, требуется детальный экспертный анализ. Это в первую очередь касается программ, используемых операционными системами на компьютерах.

Экспертный анализ программ и операционных систем подразумевает анализ используемого кода. Разделяют два вида анализа: динамический и статический (см. Рис. 1).

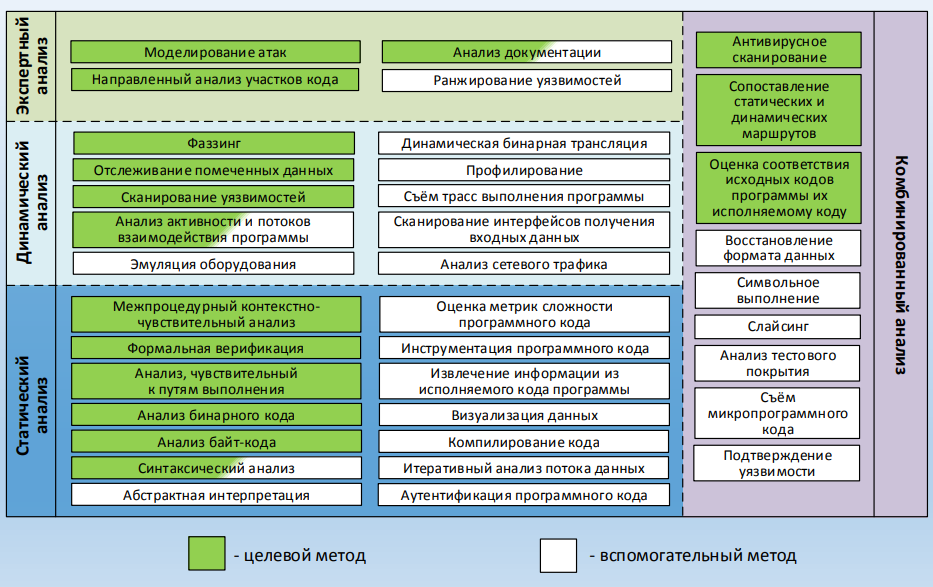


Рисунок 1 — Перечисление всех методов анализа экосистем

* + - 1. Динамический анализ

Динамический подход поиска уязвимостей заключается в исследовании системы во время её работы в реальном окружении или в виртуальном. Такой метод предлагает ряд преимуществ, таких как достоверность обнаруженных уязвимостей и возможность обойтись без исходного кода. Однако такой метод не обнаруживает потенциальные уязвимости, пригодные для использования в рамках Временной бомбы, как и эксплуатации на местах.

В него входят следующие основные методы:

Фаззинг — метод проверки ПО с помощью анализа реакции программ на получение невалидных и непредусмотренных данных.

Отслеживание помеченных данных — метод, подразумевающий ввод в программу особым образом сформированных ЕИ и дальнейшего отслеживания взаимодействий с ними.

Сканирование уязвимостей — метод, основанный на программном обнаружении доступных портов, активных IP-адресов, изучении топологии компьютерной сети и используемых сетевых сервисов.

Анализ активности и потоков взаимодействия программы — метод обнаружения уязвимостей, попыток проникновения в ИС и ошибок, основанный на непрерывном отслеживании состояния ПО в ИС.

Съем трассировки выполнения программ — метод анализа работы ПО с помощью изучения пошаговой работы программы.

Анализ сетевого трафика — метод обнаружения вредоносных программ путем изучения проходящих через сеть данных.

* + - 1. Статический анализ

Статический анализ подразумевает глубокое и всестороннее исследование информационной экосистемы. В него входят следующие методы:

Межпроцедурный контекстно-чувствительный анализ — метод анализа кода, основанный на всестороннем изучении функций программы с учетом контекста программы.

Формальная верификация — метод установления соответствия между формальным описанием ИС и её имплементацией.

Анализ байт-кода — метод изучения программ после их преобразования в байт-код.

Абстрактная интерпретация — метод представления работы выполнения кода, без его непосредственного выполнения.

Визуализация данных — метод интерпретации данных с помощью графических средств.

Итеративный анализ потока данных — метод обнаружения и накопления возможных наборов значений, образующихся в процессе выполнения программы.

* + 1. Оценка рисков и выработка рекомендаций

Обнаруженные уязвимости будут разными по своей природе, как следствие, цена за их исправление будет различаться. С другой стороны, риски реализации угроз и убытки от реализации угроз также различаются. Уязвимости и риски детально изучаются, и знания о них систематизируются.

Затем для каждой системы вырабатываются свои рекомендации на основе анализа данных об ИЭ и оценке рисков.

Для социально-трудовой системы в качестве рекомендаций составляются обновленный набор правил для сотрудников, при необходимости проведение регулярных мероприятий, как образовательных, так и связанных с непосредственной проверкой ИЭ.

Рекомендации для юридической системы напрямую связаны с оптимизацией документооборота.

Для информационной системы рекомендации, как и при анализе, будут направлены на физическую и программную части системы. Для защиты физической части системы необходимо как специальное оборудование, различные устройства, управляющие работоспособностью ИС, её подключением к сети и питанию, так и возможная реструктуризация локальной сети предприятия. Для защиты программной части информационной системы необходим комплексный подход: подключение программных продуктов, защищающиx операционные системы, средства контроля доступа, обновление программного обеспечения, подключение программного обеспечения, отслеживающего деятельность системы.

* + 1. Подготовление отчета

В отчете собираются и систематизируются все знания об ИЭ. Впоследствии отчет предоставляется лицам, принимающим решения на предприятии.

* 1. Проблема ограниченности времени и ресурсов

Несмотря на современные достижения технологического развития, информационные экосистемы остаются комплексным предметом для анализа. Сбор информации затрудняется не только структурной сложностью ИЭ, но и обильным количеством собираемой информации.

Современные предприятия используют технологии больших данных как основу принятия решений. Для этого требуется собирать всю доступную информацию о клиентах и продукции. Каждая деталь должна быть отражена в цифровом виде. Датчики и программы логирования устанавливаются повсеместно в ИЭ. Такой подход к сбору информации неизбежно создает большое количество входов в систему. Более того, сбор разнообразной информации требует различных программных решений, что непременно увеличивает сложность информационной системы и количество функциональных модулей.

С другой стороны, вся информация, проходящая через ИЭ, обрабатывается в локальных функциональных модулях. При компрометации или выходе из строя модуля для большей части системы нарушаются принципы доступности и целостности. Это важно потому что вся экосистема работает используя результаты от анализа собранных данных. Принцип доступности критически важен для экосистемы, так как обычно в таких структурах решения принимаются основываясь на данных из разных частей экосистемы. В случае нарушения принципа доступности, принять оптимальное решение будет невозможно. Нарушение принципа целостности может привести к серьезному искажению понимания информации системой.

Для глубокого экспертного анализа таких экосистем при аудите требуются огромные затраты времени и усилий. Наибольшую трудность составляет анализ кода, т.к. экосистемы объемны по своим размерам и разнообразны внутри за счет использования различных технологий для работы с информацией. Для динамического анализа кода существует широкий спектр программного обеспечения, автоматизирующий проверку уязвимостей в системе, при этом такое ПО масштабируемое.

Однако статический анализ до сих пор в основном сводится к ручному анализу кода, что есть трудоемкий и времязатратный процесс. Хоть современное ПО позволяет частично автоматизировать процесс интерпретации кода, этого недостаточно, чтобы значительно увеличить скорость статического анализа. Возможное решение данной проблемы представляет собой распределение приоритетов в ходе анализа. Избирательный анализ кода жертвует эффективностью, но значительно ускоряет процесс. Когда программные файлы ранжированы, важнейшие из ниx будут изучены в первую очередь, что может принести ключевые знания о системе.

Закономерно встает вопрос определения “важности” файла. Для этого следует обратиться к правильной постановке представления информационной экосистемы.

Представление экосистемы с помощью таблиц удобно для применения различных статистических методов. Статические методы исправно работают с однородными видами информации, в то время как экосистема отличается разнообразием функциональных модулей, что делает какую-либо выборку о ИЭ нерепрезентативной, а информацию в таблице избыточной.

Представление экосистемы с помощью текста, иными словами документация, важно и полезно для получения деталей об ИЭ. Однако документация не всегда ведется. При наличии документации ранжировать программные файлы для исследования не обязательно, однако такое ранжирование будет целесообразно для понимания программной логики работы функциональных модулей. Тем не менее, придется внимательно изучать документацию и проверять соответствие экосистемы тексту.

С точки зрения гибкого, читаемого и репрезентативного представления экосистемы предлагается использовать графы. Главное преимущество графов — масштабируемость. Также при правильном оформлении графы хорошо читаемы, и к ним можно применять методы статистического анализа.

* 1. **Графы в статическом анализе**

При статическом анализе графов выделяют различные структурные подходы: выявления аномалий на основе признаков узлов, групп узлов, на основе сообществ, на основе метрик близкого расположения.

Графы разделяются на обладающие свойствами и не обладающие свойствами на узлах и ребрах, ориентированные и неориентированные.

На данный момент в графах наиболее современные подходы анализируют формы, закономерности связей между узлами, характеристики узлов.

* + 1. Графы без свойств

Графы без свойств являются самыми простыми графами. В случае исследования таких графов основным объектом изучения является структура графов, кучность узлов.

Для исследования структур и кучности используются различные методы. PageRank и Personalized PageRank одни из них, они основаны на случайном прохождении графа. Другим методом является SimRank — метод, рассчитывающий похожесть частей графа. Метод Katz рассчитывает отношение веса к длине граней. Метод HITS (Hyperlink Induced Topic Search) анализирует ориентированные графы, основываясь на количестве входящих и исходящих из узла граней, алгоритм присваивает им качественные оценки.

Также для таких графов стоит применять методы матричной факторизации, основной идеей которой является преобразование графа из одного в несколько, сохраняющих структурные свойства оригинального.

* + 1. Графы со свойствами

Графы, обладающие свойствами, могут предоставить более емкую информационную картину системы, с которой они спроецированы, по сравнению с графами без свойств.

Методы, использующиеся для графов без свойств применяются с учетом наличия свойств. С другой стороны, наличие свойств позволяет различать узлы друг от друга, на этой основе можно делать выводы, какие узлы являются нормальными, а какие аномальными.

Для таких графов используются следующие методы:

Minimum Description Length — метод основанный на поиске “лучших структур”, подграфов, которые повсеместно и часто появляются в графе. Выделяя такие подграфы выявляются структуры, отличающиеся от “лучших”.

CODA — кластерный вероятностный метод, выявляющий выбросы в графе, который основывается на свойствах узлов и групп узлов.

gOutRank — метод, обнаруживающий выбросы в графах больших размерностей, из-за масштабов графа любые структуры в нем выглядят похоже, однако при детальном рассмотрении подграфов, объединенных по общему признаку, авторы выявляют аномалии.

1. Концепция метода решения задачи
   1. Необходимые для применения Метода условия

Метод интеллектуального анализа и визуализации графов (далее — Метод) предназначен для использования экспертным аудитором. Метод предназначен для применения в аудите информационной безопасности. Для программной имплементации Метода необходимы:

* компьютер;
* источник данных об исследуемой системе;
* программная среда разработки с использованием языков программирования.

Метод применяется для экосистем всех размеров. Метод пригоден для использования на всех операционных системах, на личных компьютерах, серверах и локальных сетях.

* 1. Общее описание метода

Метод представляет собой последовательное интерпретирование ИЭ с помощью графов, как математически, так и визуально. Базовая идея Метода состоит из 4-х шагов:

* + 1. Определение узлов.

Узлом могут быть отдельные программные файлы, программы, несколько программ, выполняющих общий функционал. В отдельном случае узлами могут являться персональные компьютеры, сервера или люди.

* + 1. Установление связей между узлами

Связи могут быть двух видов:

Ненаправленными связями являются те, в которых отношения между модулями равноправны, т.е. модули могут посылать и принимать ЕИ друг от друга. В основном, это связи персональных компьютеров друг с другом и с серверами.

Направленными связями являются те, в которых отношения между модулями неравноправны, т.е. узел может либо посылать, либо принимать ЕИ другого.

* + 1. Анализ метрик модуля

Для анализа графов используются два алгоритма анализа, основанные на итеративном расчете метрик узлов:

* + - 1. Hyperlink Induced Topic Search (далее — HITS)

Этот алгоритм использует две метрики, Переходность и Авторитетность.

Переходность — метрика, выражающая значимость узла в зависимости от количества исходящих в узел связей и Авторитетности этих узлов.

,

Где Pfrom — это массив всех исходящих связей.

Авторитетность — метрика, выражающая значимость узла в зависимости от количества входящих в узел связей и Переходности этих узлов.



Где Pto — это массив всех входящих связей.

Этот алгоритм неприменим для неориентированных графов.

* + - 1. PageRank

Алгоритм вычисляет метрику Важности. Алгоритм использует случайное блуждание, которое берет в учет Важность узлов. Если у узла высокая Важность, то очередная итерация блуждания вероятнее всего будет включать этот узел.

Важность каждого узла рассчитывается, опираясь на соседние узлы, их Важность и количество исходящих связей, которое они имеют.



Где D — это демпинговый фактор, значение, предотвращающее исключение узлов с небольшой Важностью,

P — узел,

n — количество узлов,

PR — значение Важности,

Outdegree — количество исходящих связей узла,

E — все узлы с входящими к текущему узлу связями.

Для обоих алгоритмов изначальным значением каждой метрики каждого узла является 1.

* + 1. Визуализация графа

На этом шаге граф визуализируется. Узлы окрашиваются в цвет соответствующего значения метрики относительно градиента. Также на визуальном графе отображаются грани и их направления.

Взаимодействия с ЕИ Методом не учитываются, т.к. при работе с ЕИ необходим другой подход.

* 1. Применение Метода на различной глубине анализа

Информационные системы и архитектура программ используют в своей основе принцип модульности. Это значит, что части программ и ИС заменяемы, но также значит, что они ограничены и связаны с другими частями, в программах посредством подключения библиотек, в ИС посредством локальной сети.

Как следствие, Метод можно применять и к ИС и к программам.

* + 1. Уровень программы

На уровне программы в качестве узлов выступают программные файлы: подключаемые библиотеки, базовые программы.

Направленной к узлу гранью является программное подключение библиотек и вызовы базовых программ. Направленной от узла грани в данном контексте не существует, т.к. программа самостоятельно не совершает действия в другой базовой программе или библиотеке, только если она не является вредоносной. Для обнаружения такого взаимодействия требуются другие подходы, которые данной работой не предусмотрены.

Алгоритмы анализа графа применяются с параметрами по умолчанию. При обоих алгоритмах первоначальные значения всех метрик равняется единице. Для алгоритма PageRank демпинговый фактор по умолчанию равен 0.15. Алгоритмы итеративно рассчитывают значения метрик для каждого узла. После определенного количества итераций метрики не изменяются, т.к. система достигает равновесия. Метрики Важности и Авторитетности указывают на узлы сильно влияющие на систему. Узлы с высокой Переходностью чаще всего содержат в себе логику программы.

На основе узлов и их взаимодействий рисуется граф. Для HITS изображаются 2 рисунка, для PageRank — 1 рисунок. На рисунках цвет каждого узла соответствует условному значению метрики. Цветовая палитра конфигурируется вручную, её минимальное значение соответствует минимальному значению метрики и наоборот. На графе также отображаются названия узлов.

* + 1. Уровень функционального модуля

На уровне функционального модуля в качестве узлов выступают программные файлы. Однако в отличии от программы функциональные модули состоят из нескольких программ. Обычно функциональные модули состоят из небольшого количества программ, поэтому проблем с читаемостью и скоростью анализа или визуализации графа не возникнет. Тем не менее, возможное решение таких проблем описано в следующем пункте.

Направленные грани образовываются таким же образом, как и при анализе программ.

Алгоритмы применяются с параметрами по умолчанию. Метрики рассчитываются для всех узлов. Принципиальным отличием является глубина анализа, т.к. ранжируются файлы не одной программы, а нескольких.

Визуализация графа аналогична уровню программы.

* + 1. Уровень операционной системы

На уровне операционной системы в качестве узлов могут выступать как программные файлы, так и функциональные модули.

В случае программных файлов во избежание проблем с читаемостью и скоростью анализа графа резонно применять методы уменьшения размерности графа. При этом необходимо учесть такое преобразование во время применения алгоритмов.

В случае функциональных модулей, несколько программных файлов, необходимых для выполнения функции, объединяются в один узел. Такое преобразование производится заранее, и оно основано на общем функционале программ, а не на параметре близости, как при методах уменьшении размерности.

Направленные грани в случае программных файлов образовываются так же, как на уровне программы. При сокращении размерности связи сохраняются. В случае функционального модуля, грани образовываются, если у хотя бы одного программного файла из узла есть хотя бы одна связь с программным файлом в другом функциональном модуле.

Алгоритмы применяются с параметрами по умолчанию, за исключением начальных значений метрик узлов для объединенных программных файлов, значение их начальных значений метрик суммируются. Метрики рассчитываются для всех узлов. Принципиальным отличием является глубина анализа, т.к. ранжируются файлы не одной программы, а нескольких.

Визуализация графа аналогична уровню программы.

* + 1. Уровень локальной сети

На уровне локальной сети в качестве узлов выступают персональные компьютеры и сервера в ИЭ.

Грани в локальной сети ненаправленные, т.к. компьютеры и сервера, подключенные друг к другу, обмениваются информацией в обе стороны. Тем не менее грани могут быть направленными, если соединения в локальной сети настроены на передачу информации только в одну сторону.

Алгоритм PageRank применяется с параметрами по умолчанию. Метрики Важности указывает на самый влиятельный компьютер или сервер.

Визуализация графа аналогична уровню программы.

* + 1. Уровень модулей обработки информации

На уровне обработки в качестве узлов выступают программные файлы или функциональные модули, непосредственно взаимодействующие с единицами информации и преобразующие их.

В качестве направленной грани в графе будут выступать информационные взаимодействия между узлами, а именно: если узел передает ЕИ, то образуется связь между источником и получателем ЕИ.

Алгоритмы применяются с параметрами по умолчанию. Метрики Важности и Авторитетности указывают на узлы сильно влияющие на систему. Узлы с высокой Переходностью чаще всего содержат в себе логику программы относительно взаимодействий с ЕИ.

Визуализация графа аналогична уровню программы.

* + 1. Уровень социально-трудовой системы

В качестве узлов выступают люди. Этот уровень пригоден для поиска злонамеренных инсайдеров.

В качестве ненаправленных граней выступают взаимодействия людей, информация об этом изымается из записей общения посредством электронной почты и/или из записей отдела кадров. При этом важную роль играет частота обмена сообщениями между сотрудниками.

Наименее Важные и Авторитетные люди по оценке алгоритмов являются ключевыми к нахождению инсайдеров, т.к. асоциальные люди чаще склонны совершать неправомерные действия. Частота обмена сообщениями между сотрудниками также играет важную роль в оценке Важности и Авторитетности.

При визуализации ярким цветом следует выделять узлы с наименьшей Важностью и Авторитетностью. Ширина грани на графе отображает частоту общения между сотрудниками.

1. Разработка алгоритма имплементации метода
   1. Логическая структура алгоритма

Алгоритм достоверно применяется в \*nix-подобных операционных системах и в операционных системах семейства Windows. Алгоритм предположительно аналогично работает для операционных систем семейства macOS.

Существование алгоритма обусловлено файловой структурой и организацией процессорных потоков вышеперечисленных операционных систем. Исторически для экономии памяти в программировании была сформулирована идея библиотек, файлов, не являющихся исполнительными, но хранящими в себе описание функций, обычно реализующих определенный функционал. Такая организация программ экономила место, однако она также создавала уязвимости и баги различных видов.

Алгоритм работает с исполнительными и библиотечными программными файлами и состоит из нескольких шагов. Предварительный анализ системы подразумевает определение групп файлов относящихся к определенным программным средствам или функциональным модулям. Файлы, принадлежащие к программному средству, обычно тесно друг с другом связаны, а также находятся в одной и той же папке или папке рядом по иерархии.

* + 1. Подготовка

На шаге подготовки уже должны быть определены цель исследования системы, область в системе, которую следует изучить. Все файлы из этой области должны быть доступны аудитору. Областью может быть программа, исполнительный программный файл, функциональный модуль, все программы реализующие функционал модуля, или вся операционная система.

Для работы алгоритма необходимо среда разработки и программное средство визуализации графов.

* + 1. Сканирование файловой системы

Под сканированием подразумевается последовательное считывание наименований файлов и определение абсолютных путей к ним. Результатом сканирования частей системы станет список абсолютных путей ко всем файлам в выбранной области файловой системы.

* + 1. Определение типов файлов

Общий случай определения типа файла заключается в чтении сигнатур файла или классификации его структуры. Для каждой операционной системы существуют свои особенности определения типа файла, так, например, в unix-подобных системах существуют утилита, классифицирующая файлы с помощью тестов. В Windows же, для простого и быстрого определения типа файла системой используется расширение, короткий набор символов в конце наименования файла после точки.

После проведения сканирования области файловой системы необходимо определить, какие из файлов представляют наибольший интерес с точки зрения уязвимости и функционала. В первую очередь это исполняемые файлы, типов ELF и PE. Файлы типа ELF и PE одновременно реализуют функционал исполнительных файлов и файлов библиотек. В их структуре также явно указываются взаимодействия с файлами таких же типов.

* + 1. Выявление точек взаимодействия

Далее необходимо установить в каких взаимосвязях находятся файлы. Для всех файлов в списке необходимо установить местоположения точек взаимодействий с другими файлами. Исполнительные файлы типа PE и ELF непосредственно связаны с другими файлами таких же типов двумя способами: обращением к другим файлам в сегменте кода, обращение к другим файлам в сегменте импорта или сегменте динамического связывания. Сведения о связях изымаются программным путем.

В конкретной имплементации связи обозначают импорт. Направление ориентированной грани определяется следующим образом:узел в импорте которого указан другой узел является начальным, а указанный узел — конечным.

* + 1. Построение модели связей

Для построения графа используется модель. Модель представлена графом, узлами графа являются файлы, ребрами графа являются связи между файлами, граф может располагаться в трехмерном пространстве, где на каждой плоскости расположены файлы определенного типа. Так как одни файлы обращаются к другим, ребра на графе будут ориентированными. Обращения между файлами может быть разными, это должно отражаться и на стиле ребер.

В конкретной имплементации граф располагается в двухмерном пространстве с единственным видом связей, а именно импортом. Программным образом формируется граф, содержащий записи об узлах, гранях и свойствах обоих. Для этого используются две библиотеки. Библиотека networkx используется для последующего графического представления графа. Библиотека PageRank-HITS-SimRank используется для расчета значений метрик алгоритмов HITS и PageRank.

* + 1. Интеллектуальный анализ

Интеллектуальный анализ основывается на модели. Для файлов естественно обращаться к системным файлам, файлам динамических библиотек того же программного средства, исполняемым файлам того же программного средства, файлам языковых сред.

Интеллектуальный анализ модели посвящен обнаружению выделяющихся узлов. В качестве методов интеллектуального анализа выступают алгоритмы HITS и PageRank.

После проведения вычислений метрик узлов с помощью алгоритмов, значения метрик присваиваются узлам в виде свойств.

* + 1. Визуализация графа

Заключительным шагом является визуализация графа. На этом шаге две ступени, предварительная визуализация и полноценная визуализация.

Предварительная визуализация графа проводится непосредственно в среде разработки с помощью библиотек networkx и matplotlib. Благодаря предварительной визуализации можно выяснить как примерно будет выглядеть граф и есть ли необходимость изменения конфигурации алгоритмов для достижения более качественного результата визуализации и читаемости.

Полноценная визуализация проводится с помощью программного средства Gephi. Сначала граф со всей информацией об узлах, взаимодействиях и их свойствах импортируется в формат GEXF, затем файл открывается в Gephi. Используя первый интерфейс Gephi, настраивается местоположение и наименования узлов графа, вид узлов и граней. В третьем интерфейсе отображается предпросмотр настраиваемого изображения графа. С помощью третьего интерфейса граф импортируется для последующего экспертного визуального анализа.

* 1. Используемые ресурсы

В рамках разработанного Метода и имплементированного алгоритма была разработана программа и проведен эксперимент. Для имплементации алгоритма были использованы различные программные технологии.

В качестве предмета для анализа была взята операционная система Linux, установленная на ноутбуке.

Характеристики операционной системы и ноутбука:

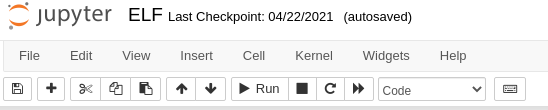
* Operating System: Arch Linux;
* Kernel: Linux 5.12.1-arch1-1;
* Architecture: x86-64;
* CPU model name: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz;
* Hardware Model: HP 250 G5 Notebook PC.

Для разработки программы был использован язык программирования Python 3.9.5. Этот язык был выбран ввиду емкого, но не избыточного синтаксиса, обширного количества библиотек, посвященных анализу данных.

* + 1. Среда разработки Jupyter Notebook

В качестве среды разработки использовалось окружение Jupyter Notebook. Jupyter Notebook представляет собой гибкое веб-приложение, отличающееся от других интегрированных сред разработки возможностью ячеечной интерпретации кода.

Файлы Jupyter Notebook имеют расширение “.ipynb”. Файлы открываются в браузере в интерфейсе окружения, запущенного как отдельное приложение внутри операционной системы. Файлы представляют собой страницу в браузере, состоящую из панели инструментов (см. Рис. 2), ячеек кода, ячеек разметки.

Рисинок 2 — Панель инструментов файлы Jupyter Notebook 

Каждая ячейка с кодом или разметкой представляет собой встроенное окно с возможностью редактирования текста. Ячейки разметки используются для комментариев, пояснений и вставки изображений в файл программы. Ячейки кода используются непосредственно для написания программ(см. Рис. 3). Код программы может содержаться как в одной ячейке, так и в нескольких. Jupyter Notebook для каждого файла использует индивидуальное пространство памяти, что позволяет обращаться к переменным, которые были объявлены в файле, но не были объявлены в ячейке кода.



Рисунок 3 — Ячейки кода и разметки Jupyter Notebook

В программе были использованы следующие языковые библиотеки:

LIEF — библиотека, предназначенная для изъятия информации из исполнительных файлов различных операционных систем, а также для изменения таких файлов.

OS — библиотека, предназначенная для взаимодействия с файловой системой ОС, функционал позволяет считывать названия файлов и директориев, а также обращаться к переменным окружения операционной системы.

lddwrap — небольшая библиотека, реализующая функционал метода “ldd”. Метод “ldd” выводит в консоль все динамические библиотеки, которые использует файл.

pathlib — библиотека, предоставляющая доступ к объектам типа “Path”. Предназначена для манипуляций путями к файлам.

networkx — библиотека, предназначенная для работы с графами и предоставляющая широкий функционал по взаимодействию с ними. В функционал входит программно-математическое описание графов, направленных и ненаправленных, мультиграфов, а также визуализация и импорт графов.

matplotlib — библиотека, используемая для программной отрисовки данных. Это библиотека используется networkx для визуализации графов.

PageRank-HITS-SimRank — пользовательская библиотека, предназначенная для расчета метрик, используя математические алгоритмы PageRank, HITS и SimRank.

* + 1. Средство визуализации графов Gephi

Помимо окружения Jupyter Notebook в работе используется программное средство Gephi, предназначенное для визуализации графов и взаимодействия с ними. Gephi использует файловое расширение GEXF, которое может хранить в себе описание графов, их узлов, граней, свойств.

Gephi обладает широким функционалом для работы с графами. Программа предоставляет три интерфейса для различных манипуляций с графами. Первый представляет собой окно для ручного взаимодействия с графом, он содержит функции настройки вида узлов, граней, наименований узлов и граней, алгоритмы оптимизации размещения графа в декартовой системе координат, а также инструменты фильтрации отображаемых узлов и граней графа (см. Рис. 4).

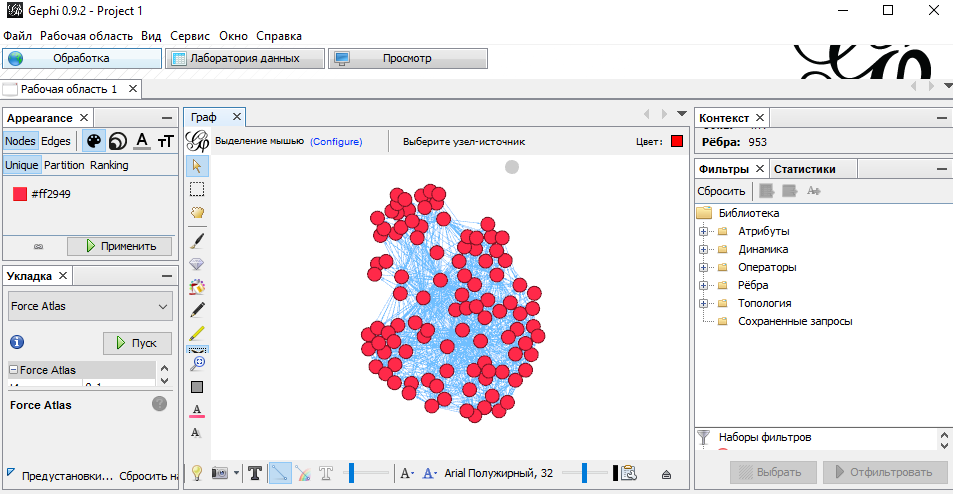


Рисунок 4 — Первый интерфейс Gephi

Второй интерфейс представляет собой граф в виде таблицы, описывающей граф. В таблице две вкладки для граней и их свойств, для узлов и их свойств. Этот интерфейс также предоставляет функционал для работы с таблицей, её строками и столбцами (см. Рис. 5).

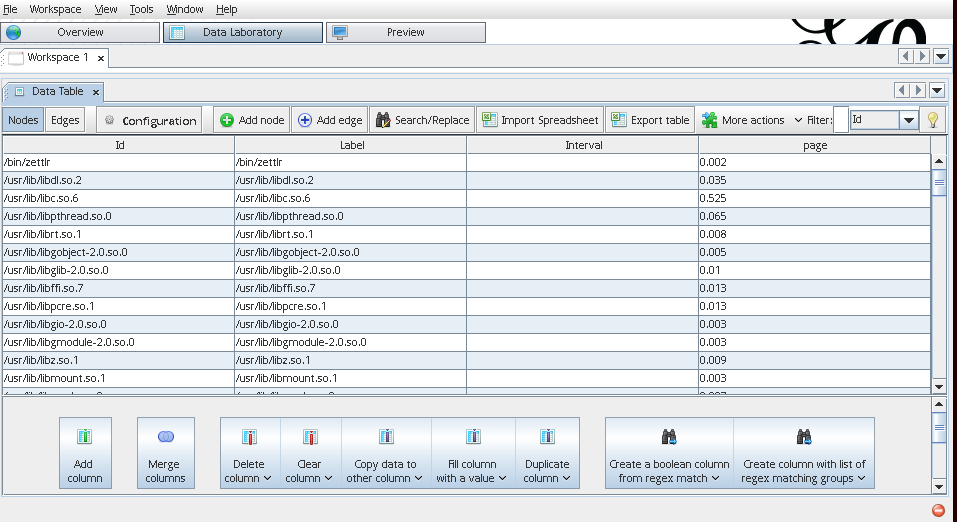


Рисунок 5 — Второй интерфейс Gephi

Третий интерфейс, предпросмотр, предоставляет функционал для настраиваемой визуализации графа и экспорта изображения в различные форматы (см. Рис. 6).

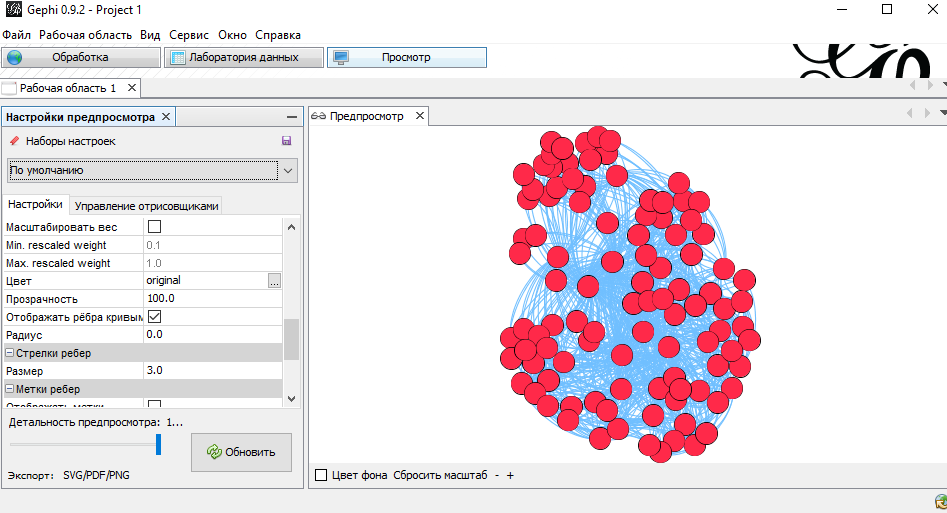


Рисунок 6 — Третий интерфейс Gephi

* 1. Примеры имплементации

Для конкретной имплементации взаимодействия будут производиться с исполнительными программными файлами, хранящимися в директории “/bin”. Для каждого примера будет определена своя область исследования.

Из области исследования изымаются наименования всех программных файлов (см. Рис. 7), с которыми взаимодействует объект исследования. В конкретной имплементации это осуществляется с помощью функции ldd, возвращающую список “shared objects” (далее — SO), используемые программными файлами. Ввиду особенностей системы линукс, все исполнительные файлы по умолчанию используют 2 виртуальных SO:

linux-vdso.so.1 — виртуальная библиотека, обеспечивающая исполнительному файлу быстрый доступ к системным функциям.

libffmpeg.so — кодек, позволяющий преобразование аудио- и видеофайлов в поддерживаемые системой форматы.

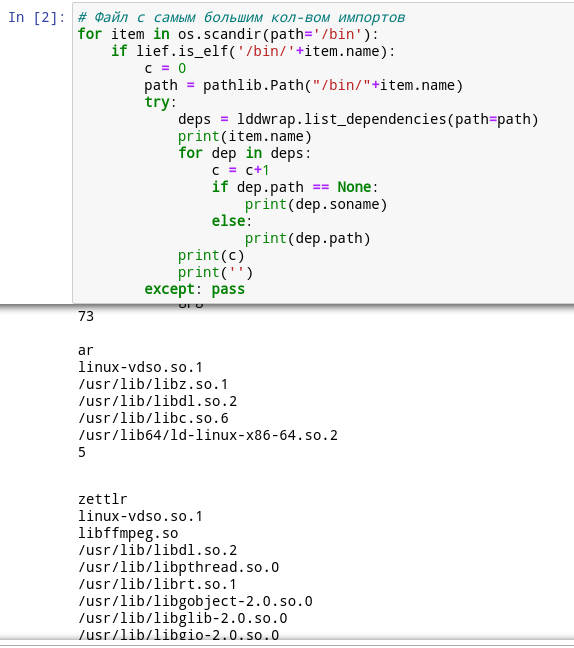


Рисунок 7 — Код сканирования системы и вывода списка импортов

Эти 2 SO следует исключить из списка. Из остальных SO следует также извлечь все импортируемые SO, чтобы определить междуфайловые связи. Все импорты считаются однонаправленной гранью, и из этих взаимодействий формируется список граней.

* + 1. Пример Анализа единственной программы

Объектом изучения является простая программа. Так как программа выбиралась среди всех программ в операционной системе, была выбрана программа с наибольшим количеством взаимодействий. Для этого были просканированы все программные исполнительные файлы директории /bin, из каждого файла были извлечены записи об импортах SO.

В качестве исследуемой программы была выбрана “Zettlr”, эта программа импортирует 100 SO. Для анализа программы из нее извлекаются все импортируемые SO и из них формируется список. Для каждой записи в этом списке также производится изъятие импортов и запись их в память среды разработки.

Далее из списка всех взаимодействий программы с SO и SO друг с другом формируется граф (см. Рис. 8).

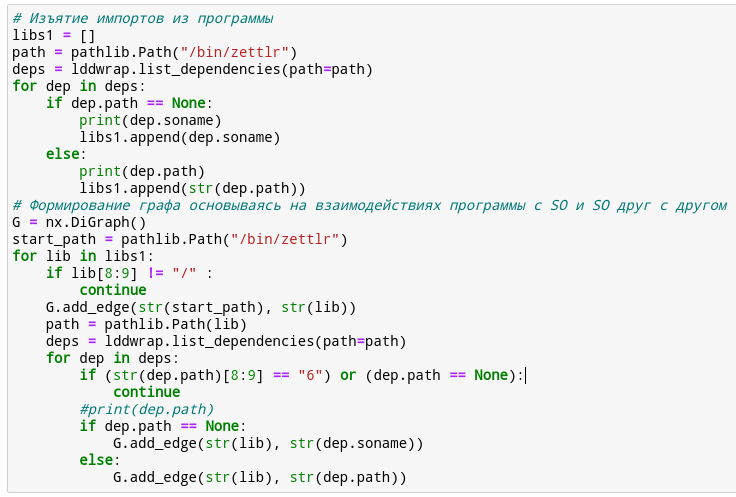


Рисунок 8 — Код извлечения импортов и формирования графа

Затем производиться расчет метрик алгоритмов PageRank и HITS для каждого узла. Результаты расчетов выводятся в виде таблицы и предварительно визуализируются в среде разработки.

На предварительной визуализации явно выделяется красный узел, значение метрики Важности для него выше по сравнению с другими значениям (см. Рис. 9).

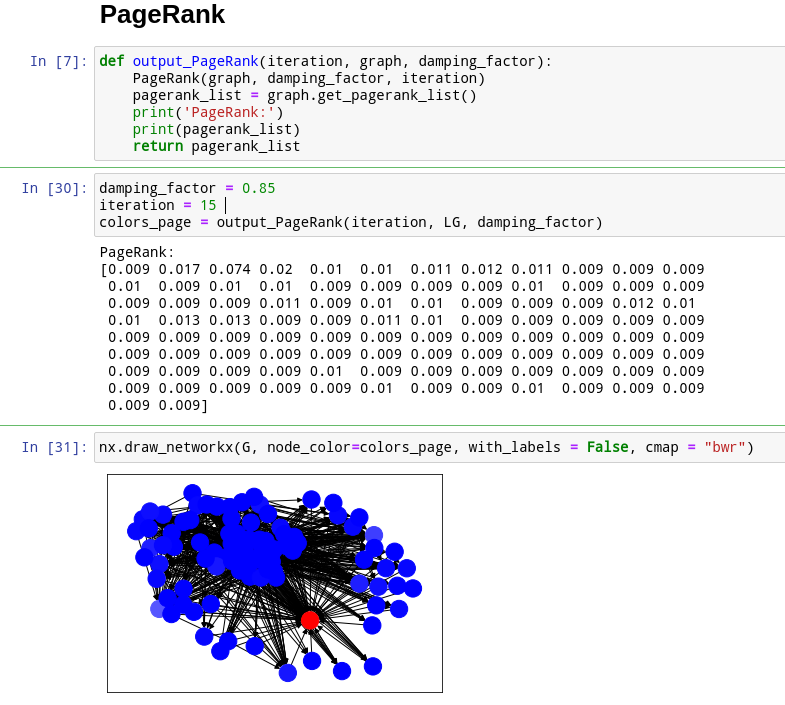


Рисунок 9 — Код расчета метрик PageRank и предварительной визуализации графа

Рассчитываются метрики алгоритма HITS, переходность и авторитет. Результаты выводятся в виде таблицы для каждой метрики и предварительно визуализируются в среде разработки (см. Рис. 10).

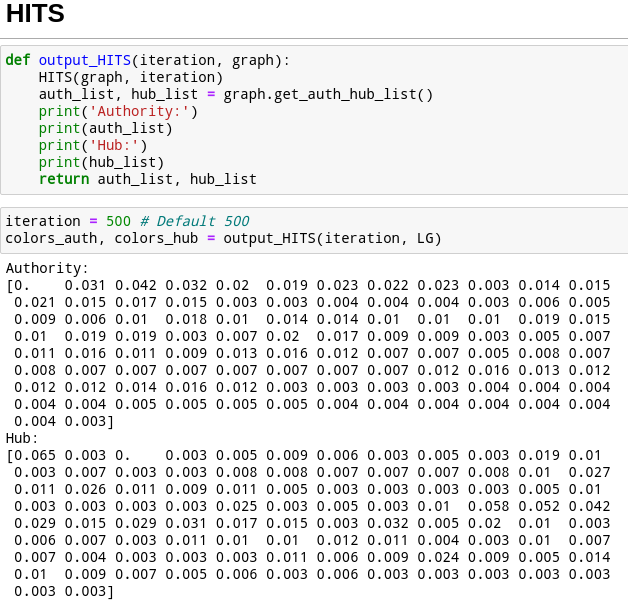
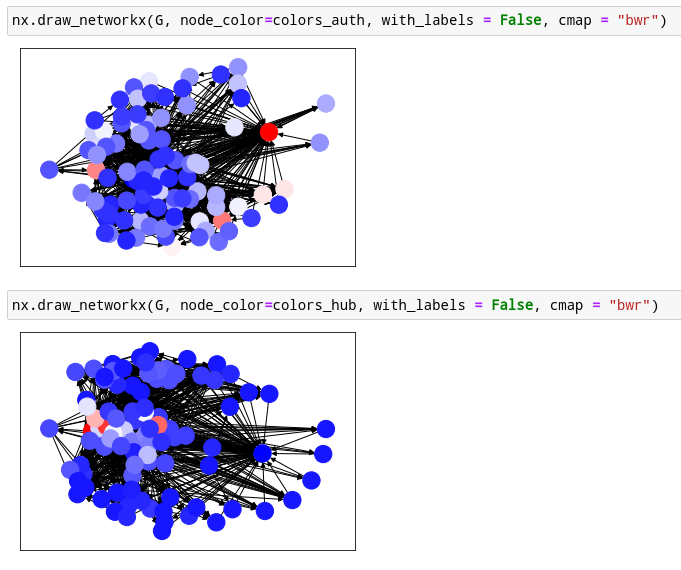
а) б)

Рисунок 10 — Код и вывод расчета метрик алгоритма HITS (а),

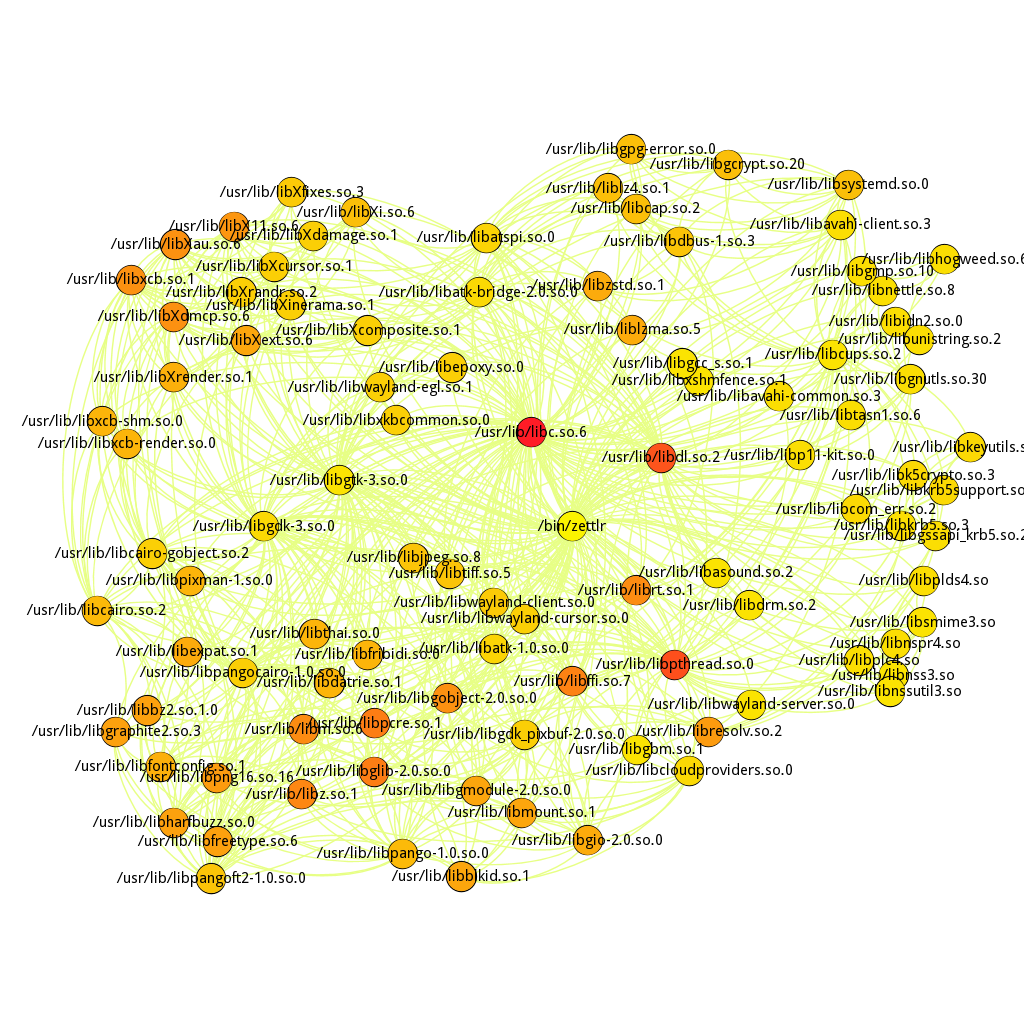
Предварительная визуализация графа метриками HITS (б)

Из предварительной визуализации явно выделяются некоторые красные узлы и кластеры узлов. Их необходимо исследовать более детально, для этого графы со значениями метрик импортируются в формат GEXF (см. Рис. 11).

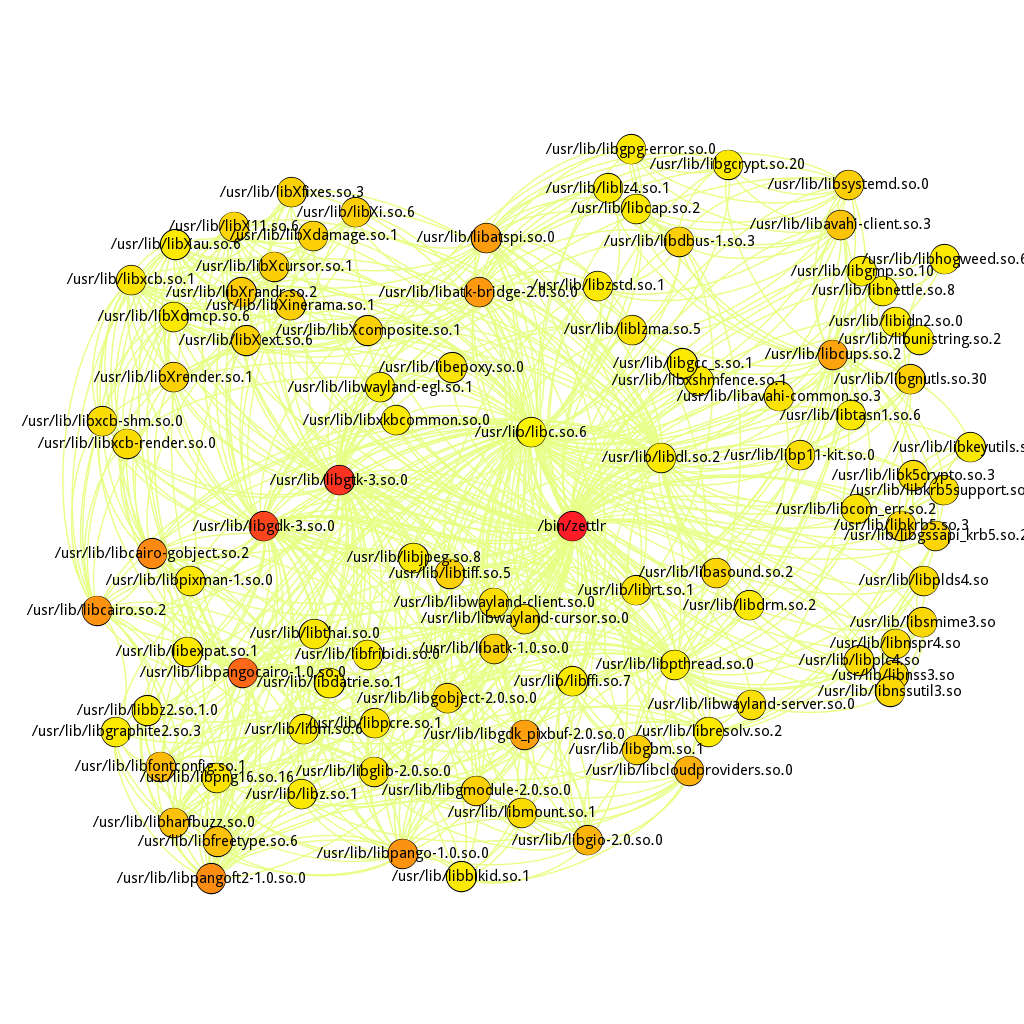


Рисунок 11 — Код импорта графа в формат GEXF

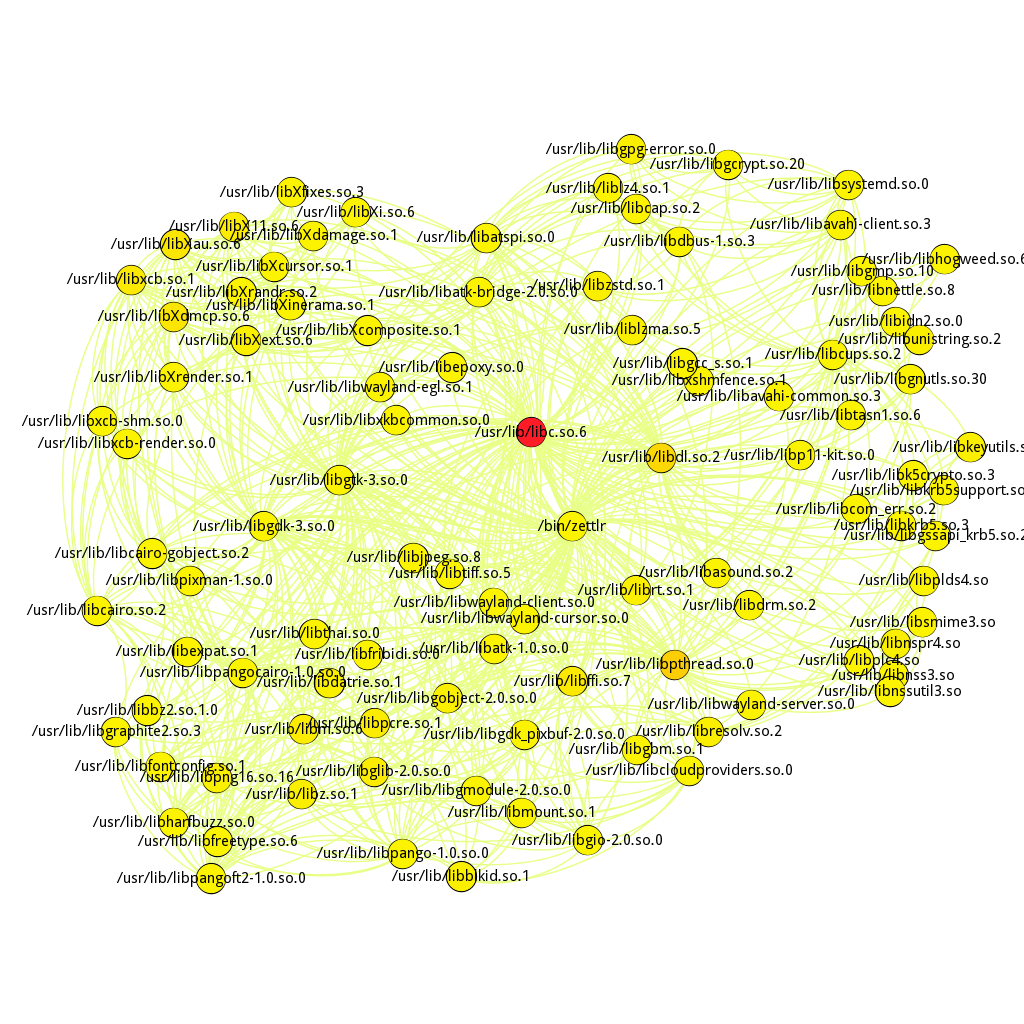
Открыв граф в Gephi его необходимо настроить для читаемой визуализации. В первую очередь применяется алгоритм force atlas. Это алгоритм, симулирующий физическую силу, где грани выступают в качестве пружин. Благодаря созданию такой симуляцию граф самостоятельно перестраивается, достигая состояние равновесия для всех граней. Далее настраиваются цвета узлов и граней для улучшения читаемости графа. Ранжирование цвета соответствует числовому значению метрики, желтый — малое значение, оранжевый — среднее, красное — высокое. На графе также отображены названия узлов (см. Рис. 12).



а)



б)



в)

Рисунок 12 — Визуализация графа с метриками Авторитета (а), Проходимости (б), Важности (в)

* + 1. Пример Анализа функционального модуля подключения к сети

Объектом изучения является модуль подключения к сети. Этот модуль состоит из следующих программ (см. Рис. 13):

* nmcli — NetworkManager;
* dhcpcd — DHCP- и DHCPv6-клиент;
* networkctl — системный даемон для управления сетевыми настройками;
* ping — утилита для проверки качества соединений в сетях на основе TCP/IP;
* ip — интерфейс взаимодействия с сетевыми устройствами;
* ss — утилита по взаимодействию с сокетами.

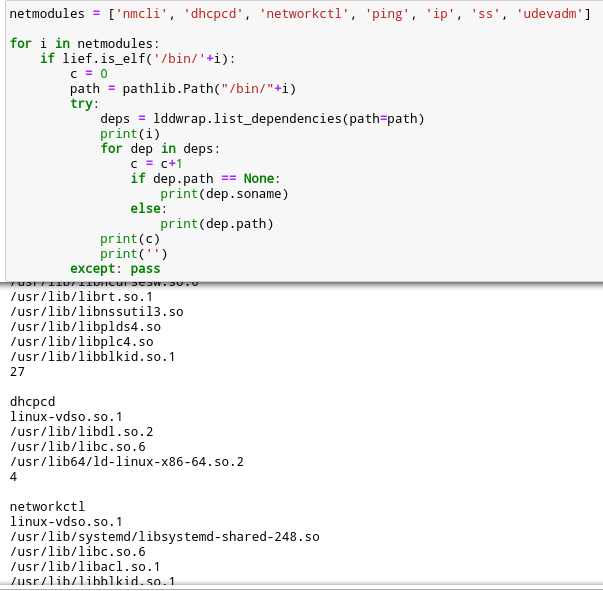


Рисунок 13 — Код сканирования системы и вывода списка импортов

Далее из списка всех взаимодействий программ друг с другом и программ с SO формируется граф (см. Рис. 14).



Рисунок 14 — Код извлечения импортов и формирования графа

Затем производится расчет метрик алгоритмов PageRank и HITS для каждого узла. Результаты расчетов выводятся в виде таблицы и предварительно визуализируются в среде разработки.

На предварительной визуализации явно выделяется красный узел, значение метрики Важности для него выше по сравнению с другими значениям (см. Рис. 15).

Рассчитываются метрики алгоритма HITS, переходность и авторитет. Результаты выводятся в виде таблицы для каждой метрики и предварительно визуализируются в среде разработки (см. Рис. 16).

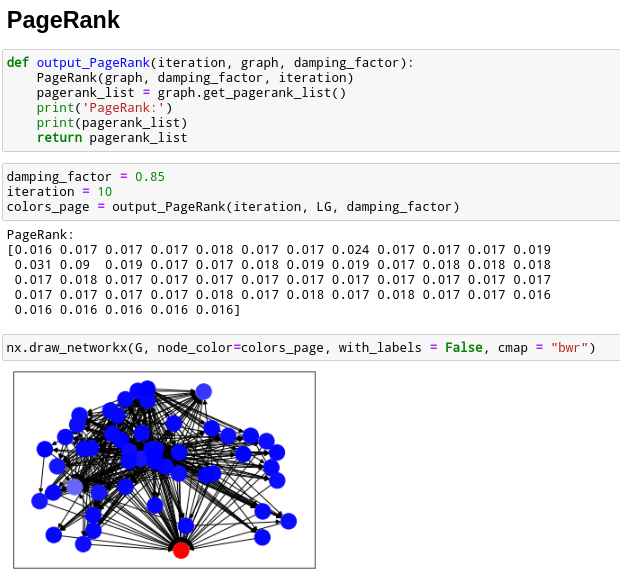


Рисунок 15 — Код расчета метрик PageRank и предварительной визуализации графа

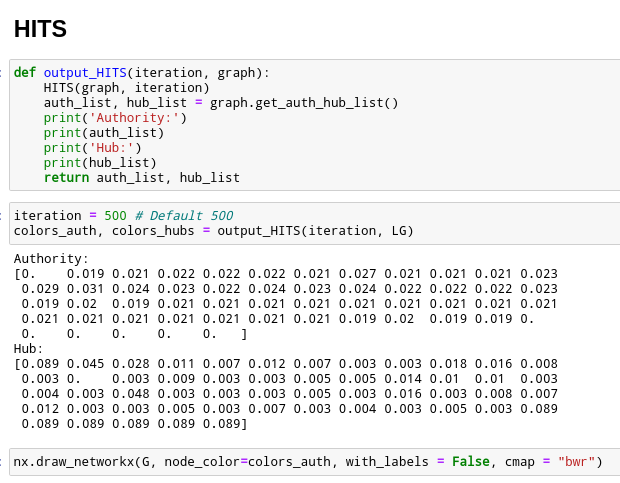
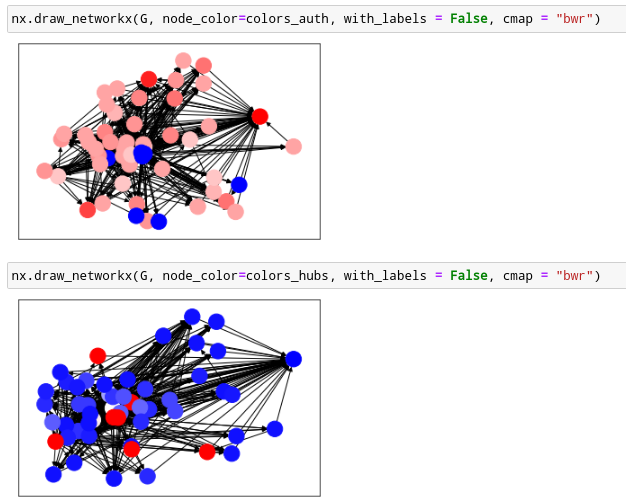
а)б)

Рисунок 16 — Код и вывод расчета метрик алгоритма HITS (а),

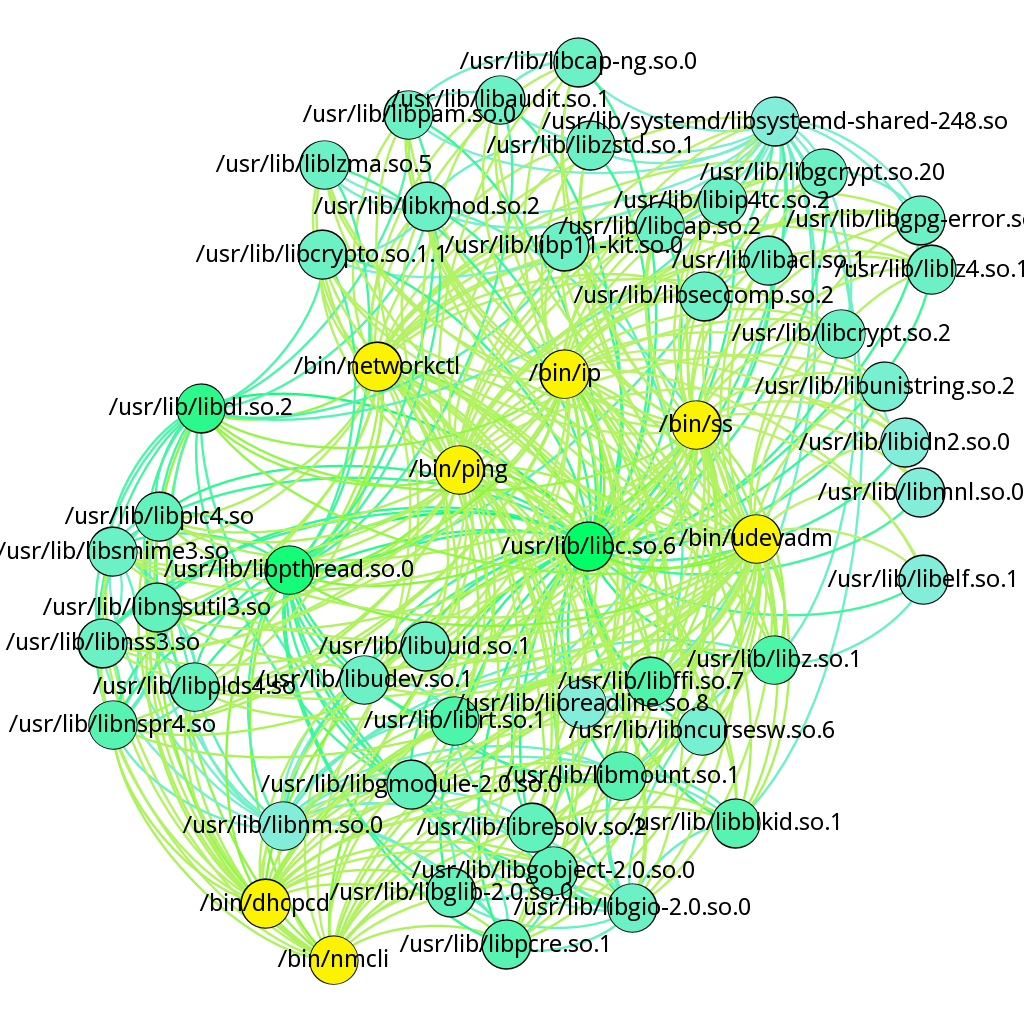
Предварительная визуализация графа метриками HITS (б)

Из предварительной визуализации явно выделяются некоторые красные узлы и кластеры узлов. Их необходимо исследовать более детально, для этого графы со значениями метрик импортируются в формат GEXF (см. Рис. 17).

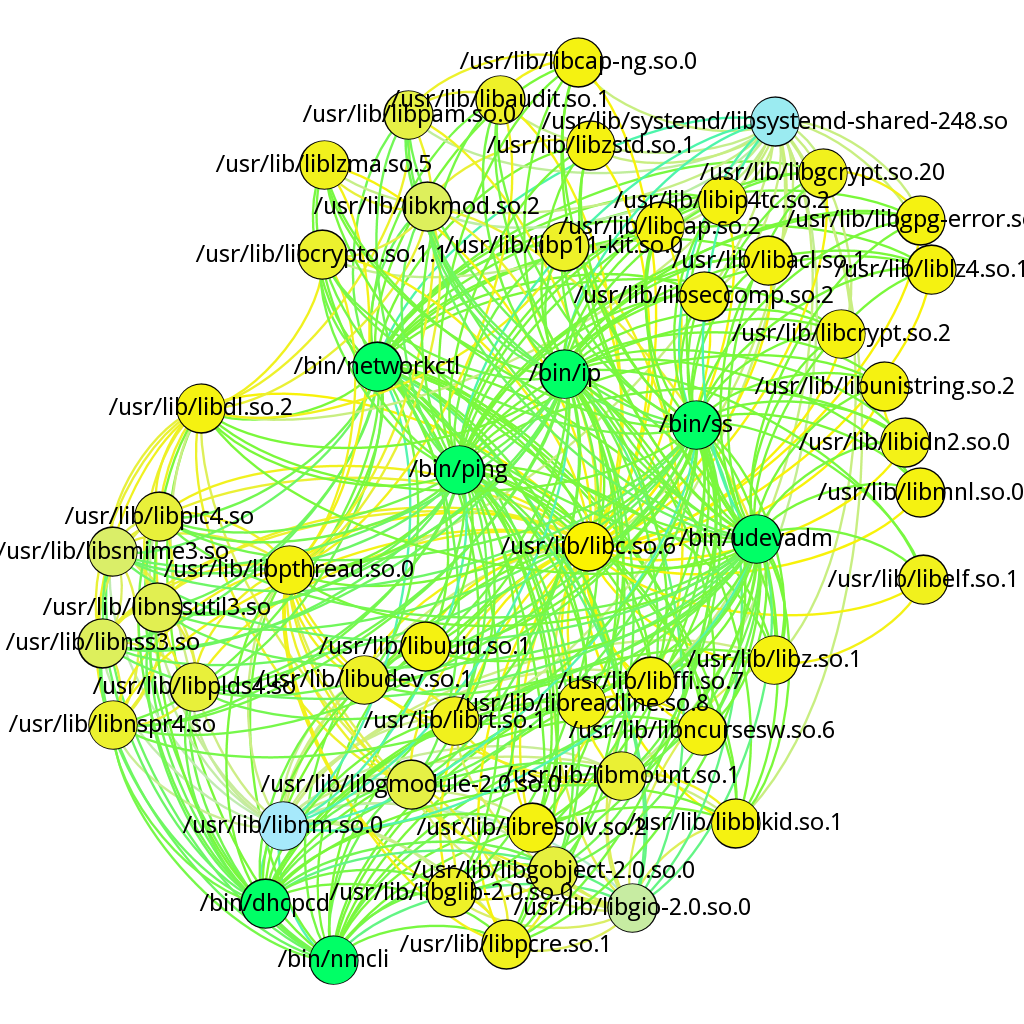


Рисунок 17 — Код импорта графа в формат GEXF

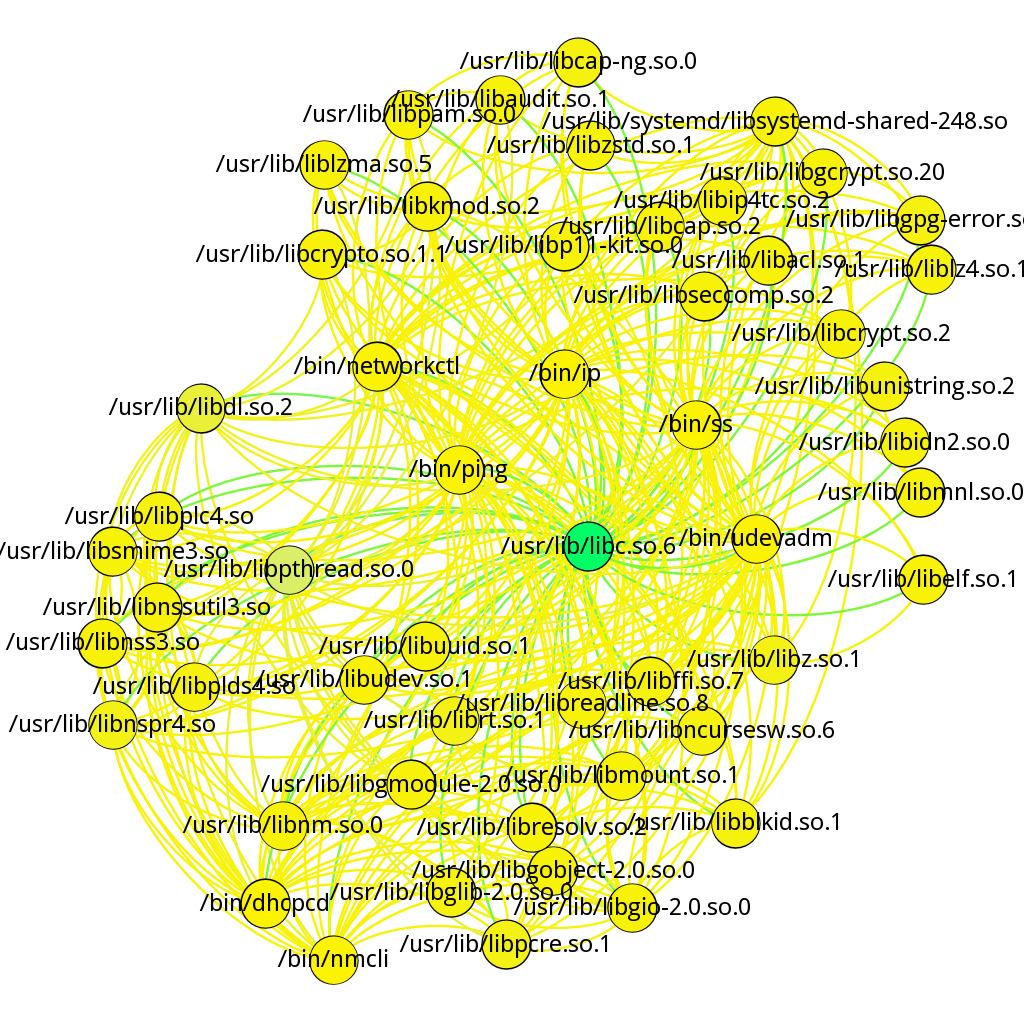
Открыв граф в Gephi, с помощью первого интерфейса программы он настраивается для читаемой визуализации. В первую очередь применяется алгоритм force atlas. Это алгоритм, симулирующий физическую силу, где грани выступают в качестве пружин. Благодаря созданию такой симуляцию граф самостоятельно перестраивается, достигая состояния равновесия для всех граней. Далее настраиваются цвета узлов и граней для улучшения читаемости графа. Ранжирование цвета соответствует числовому значению метрики, желтый — малое значение, голубой — среднее, зеленый — высокое. На графе также отображены названия узлов (см. Рис 18).



а)



б)



в)

Рисунок 18 — Визуализация графа с метрикой Авторитета (а), Проходимости (б), Важность (в)

* 1. Выводы на основе результатов алгоритма

В первую очередь следует отметить, что алгоритм масштабируется как при увеличении количества исследуемых объектов, в частности программ, так и при увеличении количества объектов в целом, а именно числа SO.

Благодаря алгоритму среди файлов возможно выделить файлы с наибольшими значениями метрик.

В случае авторитета такие файлы могут потенциально стать предпочтительной целью для внедрения Временной бомбы, в случае срабатывания которой система мгновенно выйдет из строя ввиду возможности распространения зловредного кода из авторитетного файла.

Файлы с большой проходимостью скорее всего хранят в себе ключевую логику программ, изучение таких файлов позволит понять принципы работы программы.

Важные файлы являются самыми влиятельными в программе или системе, так как при их компрометации возможны нарушения принципов доступности и целостности программы или системы.

1. Моделирование проведения аудита информационной безопасности с использованием Метода
   1. Описание предприятия

Вымышленное предприятие которое будет использовано для оценки эффективности Метода представляет собой фабрику по производству микросхем. Такое предприятие было выбрано по нескольким причинам:

* фабрика выпускает простой электротехнический продукт, который самостоятельно генерирует информацию;
* фабрика выпускает продукт, который является компонентом для сложных электротехнических устройств, например телефонов;
* продукт фабрики представляет интерес для таких групп пользователей, как производители устройств, разработчики и программисты микросхем, конечные пользователи устройств.

Естественным образом информационная экосистема фабрики состоит из функциональных систем.

Социально-трудовая система выполняет функции найма работников, регулирования их деятельности на месте, принятия решений.

Юридическая система ведет налоговую отчетность перед государством, документацию деятельности внутри предприятия и разрешает юридические проблемы фабрики.

Информационная система тесно взаимодействует с двумя предыдущими системами. Она выполняет функции хранения всей информации на фабрике, сбора и обработки информации, формирования решений, автоматизации процессов.

В информационную систему данные поступают из микросхем внутри устройств. Цифровая информация о работоспособности микросхем, например логи, отправляется на сервера фабрики. Другим источником информации являются производители устройств, которые предоставляют технические данные о взаимодействии микросхем с другими частями устройств и обратную связь от пользователей устройств.

Также информация поступает от разработчиков и программистов микросхем, а также от непосредственно пользователей устройств.

Для обработки всей информации ИС имеет следующие функциональные модули:

* Модуль организации поставок;
* Модуль централизованного хранения и обработки цифровой информации;
* Модуль системы контроля доступом;
* Модуль управления технологическими процессами;
* Модуль электронного документооборота.

Производственный процесс на фабрике выглядит следующим образом:

* Закупаются компоненты и необходимые для микросхем материалы;
* Автоматизированное оборудование под управлением сотрудников собирает микросхемы;
* Микросхемы складируются и отсылаются производителям устройств.

На основе собранной и обработанной информации ИС и аналитики формируют решение для руководства фабрики относительно изменений в производственный процесс.

При создании информационной системы фабрики велась частичная документация.

* 1. Оценка Метода

Естественный образом, в жизненном цикле такого предприятия как фабрика микросхем возникают ситуации, требующие аудита. Как было описано в начале исследования, аудиты имеют общую структуру, однако цели аудитов могут различаться. Далее в рамках исследования необходимо привести несколько искусственных сценариев из жизни фабрики, в которых требуется проведение аудита с использованием Метода.

* + 1. Сценарий I — Утечка данных по технической причине

В результате инцидента информационной безопасности, а именно критической ошибки в информационной системе, данные были выложены в интернет. Единственная информация об инциденте — это сами скомпрометированные единицы информации, и что утечка была вызвана дефектом системы технического характера. Для предотвращения подобных инцидентов фабрика нанимает компанию для проведения внешнего аудита и выявления причины утечки. До выяснения причин работа фабрики приостановлена во избежание повторной утечки.

* + - 1. Подготовка проведения аудита

Основной целью аудита является выяснение причин утечки. Аудит должен быть проведен в краткие сроки. Аудитору предоставляется доступ ко всей информации об информационной экосистеме, к электронной почте сотрудников, за исключением личных данных сотрудников и чувствительной информации о производственном процессе.

* + - 1. Сбор информации об ИЭ

В первую очередь изучаются скомпрометированные данные, выясняется, в какой части информационной системы они циркулируют. Таким образом ограничивается круг поиска утечки. В данном случае утерянные данные используются в модуле хранения. Следовательно, изучение модуля обработки и хранения данных является первостепенной задачей, причем как с точки зрения социально-трудовой системы, так и с точки зрения информационной системы. Из социально-трудовой системы изымается переписка сотрудников работающих с модулем, а также записи о них и их обязанностях из отдела кадров. Из информационной системы формируется список всех программных файлов, работающих в рамках модуля для каждого компьютера и сервера в этом модуле.

* + - 1. Анализ данных аудита

Собранная и предварительно структурированная информация об экосистеме анализируется. В целях сохранения времени некоторые действия при анализе приоритизируются. В первую очередь изучается инфраструктура системы относительно потока данных. Обращение к документации в этом случае может серьезно помочь, однако написанное в документации все равно следует сверить с действительностью.

Удобным способом визуализации потока данных является Метод, в первую очередь результатом работы Метода на уровне модуля обработки информации является граф потока данных. Более того, на графе будут выделены программные файлы с наибольшим количеством взаимодействий с единицами информации, с наибольшей метрикой Проходимости. Вероятнее всего в этих файлах будет код, связанный со скомпрометированными данными. Соответственно их изучение является первостепенной задачей. Далее используются классические методы анализа кода для выявления дефекта системы, вызвавшего утечку.

* + - 1. Оценка рисков и выработка рекомендаций

После выявления дефекта проводится оценка возможности повторной утечки, а также других возможностей непредусмотренной эксплуатации системы.

На основе оценки рисков вырабатываются рекомендации по преобразованию системы, в данном случае модуля обработки информации, таким образом, чтобы дефект был устранен. В зависимости от дефекта для правильного функционирования системы могут потребоваться изменения от минимальной реконфигурации базы данных до полноценной реструктуризации модуля с изменением путей циркуляции данных.

* + - 1. Подготовление отчета

В отчете собираются все знания, полученные в ходе исследования и анализа. Отчет предоставляется лицам, принимающим решения на фабрике.

* + 1. Сценарий II — Утечка данных из-за инсайдера

В результате инцидента информационной безопасности, а именно неправомерных действий сотрудника в системе, чувствительные данные были выложены в интернет. Единственная информация которая известна об инциденте — это сами чувствительные данные. В целях предотвращения подобных инцидентов фабрика нанимает компанию для проведения внешнего аудита и выявления причины утечки. До выяснения причин работа фабрики приостановлена во избежание повторной утечки.

* + - 1. Подготовка проведения аудита

Основной целью аудита является выяснение причин утечки. Аудит должен быть проведен в краткие сроки. Аудитору предоставляется доступ ко всей информации об информационной экосистеме, к электронной почте сотрудников, за исключением личных данных сотрудников и чувствительной информации о производственном процессе.

* + - 1. Сбор информации об ИЭ

В первую очередь изучаются скомпрометированные данные, выясняется в какой части информационной системы они циркулируют. Таким образом ограничивается круг поиска утечки. В данном случае чувствительные данные представляют собой планы на закупку материалов, которые используются в модуле организации поставок. Следовательно, изучение модуля организации поставок является первостепенной задачей, причем как с точки зрения социально-трудовой системы, так и с точки зрения информационной системы. Из социально-трудовой системы изымается переписка сотрудников, работающих с модулем, записи о них и их обязанностях из отдела кадров. Также составляется список всех людей в полномочия которых входила работа с чувствительной информацией. Из информационной системы формируется список всех программных файлов, работающих в рамках модуля для каждого компьютера и сервера в этом модуле.

* + - 1. Анализ данных аудита

Собранная и предварительно структурированная информация об экосистеме анализируется. В целях сохранения времени некоторые действия при анализе приоритизируются. В первую очередь ведутся интервью с сотрудниками. Однако, чтобы не допрашивать каждого сотрудника, следует предварительно изучить информацию, доступную о сотрудниках. Чаще всего злонамеренными инсайдерами являются сотрудники неудовлетворенные своим положением, материальным или социальным. Это косвенно сказывается на их социальных связях, иными словами злонамеренные инсайдеры реже обычного общаются с коллегами.

Используя Метод на уровне социально-трудовой системы можно выяснить кто из сотрудников не популярен в коллективе. Сопоставив эту информацию с таблицей полномочий каждого сотрудника, аудитор сужает круг поиска инсайдера.

С другой стороны инсайдером может оказаться халатный сотрудник, случайно или намеренно не последовавший определенной процедуре правил информационной безопасности. В таком случае следует изучить информационную систему, используя Метод по принципу описанному в прошлом сценарии. Далее используются классические методы обнаружения инсайдера.

* + - 1. Оценка рисков и выработка рекомендаций

В результате анализа аудитор может выяснить сотрудник превысивший свои полномочия. Причина его поступка может заключаться как в его собственных мотивах, так и являться следствием неправильного устройства социально-трудовой части экосистемы. Это может выражаться в несправедливом распределении заработной платы, избыточной регуляции работы или других негативно влияющих на условия работы факторах. В случае утечки по неосторожности особое внимание следует уделить правилам, их необходимости, и полномочиям сотрудников. Возможно, у сотрудников, мало работающих с данными, необоснованно много полномочий. Во всех случаях оцениваются риски и составляются рекомендации к каждой из частей экосистемы.

* + - 1. Подготовление отчета

В отчете собираются все знания, полученные в ходе исследования и анализа. Отчет предоставляется лицам, принимающим решения на фабрике.

* + 1. Сценарий III Добавление нового модуля в ИЭ

Фабрика запланировала расширение производства для создания розничного сектора продаж микросхем. Для этого компания создала специальный функциональный модуль для работы с розничными продажами и поставками. Чтобы внедрить модуль в экосистему необходимо провести аудит информационной безопасности.

* + - 1. Подготовка проведения аудита

Основной целью аудита является проверка функционального модуля. Аудитору предоставляется доступ к документации модуля, обязанностям и полномочиям сотрудников работающих с новым модулем, программные файлы нового модуля.

* + - 1. Сбор информации о функциональном модуле

Большие производства ввиду своих специализаций вынуждены использовать проприетарное программное обеспечение. Нередко информационная система является совокупностью нескольких различных фреймворков. Так как главной целью совокупности технологий является работоспособность, то глубокое продумывание организации системы и безопасность часто приносятся в жертву. Поэтому критически важно отслеживать непредусмотренные эксплуатации системы. Присоединение модуля является созданием новой совокупности технологий. О функциональном модуле собирается информация из всех частей экосистемы.

* + - 1. Анализ данных аудита

В целях обнаружения потенциальных критических уязвимых мест в модуле следует использовать Метод на уровне функционального модуля, включая в анализ все пограничные взаимодействия с другими модулями и их программными файлами. В результате на представленном графе файлы будут размечены по Важности, Авторитетности и Проходимости.

Самые Важные и Авторитетные файлы будут соединены друг с другом и будут иметь наибольшее количество исходящих связей, как следствие эти файлы в большей степени будут отвечать за правильное функционирование модуля. Соответственно в случае нарушения принципов информационной безопасности для этих файлов, модуль вероятнее всего будет неработоспособным.

С другой стороны проходимые файлы, которые мало влияют на систему, могут содержать в своем коде возможности для непредусмотренных действий, исследование таких возможностей позволит обнаружить уязвимости связанные с Временными бомбами.

Далее модуль исследуется классическими методами статического и динамического анализа.

* + - 1. Оценка рисков и выработка рекомендаций

При оценке рисков в первую очередь следует уделить внимание возможности выхода из строя модуля, затем непредусмотренным взаимодействиям модуля с остальной информационной системой. Также необходимо предусмотреть и халатное отношение к правилам ИБ на фабрике.

При выработке рекомендаций для функционального модуля необходимо учесть не только риски каждой системы индивидуально, но и взаимодействия систем друг с другом.

* + - 1. Подготовление отчета

В отчете собираются все знания, полученные в ходе исследования и анализа. Отчет предоставляется лицам, принимающим решения на фабрике.

* + 1. Сценарий IV Первичный аудит

После окончания строительства и установки оборудования перед началом работы фабричная информационная экосистема должна пройти аудит информационной безопасности, и, оценив риски, установить правила и оборудование предотвращающие инциденты информационной безопасности.

* + - 1. Подготовка проведения аудита

Основной целью аудита является изучение всей информационной экосистемы на предмет уязвимостей. Аудитору предоставляется доступ к документации ИЭ, обязанностям и полномочиям всех сотрудников, программные файлы всех модулей ИЭ.

* + - 1. Сбор информации об ИЭ

Составление полноценной детальной картины структуры информационной экосистемы — комплексный процесс. Изучение документации, при её наличии, занимает первостепенной значение, так как дает общее представление об ИЭ. Не менее необходимо понимание бизнес процессов на фабрике и роли всех частей ИЭ в них. Для непосредственного понимания информационной части ИЭ обязательны тестовые логи, программные файлы. Социально-трудовая часть ИЭ формируется по мере запуска самой фабрики, поэтому на этапе первичного аудита сформирована лишь частично.

* + - 1. Анализ данных аудита

Анализ всей ИЭ фабрики многоуровневый процесс, ключевая для информационной безопасности информационная система должна быть исследована с разных сторон.

Для первичного анализа сети подойдет сетевой уровень Метода. В результате его применения выделяются самые Важные и Авторитетные компьютеры. Организация доступа к таким компьютерам является первостепенной задачей социально-трудовой системы. Вследствие компрометации таких компьютеров фабрика может понести критические убытки. Далее сеть анализируется и исследуется с помощью классических статических и динамических методов анализа.

Глубинный анализ кода информационной системы это ресурсозатратный процесс. В целях его оптимизации необходимо упорядочить последовательность изучения функциональных модулей. Порядок изучения модулей можно установить применив Метод на уровне операционных систем, где программные файлы всех модулей, объединенные с помощью методов уменьшения размерности, и кластеризованные в соответствии с модулями которым они принадлежат, ранжированы по Важности, Проходимости и Авторитетности. Модули с наибольшим количеством Проходимых узлов приоритетны для изучения. Изучение этих модулей в первую очередь даст понимание логики модуля и системы в общем. Как следствие уязвимости будут обнаружены быстрее. Также необходимо обратить внимание на наименее Важные и Авторитетные узлы графа, они могут быть эксплуатированы для закладки временных бомб при спланированных атаках на экосистему. Далее для изучения информационной системы используются классические методы статического и динамического анализа.

Также необходимо изучить пути данных, и, хоть, для этой задачи лучше всего подходит динамический анализ, применение Метода на уровне модуля обработки информации позволит получить визуальное представление путей данных, основанное непосредственно на коде из системы.

Далее ИЭ исследуется классическими методами статического и динамического анализа.

* + - 1. Оценка рисков и выработка рекомендаций

Оценка рисков для всей ИЭ нетривиальная задача, необходимо учесть все собранные знания из исследования и учесть все потенциальные уязвимости, которые могут быть эксплуатированы. Оценить самые важные активы фабрики и предположить какие угрозы могут быть самыми критичными.

В соответствии с вычисленными рисками составляются рекомендации для защиты всей информационной экосистемы.

* + - 1. Подготовление отчета

В отчете собираются все знания, полученные в ходе исследования и анализа. Отчет предоставляется лицам, принимающим решения на фабрике.

ЗАКЛЮЧЕНИЕ

Основной целью работы была частичная оптимизация статического анализа информационных экосистем. Основной проблемой статического анализа информационных систем были не масштабируемые методы. Проблема была решена разработкой интеллектуального метода позволяющего оптимизировать процесс изучения информационных экосистем на различных уровнях, в частности уровни информационных систем, уровень потока данных, уровень социально-трудовой системы.

Опираясь на Метод была разработана программная имплементация доказывающая масштабируемость и функциональность Метода. Дополнительно были приведены реалистичные сценарии аудитов показывающая возможное применение Метода практически для всех целей аудита.

СПИСОК ЛИТЕРАТУРЫ

1. Интеллектуальный метод визуализации взаимодействий программ в интересах аудита информационной безопасности операционной системы Буйневич М.В., Ганов Г.А., Израилов К.Е. // Информатизация и связь. 2020. № 4. С. 67-74.
2. Аналитическое моделирование работы программного кода с уязвимостями Буйневич М.В., Израилов К.Е. // Вопросы кибербезопасности. 2020. № 3 (37). С. 2-12.
3. Способ визуализации модулей системы обеспечения информационной безопасности Буйневич М.В., Покусов В.В., Израилов К.Е. // Научно-аналитический журнал «Вестник Санкт-Петербургского университета Государственной противопожарной службы МЧС России». 2018. № 3. С. 81-91.
4. Информационные объекты в системе обеспечения информационной безопасности Израилов К.Е., Покусов В.В., Столярова Е.С. // Теоретические и прикладные вопросы комплексной безопасности: материалы I Международной научно-практической конференции. Петровская академия наук и искусств (Санкт-Петербург, 2018). 2018. С. 166-169.
5. Программа визуализации процесса кластеризации многомерных точек для учебных целей Израилов К.Е., Красов А.В. // Свидетельство о регистрации программы для ЭВМ № 2020617704 от 10.07.2020
6. Программа для графовой визуализации последовательности одношагоных сетевых атак нарушителя в режиме реального времени Израилов К.Е., Курта П.А. // Свидетельство о регистрации программы для ЭВМ № 2020662669 от 16.10.2020. Заявка № 2020661665 от 05.10.2020
7. Сравнительный анализ моделей угроз информационной безопасности в интересах применимости для многоэтапных схем атак Бирюков А.А., Израилов К.Е. // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО-2017): сборник научных статей VI Международной научно-технической и научно-методической конференции (Санкт-Петербург, 2017). 2017. С. 108-112.
8. Архитектурные уязвимости программного обеспечения Израилов К.Е. // Шестой научный конгресс студентов и аспирантов Инжэкон-2013: тезисы докладов Научно-практической конференции факультета права и экономической безопасности «Актуальные проблемы государства и права» (Санкт-Петербург, 2013). 2013. С. 35.
9. Программа для визуализации динамического изменения графов, заданных в формате Dot, в режиме реального времени Израилов К.Е., Буйневич М.В., Красов А.В., Хорошенко С.В. // Свидетельство о государственной регистрации программы для ЭВМ № 2019667695 от 17.12.2019.

Методика выявления уязвимостей уязвимостей и недекларированных возможностей в программном обеспечении <http://new.groteck.ru/images/catalog/70840/e75c72a254fcc880fa65657fdb144063.pdf>

Гуд выводы по стат анализу ([О статическом анализе начистоту / Блог компании Ростелеком-Солар / Хабр](https://habr.com/ru/company/solarsecurity/blog/439286/))

Аудит по простому ([Как правильно проводить аудит внутренней информационной безопасности?](https://www.staffcop.ru/blog/pravila-audita-vnutrennej-informatsionnoj-bezopasnosti))

[Автоматическое тестирование программ / Хабр](https://habr.com/ru/post/128503/)

[Динамический анализ программ с целью поиска ошибок и уязвимостей при](https://www.ispras.ru/proceedings/docs/2014/26/1/isp_26_2014_1_375.pdf)

https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-2019-2020/#id3

https://www.ptsecurity.com/ru-ru/research/analytics/

https://www.angarapro.ru/reports\_file/%D0%9E%D1%82%D1%87%D0%B5%D1%82%20%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%20ACRC%20%D0%B7%D0%B0%20I%20%D0%BF%D0%BE%D0%BB%D1%83%D0%B3%D0%BE%D0%B4%D0%B8%D0%B5%202020.pdf

https://arxiv.org/abs/1404

<https://www.google.com/url?q=https://towardsdatascience.com/hits-algorithm-link-analysis-explanation-and-python-implementation-61f0762fd7cf&sa=D&source=editors&ust=1621078253564000&usg=AOvVaw2V28s5aPyQpOzTRaaW-Kgy>

https://towardsdatascience.com/pagerank-3c568a7d2332

http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

https://www.google.com/url?q=https://habr.com/ru/company/ods/blog/464715/&sa=D&source=editors&ust=1621083364405000&usg=AOvVaw1jKbz9ePWlOuQ3ZdUM2x\_B

<https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B9%D0%BF%D1%81%D0%BA%D0%B2%D0%BE%D1%82%D1%82%D0%B8%D0%BD%D0%B3>

<https://ru.wikipedia.org/wiki/DLL-%D0%B8%D0%BD%D1%8A%D0%B5%D0%BA%D1%86%D0%B8%D1%8F>

Приложение А

import lief # Утилка для работы с ELFами

import os # Парсинг папок

import lddwrap # Парсинг ELF библиотек

import pathlib # для lddwrap

import networkx as nx # Графы

import matplotlib.pyplot as plt

from pathlib import Path

from src.Graph import Graph # Локальные графы для анализа

from src.PageRank import PageRank # Алгоритм PageRank

from src.HITS import HITS # Алгоритм HITS

In [5]:

# Файл с самым большим кол-вом импортов

for item in os.scandir(path='/bin'):

if lief.is\_elf('/bin/'+item.name):

c = 0

path = pathlib.Path("/bin/"+item.name)

try:

deps = lddwrap.list\_dependencies(path=path)

print(item.name)

for dep in deps:

c = c+1

#if dep.path == None:

#print(dep.soname)

#else:

#print(dep.path)

print(c)

except: pass

**Формирование графов, NX и анализируемого графа**

In [5]:

# Изъятие импортов из программы

libs = []

path = pathlib.Path("/bin/zettlr")

deps = lddwrap.list\_dependencies(path=path)

for dep in deps:

if dep.path == None:

libs.append(dep.soname)

else:

libs.append(str(dep.path))

# Формирование графа основываясь на взаимодействиях программы с SO и SO друг с другом

G = nx.DiGraph()

start\_path = pathlib.Path("/bin/zettlr")

for lib in libs:

if lib[8:9] != "/" :

continue

G.add\_edge(str(start\_path), str(lib))

path = pathlib.Path(lib)

deps = lddwrap.list\_dependencies(path=path)

for dep in deps:

if (str(dep.path)[8:9] == "6") or (dep.path == None):

continue

#print(dep.path)

if dep.path == None:

G.add\_edge(str(lib), str(dep.soname))

else:

G.add\_edge(str(lib), str(dep.path))

In [6]:

LG = Graph()

start\_path = pathlib.Path("/bin/zettlr")

for lib in libs:

if lib[8:9] != "/" : # Отсеивание либ не из /usr/lib/

continue

LG.add\_edge(str(start\_path), str(lib))

path = pathlib.Path(lib)

deps = lddwrap.list\_dependencies(path=path)

for dep in deps:

if (str(dep.path)[8:9] == "6") or (dep.path == None): # Отсеивание либ не из /usr/lib/

continue

if dep.path == None:

LG.add\_edge(str(lib), str(dep.soname))

else:

LG.add\_edge(str(lib), str(dep.path))

**PageRank**

In [21]:

def output\_PageRank(iteration, graph, damping\_factor):

PageRank(graph, damping\_factor, iteration)

pagerank\_list = graph.get\_pagerank\_list()

print('PageRank:')

print(pagerank\_list)

return pagerank\_list

In [30]:

damping\_factor = 0.85

iteration = 15

colors\_page = output\_PageRank(iteration, LG, damping\_factor)

PageRank:

[0.009 0.017 0.074 0.02 0.01 0.01 0.011 0.012 0.011 0.009 0.009 0.009

0.01 0.009 0.01 0.01 0.009 0.009 0.009 0.009 0.01 0.009 0.009 0.009

0.009 0.009 0.009 0.011 0.009 0.01 0.01 0.009 0.009 0.009 0.012 0.01

0.01 0.013 0.013 0.009 0.009 0.011 0.01 0.009 0.009 0.009 0.009 0.009

0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009

0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009 0.009

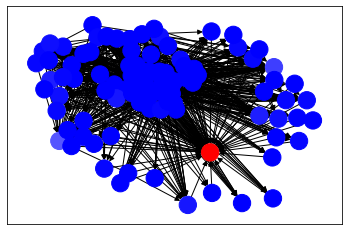
0.009 0.009 0.009 0.009 0.01 0.009 0.009 0.009 0.009 0.009 0.009 0.009

0.009 0.009 0.009 0.009 0.009 0.01 0.009 0.009 0.01 0.009 0.009 0.009

0.009 0.009]

In [31]:

nx.draw\_networkx(G, node\_color=colors\_page, with\_labels = False, cmap = "bwr")



**HITS**

In [22]:

def output\_HITS(iteration, graph):

HITS(graph, iteration)

auth\_list, hub\_list = graph.get\_auth\_hub\_list()

print('Authority:')

print(auth\_list)

print('Hub:')

print(hub\_list)

return auth\_list, hub\_list

In [40]:

iteration = 500 # Default 500

colors\_auth, colors\_hubs = output\_HITS(iteration, LG)

Authority:

[0. 0.031 0.042 0.032 0.02 0.019 0.023 0.022 0.023 0.003 0.014 0.015

0.021 0.015 0.017 0.015 0.003 0.003 0.004 0.004 0.004 0.003 0.006 0.005

0.009 0.006 0.01 0.018 0.01 0.014 0.014 0.01 0.01 0.01 0.019 0.015

0.01 0.019 0.019 0.003 0.007 0.02 0.017 0.009 0.009 0.003 0.005 0.007

0.011 0.016 0.011 0.009 0.013 0.016 0.012 0.007 0.007 0.005 0.008 0.007

0.008 0.007 0.007 0.007 0.007 0.007 0.007 0.007 0.012 0.016 0.013 0.012

0.012 0.012 0.014 0.016 0.012 0.003 0.003 0.003 0.003 0.004 0.004 0.004

0.004 0.004 0.005 0.005 0.005 0.005 0.004 0.004 0.004 0.004 0.004 0.004

0.004 0.003]

Hub:

[0.065 0.003 0. 0.003 0.005 0.009 0.006 0.003 0.005 0.003 0.019 0.01

0.003 0.007 0.003 0.003 0.008 0.008 0.007 0.007 0.007 0.008 0.01 0.027

0.011 0.026 0.011 0.009 0.011 0.005 0.003 0.003 0.003 0.003 0.005 0.01

0.003 0.003 0.003 0.003 0.025 0.003 0.005 0.003 0.01 0.058 0.052 0.042

0.029 0.015 0.029 0.031 0.017 0.015 0.003 0.032 0.005 0.02 0.01 0.003

0.006 0.007 0.003 0.011 0.01 0.01 0.012 0.011 0.004 0.003 0.01 0.007

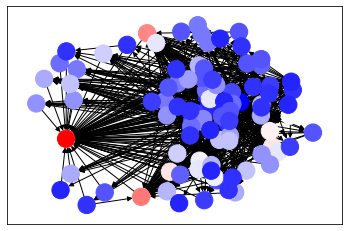
0.007 0.004 0.003 0.003 0.003 0.011 0.006 0.009 0.024 0.009 0.005 0.014

0.01 0.009 0.007 0.005 0.006 0.003 0.006 0.003 0.003 0.003 0.003 0.003

0.003 0.003]

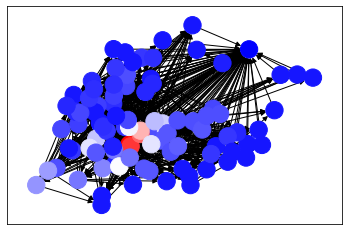
In [38]:

nx.draw\_networkx(G, node\_color=colors\_auth, with\_labels = False, cmap = "bwr")



In [39]:

nx.draw\_networkx(G, node\_color=colors\_hub, with\_labels = False, cmap = "bwr")



**Импорт**

In [57]:

dict\_page = {}

k = 0

for i in colors\_page:

dict\_page.update({LG.nodes[k].name: i})

k = k + 1

nx.set\_node\_attributes(G, dict\_page, "page" )

dict\_auth = {}

k = 0

for i in colors\_auth:

dict\_auth.update({LG.nodes[k].name: i})

k = k + 1

nx.set\_node\_attributes(G, dict\_auth, "auth" )

dict\_hubs = {}

k = 0

for i in colors\_hubs:

dict\_hubs.update({LG.nodes[k].name: i})

k = k + 1

nx.set\_node\_attributes(G, dict\_hubs, "hubs" )

nx.write\_gexf(G, "zetshell.gexf")

**Применение алгоритма для интернет модуля**

netmodules = ['nmcli', 'dhcpcd', 'networkctl', 'ping', 'ip', 'ss', 'udevadm']

for i in netmodules:

if lief.is\_elf('/bin/'+i):

c = 0

path = pathlib.Path("/bin/"+i)

try:

deps = lddwrap.list\_dependencies(path=path)

print(i)

for dep in deps:

c = c+1

if dep.path == None:

print(dep.soname)

else:

print(dep.path)

print(c)

print('')

except: pass

# Изъятие импортов из программ

libs = []

G = nx.DiGraph()

for i in netmodules:

path = pathlib.Path("/bin/"+i)

deps = lddwrap.list\_dependencies(path=path)

for dep in deps:

if dep.path == None:

libs.append(dep.soname)

else:

libs.append(str(dep.path))

# Формирование графа основываясь на взаимодействиях программы с SO и SO друг с другом

start\_path = pathlib.Path("/bin/"+i)

for lib in libs:

if lib[8:9] != "/" :

continue

G.add\_edge(str(start\_path), str(lib))

path = pathlib.Path(lib)

deps = lddwrap.list\_dependencies(path=path)

for dep in deps:

if (str(dep.path)[8:9] == "6") or (dep.path == None):

continue

if dep.path == None:

G.add\_edge(str(lib), str(dep.soname))

else:

G.add\_edge(str(lib), str(dep.path))

In [68]:

LG = Graph()

for i in netmodules:

start\_path = pathlib.Path("/bin/"+i)

for lib in libs:

if lib[8:9] != "/" : # Отсеивание либ не из /usr/lib/

continue

LG.add\_edge(str(start\_path), str(lib))

path = pathlib.Path(lib)

deps = lddwrap.list\_dependencies(path=path)

for dep in deps:

if (str(dep.path)[8:9] == "6") or (dep.path == None): # Отсеивание либ не из /usr/lib/

continue

if dep.path == None:

LG.add\_edge(str(lib), str(dep.soname))

else:

LG.add\_edge(str(lib), str(dep.path))

**PageRank**

In [69]:

def output\_PageRank(iteration, graph, damping\_factor):

PageRank(graph, damping\_factor, iteration)

pagerank\_list = graph.get\_pagerank\_list()

print('PageRank:')

print(pagerank\_list)

return pagerank\_list

In [70]:

damping\_factor = 0.85

iteration = 10

colors\_page = output\_PageRank(iteration, LG, damping\_factor)

PageRank:

[0.016 0.017 0.017 0.017 0.018 0.017 0.017 0.024 0.017 0.017 0.017 0.019

0.031 0.09 0.019 0.017 0.017 0.018 0.019 0.019 0.017 0.018 0.018 0.018

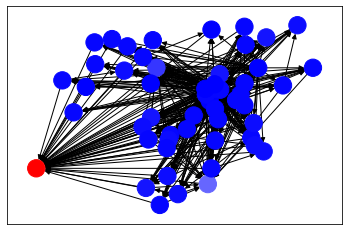
0.017 0.018 0.017 0.017 0.017 0.017 0.017 0.017 0.017 0.017 0.017 0.017

0.017 0.017 0.017 0.017 0.018 0.017 0.018 0.017 0.018 0.017 0.017 0.016

0.016 0.016 0.016 0.016 0.016]

In [71]:

nx.draw\_networkx(G, node\_color=colors\_page, with\_labels = False, cmap = "bwr")



**HITS**

In [72]:

def output\_HITS(iteration, graph):

HITS(graph, iteration)

auth\_list, hub\_list = graph.get\_auth\_hub\_list()

print('Authority:')

print(auth\_list)

print('Hub:')

print(hub\_list)

return auth\_list, hub\_list

In [73]:

iteration = 500 # Default 500

colors\_auth, colors\_hubs = output\_HITS(iteration, LG)

Authority:

[0. 0.019 0.021 0.022 0.022 0.022 0.021 0.027 0.021 0.021 0.021 0.023

0.029 0.031 0.024 0.023 0.022 0.024 0.023 0.024 0.022 0.022 0.022 0.023

0.019 0.02 0.019 0.021 0.021 0.021 0.021 0.021 0.021 0.021 0.021 0.021

0.021 0.021 0.021 0.021 0.021 0.021 0.021 0.019 0.02 0.019 0.019 0.

0. 0. 0. 0. 0. ]

Hub:

[0.089 0.045 0.028 0.011 0.007 0.012 0.007 0.003 0.003 0.018 0.016 0.008

0.003 0. 0.003 0.009 0.003 0.003 0.005 0.005 0.014 0.01 0.01 0.003

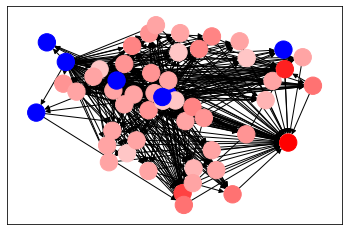
0.004 0.003 0.048 0.003 0.003 0.003 0.005 0.003 0.016 0.003 0.008 0.007

0.012 0.003 0.003 0.005 0.003 0.007 0.003 0.004 0.003 0.005 0.003 0.089

0.089 0.089 0.089 0.089 0.089]

In [74]:

nx.draw\_networkx(G, node\_color=colors\_auth, with\_labels = False, cmap = "bwr")



In [75]:

nx.draw\_networkx(G, node\_color=colors\_hubs, with\_labels = False, cmap = "bwr")

