Методологический подход к решению задачи поиска ошибок и недокументированных возможностей в программном обеспечении на основе класса задач поиска путей на графах

Н.Н. Самарин[1,2]

[1]*Научно-исследовательский институт, Москва*
[2]*Московский технический университет связи и информатики, Москва*

Аннотация: В работе методологически показана тождественность математической задачи поиска путей на графе технической задаче поиска различного рода дефектов в программном обеспечении, в частности, ошибок и недокументированных возможностей. Кратко описана графовая модель функционирования программного обеспечения, ставшая основой для представленной методологии. Заявлены новые направления исследований, базирующиеся на задачах теории графов, ранее не использовавшиеся для поиска дефектов в программном обеспечении.
Ключевые слова: графовая модель, программное обеспечение, поиск путей на графе, поиск в ширину, метод встречи посередине, вредоносное программное обеспечение.

Введение

Математический аппарат теории графов широко применяется как в сфере информационных технологий, так и в других сферах деятельности. Его удобство обосновано тем, что в виде графа легко может быть представлен объект практически любой области деятельности: связи между пользователями социальной сети [1, 2], структура молекул и химические соединения [3, 4], топология сети передачи данных [5], транспортный путь между городами [6] и т.п.

В терминах теории графов часто решаются прикладные задачи: например, задача оценки схожести графов для сравнения графов потока управления программ [7], задача поиска кратчайшего пути для оптимизации логистических перевозок [8], задача сетевого планирования для оптимальной организации работ по проекту.

Широкое применение графы также нашли в информационной безопасности: в контексте поиска ошибок и дефектов программного обеспечения (далее ПО) теория графов используется для представления графа потока управления [9, 10], для описания характерного вредоносного

ПО (далее ВПО), обнаруживаемого за счет структурного и taint-анализа. В контексте поиска ВПО [11-12] также стоит отметить тенденцию к использованию графовых нейронных сетей [13, 14], в частности с учетом изменения «весовых» характеристик графов [15].

Граф как некоторое абстрактное представление может быть успешно применено для моделирования многих объектов исследования – от сетевой инфраструктуры до отдельно взятого образца ПО. В дальнейшем исследовании показана возможность представления различных образцов бинарного ПО в виде ориентированного графа, учитывая связанные с этим аспекты его низкоуровневого функционирования. Показана связь математической задачи поиска путей на графе и задачи поиска ошибок и НДВ в ПО.

Представление ПО в виде графа

ПО, особенно в бинарном виде, представляет собой довольно сложный объект для моделирования. Это связано с наличием большого числа путей, по которым может пойти программа; с широким спектром возможных состояний, которые может принять программа, с наличием различных точек входных данных, способных повлиять на работу программы. Следует также отметить, что в части анализа состояния программы необходимо учитывать содержимое регистров процессора и ячеек памяти (переменных) на некоторый зафиксированный момент [16]. Такое состояние программы ассоциируют с вершиной графа (функциональная вершина), а переходы между ними отображают дугами. Необходимость учитывать низкоуровневые характеристики ПО усложняют модель, однако граф как математический объект имеет характеристики, позволяющие детализировать описание программы.

Ранее автором уже была предложена математическая модель, позволяющая описать программу в виде ориентированного графа [17].

Модель ориентирована на представление бинарного ПО в виде кортежа: S= <G,P>.

Вершины V={v_1,…,v_n}ориентированного графа G представляют собой базовые блоки – последовательности ассемблерных инструкций, имеющие только одну точку входа и одну точку выхода и не имеющие инструкций или операторов передачи управления ранее точки выхода [17]. Дуги E={e_1,…,e_m} графа G представляют собой перекрестные ссылки между базовыми блоками.

Множество P представляет собой множество кортежей {p_1,p_2,…,p_n}, ∀i,i=(1,n)¯ p_i имеет вид <v_i,C,reach,err>. Каждый кортеж называется «точкой» и включает: v_i – базовый блок ПО, C= <R,M> – стартовый контекст базового блока v_i, R – множество значений регистров процессора, M – множество символьных ограничений, накладываемых на память при выполнении программного кода, reach – бинарный флаг, показывающий факт достижимости базового блока v_i с контекстом C, err – бинарный флаг, показывающий факт возникновения исключения при использовании контекста C при выполнении базового блока v_i.

Помимо стартового контекста, который есть у каждого базового блока, есть еще текущий контекст C_cur – он меняется при выполнении базового блока, при этом на его значения накладываются определенные ограничения – на память и/или регистры для срабатывания перехода в данной перекрестной ссылке между парой базовых блоков. Эти ограничения в терминах теории графов могут быть представлены как пропускная способность дуги w_(e_i) – характеристика транспортной сети. Использование термина «транспортная сеть» в данном случае не противоречит ключевым положениям модели, поскольку сеть представляет собой ориентированный граф с заданными ненулевыми пропускными способностями дуг, что требует лишь введения

дополнительной характеристики дуги, связанной с текущим контекстом C_cur.

Значимым для модели является факт того, что при выполнении ПО изменяется набор значений регистров процессора, а также состояние областей памяти. Данный факт отражен в модели путем представления v_i в виде кортежа <A_i,I_i>, а e_i в виде кортежа <addr_from,addr_to>.

Для описания базового блока – вершины графа – вводится множество принадлежащих ему адресов A={addr_1,addr_2,…,addr_l} и инструкций I={〖instr〗_1,〖instr〗_2,…,〖instr〗_k}. Перекрестные ссылки (дуги графа) – характеризуются addr_from и addr_to – адресами базового блока, который ссылается, и базового блока, на который происходит ссылка, соответственно.

Таким образом, предложенная графовая модель инвариантна относительно типа бинарного ПО и учитывает необходимые аспекты, выходящие на первый план при исполнении программы.

Интерпретация задач поиска ошибок и НДВ в ПО в терминах теории графов

Моделирование объекта графом сразу переводит исследуемую предметную область на высокий уровень абстракций и подчеркивает классические графовые задачи, которые могут быть интерпретированы в контексте прикладной задачи исследования.

Так, классической задачей теории графов является задача поиска путей на графе. В зависимости от особенностей объекта моделирования, это может быть поиск пути на ориентированном/неориентированном, бесконтурном или циклическом графе, поиск кратчайшего пути или всех возможных путей между заданной парой вершин; поиск путей от заданной вершины до всех остальных, поиск путей с обязательным посещением некоторого множества вершин графа и т.д. [18].

В исследовании предложено трактовать прикладную задачу поиска ошибок и НДВ в ПО как совокупность решения теоретических задач поиска путей и вершин на графе. С технической точки зрения, ошибка и НДВ представляют собой совокупность:

места локализации ошибки – конкретные ветвь исполнения программы и базовый блок;

состояний текущего базового блока, включая память и регистры, с учетом накладываемых на них ограничений;

набора определенных входных данных, затрагивающих базовый блок – не обязательно напрямую, но, если поток входных данных попадает и в искомый базовый блок, это тоже необходимо учитывать.

Таблица № 1

Связь терминов прикладной задачи и теории графов

№	Термины прикладной задачи поиска ошибок и НДВ в ПО	Термины теории графов и графовой модели
1.	Место локализации дефекта (ошибки или НДВ) – ветвь программы и базовый блок, в котором найден потенциальный дефект	Подграф G' исходного ориентированного графа G и конкретная вершина графа v_i
2.	Состояния текущего базового блока, включая память и регистры	C = <R, M> – стартовый контекст базового блока v_i, R – множество значений регистров процессора, M – множество символьных ограничений, накладываемых на память при выполнении программного кода, а также C_{cur} – текущий контекст базового блока v_i
3.	Ограничения на память и регистры процессора, удовлетворение которых необходимо для выполнения инструкций перехода между базовыми блоками	Пропускная способность w_{e_i} ассоциированная с каждой дугой e_i графа G
4.	Набор определенных входных	Сетевой поток – f в графе G,

	данных для базового блока («полезная нагрузка», передаваемая злоумышленником для эксплуатации уязвимости или воспроизведения ошибки)	действительная функция $f: V \times V \to R$, удовлетворяющая условиям антисимметричности, ограничению пропускной способности и закону сохранения потока. В контексте «полезной нагрузки» речь идет о потоке определенной величины $\|f\|$, и возможность его передачи определяется пропускной способностью дуг w

Представление уравнений, которые необходимо решить при символьном исполнении, может быть выполнено как поиск максимального потока в сети – ориентированном графе, каждая дуга которого имеет положительную пропускную способность. Пропускная способность дуги косвенно характеризует диапазон значений, принимаемых символьной переменной, при насыщении дуги все ее значения учтены, уравнения решены.

Поиск ошибок и НДВ заключается в многократном запуске программы с варьирующимися входными данными и условиями эксплуатации с целью выявления некорректного результата как на выходе программы, так и в каждой ее точке. На практике, как правило, исследователи безопасности ПО априори предполагают наиболее вероятные участки кода, которые в первую очередь требуется проверить на наличие дефектов.

Однако теоретическое обнаружение предполагаемого дефекта (уязвимости или НДВ) не подтверждает того, что уязвимость может быть проэксплуатирована (а ошибка – воспроизведена) злоумышленником в реальных условиях. В связи с этим, необходимо подтвердить или опровергнуть достоверность найденных дефектов – ошибок и НДВ – для всего множества выявленных дефектов.

Для решения задачи определения достоверности найденных дефектов могут быть использованы терминология и параметры разработанной ранее графовой модели [17]. Пусть в результате тестирования обнаружено множество дефектов ПО, локализованных в базовых блоках программы, представляемых вершинами графа V^'={ ⟦v^'⟧ _1, ⟦v^'⟧ _2,…, ⟦v^'⟧ _n}. Тогда входные точки в программу, обозначаемые как V_start={v_start^1,v_start^2,…,v_start^m}, одновременно являются точками проникновения злоумышленника. В них он может попытаться повлиять на выполнение программы, передавая вредоносный контекст (выражаемый как сетевой поток определенной величины |f|) так, чтобы достичь определенных состояний регистров и памяти v_i^', приводящих к такому запуску этого базового блока, что в работе ПО наблюдается дефект. Обозначим такие состояния регистров и памяти как C_exp= <R_exp,M_exp>. Тогда в терминах модели достоверность дефекта Exp есть сочетание v_i^', v_start^j, начальных состояний регистров и памяти v_i^', обозначаемых C= <R,M>, и C_exp= <R_exp,M_exp>. Математически достоверность выражается как кортеж Exp=<v_i^',v_start^j,C,|f|>. Согласно графовой модели функционирования ПО, изменение текущего контекста c_cur базового блока меняется при его вызове каким-либо другим базовым блоком. Таким образом, если в начальной точке злоумышленник внесет свой вредоносный контекст и сможет донести его до целевого блока, то дефект является достоверным.

Теорема о необходимых условиях достоверности выявленного дефекта в ПО

Дано: ПО, представленное в виде кортежа S= <G,P>, где G – ориентированный граф, каждая дуга которого имеет неотрицательное значение пропускной способности; выявленный в результате тестирования

дефект, локализованный в базовом блоке 〖v^'〗_i, множество входных точек в программу V_start.

Доказать, что необходимыми условиями достоверности выявленного дефекта в базовом блоке 〖v^'〗_i являются:

1) наличие хотя бы одного пути до него;

2) возможность передачи по этому пути сетевого потока величины |f|, обеспечивающего воспроизведение дефекта (эксплуатацию уязвимости/повторение ошибки).

Доказательство.

Докажем теорему от противного.

1. Пусть в графе не существует ни одного пути между вершинами v_i^'∈V' и v_start^j∈V_start: path(v_i^'; v_start^j)=∞. Это означает, что злоумышленник не имеет возможности попадания в целевую вершину v_i^' из текущей v_start^j, следовательно, дефект не является достоверным (ошибка не может быть воспроизведена/уязвимость не может быть проэксплуатирована).

2. Пусть существует путь между вершинами v_i^'∈V' и v_start^j∈V_start: path(v_i^'; v_start^j)=p. Здесь p – максимальное значение сетевого потока, которое может быть пущено по пути path(v_i^'; v_start^j). Пусть path(v_i^'; v_start^j) включает дуги {(v_start^j,v_k),(v_k,v_l),…(v_z,v_i^')}. Если хотя бы одна из дуг имеет пропускную способность f(v_a,v_b)<C_exp, a,b∈V,a≠b, сетевой поток величины |f| не дойдет полностью до v_i^', следовательно, дефект не является достоверным (ошибка не может быть воспроизведена/уязвимость не может быть проэксплуатирована).

Таким образом, выполнение обоих условий является необходимым для подтверждения достоверности выявленного дефекта в базовом блоке 〖v^'〗_i, что и требовалось доказать.

Тогда в терминах графовой модели задача поиска ошибок и НДВ формулируется следующим образом:

для заданного кортежа <G,P> и заданного множества V^T целевых вершин найти:

- множество смежных вершин для каждой вершины множества V^T, что актуально в связи со спецификой предметной области: часть адресов между базовыми блоками являются динамическими, и определить наличие связи между базовыми блоками можно только при непосредственном исполнении программы, в терминах модели – при проходе по графу;

- множество всех возможных путей от множества вершин V^START до каждой вершины из множества V^T с учетом того, что каждый кортеж p^T, ассоциированный с вершинами V^T, имеет вид <v_i,C,reach,1>, где v_i∈V^T, а 1 характеризует установленное значение бинарного флага, свидетельствующее о наличии исключения;

- множество всех возможных путей от каждой вершины множества V^T до выходной точки программы.

Тогда можно отразить связь между методами решения задач поиска путей на графе и известными техническими методами поиска уязвимостей (таблица 2).

Из таблицы 2 можно заметить, что вариативные подходы к поиску путей на графе могут быть использованы для математической интерпретации методов поиска уязвимостей, ошибок и НДВ.

Таблица № 2

Связь методов теории графов и методов поиска ошибок и НДВ в ПО

Метод поиска ошибок и НДВ в ПО	Метод теории графов	Задача теории графов, которая может быть решена с использованием данного метода

Метод поиска ошибок и НДВ в ПО	Метод теории графов	Задача теории графов, которая может быть решена с использованием данного метода
1. Анализ машинного кода (статический анализ). 2. Анализ байт-кода (статический анализ. При анализе машинного и байт-кода с использованием статического анализа распространен подход, в рамках которого строится абстрактное синтаксическое дерево декомпилируемого кода. При его обходе методом поиска в глубину происходит вывод исходного кода и сохранение информации о соответствии номеров строк восстанавливаемого кода и позиций инструкций (в методах байткода)	Метод поиска в глубину [19]	Задача обхода рассматриваемого графа из заданной (корневой) вершины с посещением каждой вершины не более одного раза
Синтаксический анализ (статический анализ). Может использовать различные графовые методы: 1. Метод поиска в глубину. Он также используется для обхода абстрактного синтаксического дерева декомпилируемого кода, чтобы обеспечить понимание возможностей продвижения от входной точки программы к самому концу программы. 2. Поиск заданного	Метод поиска в глубину	Задача обхода рассматриваемого графа из заданной (корневой) вершины с посещением каждой вершины не более одного раза.
	Поиск заданного подграфа	Задача поиска определенного подграфа, характеризующегося заданным числом вершин и дуг, степенной последовательностью, в графе большей размерности
	Поиск изоморфных графов [20]	Задача поиска изоморфных графов, которые можно получить из исходного

Метод поиска ошибок и НДВ в ПО	Метод теории графов	Задача теории графов, которая может быть решена с использованием данного метода
подграфа в графе большей размерности для выявления известных сигнатур вредоносного ПО. 3. Поиск изоморфных графов как расширение сигнатурных шаблонов		путем нахождения для него функции перенумерации вершин и дуг
Межпроцедурный контекстно-чувствительный анализ (статический анализ). Данный метод поиска ошибок и НДВ также относится к классу методов статического анализа, однако здесь основное внимание уделяется контексту работы программы, во многом определяемому параметрами участка исходного кода	Метод поиска в глубину с учетом весов дуг	Задача обхода взвешенного графа из заданной (корневой) вершины с посещением каждой вершины не более одного раза
Анализ, чувствительный к путям выполнения (статический анализ). Ключевая идея метода состоит в рассмотрении каждого исследуемого пути по отдельности	Метод поиска в глубину с учетом весов дуг	Задача обхода взвешенного графа из заданной (корневой) вершины с посещением каждой вершины не более одного раза
	Поиск максимального потока в сети (с использованием алгоритма Форда-Фалкерсона) [21]	Задача нахождения такого потока в транспортной сети, что сумма потоков из истока максимальна.
Формальная верификация (в частности, на основе символьного исполнения).	Поиск максимального потока в сети (с	Задача нахождения такого потока в транспортной сети, что сумма потоков из

Метод поиска ошибок и НДВ в ПО	Метод теории графов	Задача теории графов, которая может быть решена с использованием данного метода
Техника символьного исполнения базируется на обобщении входных данных и представлении их в виде абстрактных символьных переменных. Каждое символьное выражение, составленное в ходе исполнения ПО, накладывает ограничения на его поток исполнения. Разрешение наложенных ограничений позволяет выяснить конкретные значения переменных	использованием алгоритма Форда-Фалкерсона)	истока максимальна
Фаззинг (динамический анализ). Фаззинг-тестирование на практике представляет собой тестирование с использованием вариативных входных данных, сформированных преимущественно некорректным образом и направленных на получение некорректных выходных данных и нарушение работы программы. При фаззинге входные данные варьируются так, чтобы максимизировать покрытие кода	Метод поиска в ширину [19]	Задача обхода графа из заданной (корневой) вершины с посещением каждой вершины не более одного раза.
	Метод поиска в глубину	Задача обхода рассматриваемого графа из заданной (корневой) вершины с посещением каждой вершины не более одного раза
Направленный ручной анализ. Как правило, выполняется специалистом высокой квалификации,	Использование графового представления программы	Задача моделирования заданного объекта в терминах теории графов

Метод поиска ошибок и НДВ в ПО	Метод теории графов	Задача теории графов, которая может быть решена с использованием данного метода
обладающим опытом в поиске дефектов программ		
Отслеживание помеченных данных (taint-анализ). Taint-анализ ориентирован на отслеживание потока входных данных и маркировку тех участков программы, выполнение которых будет затронуто этими данными. Для выполнения taint-анализа высокую эффективность демонстрирует подход, в рамках которого при поступлении входных данных в точку программы сначала определяется множество ближайших точек, в которые данные могут попасть из рассматриваемого узла	Метод поиска в ширину в сочетании с методами разметки (раскраски) графа	Задача обхода графа из заданной (корневой) вершины с посещением каждой вершины не более одного раза, усложненную раскраской графа для посещенных вершин
Моделирование атак. Как правило, при моделировании кибератак задается набор целевых участков кода, в которые необходимо попасть	Использование графового представления программы Поиск в глубину	Задачи: моделирования заданного объекта в терминах теории графов; обхода графа из заданной (корневой) вершины с посещением каждой вершины не более одного раза
Сканирование уязвимостей	Поиск заданного подграфа в графе большей размерности Поиск	Задачи: поиска определенного подграфа, характеризующегося заданным числом вершин и дуг, степенной

Метод поиска ошибок и НДВ в ПО	Метод теории графов	Задача теории графов, которая может быть решена с использованием данного метода
	изоморфных графов	последовательностью, в графе большей размерности; поиска изоморфных графов, которые можно получить из исходного путем нахождения для него функции перенумерации вершин и дуг
Анализ активности и потоков взаимодействия программы. Данный метод поиска ошибок и НДВ направлен на анализ того, какие пути исполнения программы зависят друг от друга, какие пути в зависимости от условий и входных данных влияют друг на друга, и можно ли вызвать ошибку, влияя на взаимодействующие потоки	Поиск максимального потока в сети (с использованием алгоритма Форда-Фалкерсона)	Задача нахождения такого потока в транспортной сети, что сумма потоков из истока максимальна

Систематизировав связь методов теории графов и методов поиска ошибок и НДВ в ПО, можно сделать вывод о том, что задачи анализа ПО на предмет наличия разного рода дефектов представимы в виде различных задач на графах. Это позволяет говорить о том, что в работе получен значимый научный результат, открывающий фундаментальный задел для создания новых методов анализа ПО на предмет безопасности (поиска ошибок, уязвимостей, НДВ и т.п.) на основе различных графовых задач.

Методология

В рамках глобального исследования, реализуемого автором, первостепенное внимание уделено анализу бинарных файлов ПО на предмет наличия ошибок и НДВ. Предлагаемая методология направлена на решение следующих задач:

1. Представление процесса функционирования бинарного ПО на низком уровне (с учетом состояний памяти и регистров) в виде ориентированного графа, атрибуты вершин и дуг которого характеризуют состояние программы.

2. Анализ графа на предмет наличия для заданной вершины смежных с ней вершин. Решение задачи осложняется тем, что часть дуг в графе задана благодаря статическим адресам, а часть дуг становится известной непосредственно при исполнении программы.

3. Поиск пути на графе между заданной парой вершин с учетом проблемы непостоянных дуг (из-за динамических адресов), требующей определения смежных вершин на каждом шаге поиска, и необходимости сократить время анализа в связи с большим числом потенциальных ошибок.

В рамках предлагаемой методологии предложен метод точечного фаззинг-тестирования, являющийся эффективным расширением метода фаззинг-тестирования в памяти. Фаззинг-тестирование в памяти отличается от классической техники фаззинг-тестирования тем, что программа рассматривается не полностью, а только определенный участок кода. При этом модификации подвергаются не входные данные программы, а аргументы функций. Для выполнения точечного фаззинг-тестирования техника фаззинг-тестирования в памяти дополняется механизмом общих симуляций и оценкой достижимости тестируемых базовых блоков, называемых точками, с состояниями, вызывающими ошибки [16], базируется

на графовой модели и использует модифицированный метод поиска в ширину.

Использование графового представление ПО в данном случае требуется для определения «точек» тестирования – базовых блоков, представляющих собой определенные вершины графа. При проведении фаззинг-тестирования исследуются именно заданные вершины графа, с учетом определенных ограничений, накладываемых на их атрибуты. Для заданной вершины смежные с ней могут быть частично найдены с использованием статических адресов между базовыми блоками – они представляют собой постоянные дуги.

Для определения наличия динамических дуг вершин (наличия перекрестной ссылки между парой базовых блоков) для каждого базового блока анализируются инструкции с целевыми адресами, определяющими, куда из данного блока можно перейти. Целевые адреса либо являются константами, либо они вычислены на этапе загрузки и релокации (в обоих случаях, это постоянные дуги), либо они принимают значения, вычисляемые во время исполнения программы (это и есть случай с неизвестными, переменными дугами графа). Таким образом, за описание перехода между базовыми блоками (смежности вершин) во многом отвечает кортеж $< A_i, I_i >$.

Применение алгоритма поиска в ширину, с одной стороны, максимизирует покрытие графа на ранних шагах, а с другой стороны, сразу будет решать техническую проблему, связанную с наличием динамических адресов между базовыми блоками программы. Для решения этой проблемы на каждом шаге алгоритма поиска в ширину добавляется более глубокий анализ вершин, смежных с текущей.

Для определения наличия динамических дуг вершин (наличия перекрестной ссылки между парой базовых блоков) для каждого базового

блока анализируются инструкции с целевыми адресами, определяющими, куда из данного блока можно перейти. Целевые адреса либо являются константами, либо они вычислены на этапе загрузки и релокации (в обоих случаях, это постоянные дуги), либо они принимают значения, вычисляемые во время исполнения программы (это и есть случай с неизвестными, переменными дугами графа). Таким образом, за описание перехода между базовыми блоками (смежности вершин) во многом отвечает кортеж $< A_i, I_i >$.

Тогда задача определения смежных вершин сводится к следующему: на ориентированном графе $G =< V, E >$ для заданной вершины v_i найти путь в вершину v_j, если такой существует, с учетом наличия непостоянных дуг в графе. На практике определение наличия либо отсутствия достижимости между парой блоков реализуется путем разрешения символьных ограничений. Введем функцию g:

$$g(v_i, v_j) = f(< A_i, I_i >, < A_j, I_j >) = \begin{cases} TRUE, & \exists\, e_{ij} = (v_i, v_j) \\ FALSE, & \nexists\, e_{ij} = (v_i, v_j) \end{cases} \quad (1)$$

Функция g представляет собой некоторое обобщенное описание процесса разрешения символьных ограничений, и принятие ей значения TRUE говорит о наличии дуги между заданной парой вершин и, следовательно, об их смежности.

Значимым этапом предложенной методологии является оценка достижимости выявленных в процессе фаззинг-тестирования ошибок. Характерной особенностью фаззинг-тестирования в памяти является большое число ложных срабатываний, вследствие чего актуальной является задача анализа достоверности ошибок. Математически она представляет собой

оценку достижимости базовых блоков, то есть, поиска пути между парой вершин, с учетом их характеристик.

Одним из вариантов решения задачи может являться сочетания методов поиска в ширину и поиска максимального потока в сети, поскольку для оценки достижимости целесообразно использовать метод символьного исполнения. Однако предложенная графовая модель позволяет сделать решение более простым за счет вычисления значения функции g(v_i,v_j), требующего выполнения некоторых операций над кортежами <A_i,I_i> и <A_j,I_j>.

Методология также обеспечивает оптимизацию всего процесса фаззинг-тестирования за счет повышения скорости процесса оценки достижимости. В терминах теории графов предлагается выполнять поиск в ширину из обеих вершин – начальной v_i и конечной v_j, дополнив поиск методом встречи посередине.

Поиск в ширину уже отчасти является оптимизацией, поскольку он обеспечивает нахождение именно кратчайших путей: они демонстрируют наиболее простой с точки зрения злоумышленника способ эксплуатации обнаруженной ошибки или НДВ и дают достоверный ответ на вопрос, эксплуатируем ли данный дефект.

Схема работы метода встречи посередине (meet-in-the-middle) представлена на рисунке 1. Это метод решения уравнения вида $\varphi(x)=\psi(y)$, где $x\in X, y\in Y$. Он базируется на разбиении задачи пополам и ее решении через независимые вычисления над каждой частью. Сложность работы метода составляет O(F(X)+Y+G_X (y), где F(X) – время построения множества X, G_X (y) – время поиска элемента x в множестве X, удовлетворяющее решению при заданном y, или проверка, что такого x не существует.

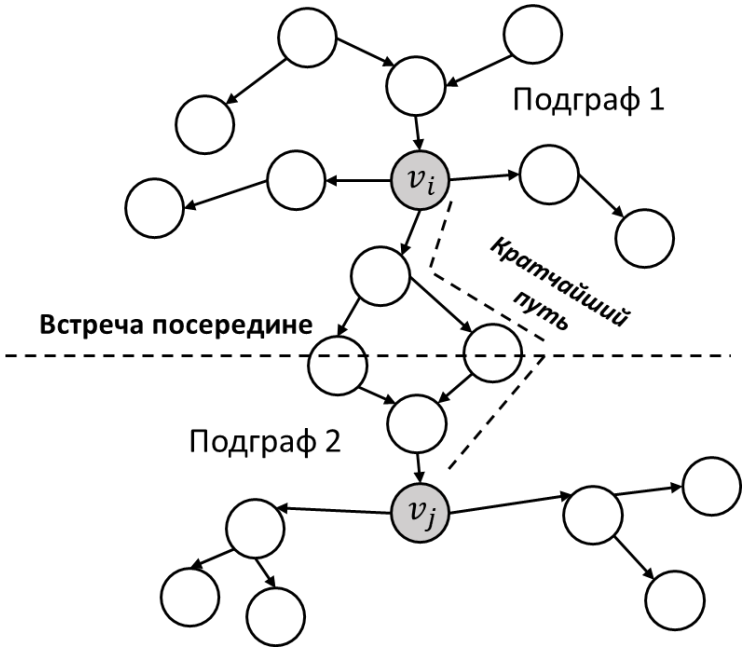

Рис. 1. – Поиск пути на графе с использованием метода встречи посередине

Практические аспекты реализации методологии

На практике разрешение ограничений может быть успешно выполнено с применением прямого символьного исполнения. Суть данного подхода состоит в использовании абстрактных символьных переменных вместо переменных с конкретными значениями в качестве входных данных. Чтобы покрыть сразу все возможное множество входных данных, при выполнении программы на символьные переменные в точках ветвления накладываются определенные ограничения. Тогда задача сводится к решению системы уравнений, число которых соответствует числу ветвлений.

Однако использования только символьного исполнения недостаточно для эффективного анализа обнаруженных фаззинг-тестированием ошибок, поскольку в случае большой по объему программы с сильно ветвящейся структурой число уравнений будет крайне велико, и задача становится все более трудоемкой.

В связи с этим в рамках метода оценки достижимости, построенного на базе графовой модели и решенной на графе задачи поиска путей, предложено реализовать также обратное символьное исполнение. Оно ориентировано на сокращения числа ветвлений внутри кода и позволяет дополнять некоторые функции собственными наборами ограничений.

Объединение данных подходов реализовано за счет их последовательного соединения по множеству накладываемых ограничений при прохождении графа (при реализации поиска в ширину). Удобство сочетания прямого и обратного символьного исполнения также подчеркивается тем, что форматы выходных методов сопоставимы, и выходные данные одного метода можно использовать как входные для второго (и наоборот). Метод встречи посередине помогает сопоставить выходные данные методов прямого и обратного символьного исполнения, он будет успешно применим в случае независимого задания для каждого метода входных данных и ограничений на них, а также точки встречи этих методов.

Результатом является вычисленное множество ограничений, являющегося пересечением множеств ограничений на выходах обоих методов. Затем происходит упрощение вычислений и представление итогового результата (в случае недостижимости целевого базового блока, множество будет пустым).

Использование графовых методов для создания новых методов поиска уязвимостей, ошибок и НДВ в ПО

Теория графов как мощный математический инструмент открывает широкие возможности для создания новых методов поиска уязвимостей, ошибок и НДВ в ПО. В качестве дальнейшего направления исследований предлагается спектр графовых задач, на основе которых можно разработать новый подход к поставленной задаче.

1. Задача поиска наиболее статистически часто встречающихся подграфов исходного графа (сетевых мотивов) в графе большей размерности [22]. Сетевой мотив может характеризовать некоторые неизменяемые, связанные между собой участки программного кода. Это может быть использовано для:

– сравнения между собой различных версий ПО;

– проверки на отсутствие модификаций, внесенных при передаче ПО, например, по незащищенному каналу;

– структурного анализа на предмет наличия участков кода, характерных для вредоносной нагрузки.

2. Задача поиска инвариантов графа применительно к потокам выполнения программы может быть использована для того, чтобы подтверждать отсутствие в программе вредоносного функционала.

3. Задача поиска гамильтоновых цепей или циклов позволит создавать оптимизированные методы тестирования, максимизирующие число посещаемых участков программного кода за один тестовый прогон.

4. Использование спектральной теории графов [23], в частности, спектра как графового инварианта, открывает широкие возможности по сравнению между собой функционала различных версий одного и того же ПО в репозиториях, но также может стать основой для быстрого поиска отличающегося функционала в программе по сравнению с ее исходной, «эталонной», версией. Добавление дополнительных возможностей неизбежно отразится на спектральных характеристиках.

Заключение

В работе показана роль теории графов в решении сложной технической задачи поиска уязвимостей, ошибок и НДВ в ПО. Разработанная автором ранее графовая модель функционирования бинарного ПО, отличающаяся от известных моделей возможностью описывать его работу на низком уровне, с учетом базовых блоков, регистров процессора и памяти, стала основой для получения значимых теоретических результатов:

– подтверждении тождественности методов поиска путей на графе, дополненных методами решения близких графовых задач, методам поиска ошибок и НДВ в ПО;

– формулировке и доказательстве теоремы о необходимых условиях достоверности выявленного дефекта в ПО.

Совокупность модели, теоретических положений и методов оценки достижимости фрагментов программного кода и точечного фаззинг-тестирования представляют собой новую методологию автоматизированного поиска ошибок в программном коде бинарных образцов ПО. Ее практическая значимость определяется аспектами реализации заявленных методов, а также оптимизацией в части скорости оценки достоверности дефектов, выявленных в процессе фаззинг-тестирования с использованием метода встречи посередине.

Теоретическая значимость предложенной методологии состоит в открытии ряда направлений по использованию задач теории графов для создания новых подходов к поиску уязвимостей, ошибок и НДВ в ПО.

Литература

1. Ye Tao, Ying Li, Su Zhang, Zhirong Hou, Zhonghai Wu. Revisiting graph based social recommendation: A distillation enhanced social graph network //Proceedings of the ACM Web Conference 2022. – 2022. – pp. 2830-2838.

2. Kumar S., Mallik A., Panda B. S. Influence maximization in social networks using transfer learning via graph-based LSTM //Expert Systems with Applications. – 2023. – V. 212. – P. 118770.

3. Zhichun Guo, Kehan Guo, Bozhao Nan, Yijun Tian, Roshni G. Iyer, Yihong Ma, Olaf Wiest, Xiangliang Zhang, Wei Wang, Chuxu Zhang, Nitesh V. Chawla. Graph-based molecular representation learning //arXiv preprint arXiv:2207.04869. – 2022.

4. Rocío Mercado, T. Rastemo, Edvard Lindelöf, G. Klambauer, O. Engkvist, Hongming Chen, Esben Jannik Bjerrum. Graph networks for molecular design //Machine Learning: Science and Technology. – 2021. – V. 2. – №. 2. – P. 025023.

5. Wang C., An J., Mu G. Power system network topology identification based on knowledge graph and graph neural network //Frontiers in Energy Research. – 2021. – V. 8. – P. 613331.

6. Zhang L., Long C. Road network representation learning: A dual graph-based approach //ACM Transactions on Knowledge Discovery from Data. – 2023. – V. 17. – №. 9. – pp. 1-25.

7. Zhuo W., Tan G. Efficient graph similarity computation with alignment regularization //Advances in Neural Information Processing Systems. – 2022. – V. 35. – pp. 30181-30193.

8. Smirnov A. V. The optimized algorithm of finding the shortest path in a multiple graph //Modeling and Analysis of Information Systems. – 2023. – V. 30. – №. 1. – pp. 6-15.

9. Minmin Zhou, Jinfu Chen, Yisong Liu, Hilary Ackah-Arthur, Shujie Chen, Qingchen Zhang & Zhifeng Zeng. A method for software vulnerability detection based on improved control flow graph //Wuhan University Journal of Natural Sciences. – 2019. – V. 24. – №. 2. – pp. 149-160.

10. Filippo Contro; Marco Crosara; Mariano Ceccato; Mila Dalla Preda. Ethersolve: Computing an accurate control-flow graph from ethereum bytecode //2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC). – IEEE, 2021. – pp. 127-137.

11. Dana Warmsley, Alex Waagen, Jiejun Xu, Zhining Liu, and Hanghang Tong. A survey of explainable graph neural networks for cyber malware analysis. 2022 IEEE International Conference on Big Data (Big Data), 2022. – pp. 2932-2939.

12. Rui Zhu, Chenglin Li, Di Niu, Hongwen Zhang, and Husam Kinawi. Android malware detection using large-scale network representation learning. arXiv preprint arXiv:1806.04847, 2018.

13. Pengbin Feng, Jianfeng Ma, Teng Li, Xindi Ma, Ning Xi, and Di Lu. Android malware detection based on call graph via graph neural network. 2020 International Conference on Networking and Network Applications (NaNA), 2020. – pp. 368-374.

14. Qian Wang, Zhengdao Li, Hetong Liang, Xiaowei Pan, Hui Li, Tingting Li, Xiaochen Li, Chenchen Li, Shikai Guo. Graph Confident Learning for Software Vulnerability Detection // Engineering Applications of Artificial Intelligence. – 2024. – V. 133. – P. 108296.

15. Huijiang Liu, Shuirou Jiang, Xuexin Q, Yang Qu, Hui Li, Tingting Li, Cheng Guo, Shikai Guo. Detect software vulnerabilities with weight biases via graph neural networks // Expert Systems with Applications. – 2024. – V. 238. – P. 121764.

16. Самарин, Н. Н. Поиск ошибок в исполняемом коде методом точечного фаззинга // Методы и технические средства обеспечения безопасности информации. – 2023. – № 32. – С. 75. – EDN VVDPIG.

17. Самарин Н. Н. Модель безопасного функционирования программного обеспечения, формализующая контроль использования памяти и обращений

к ней процессора // Наукоемкие технологии в космических исследованиях Земли. – 2021. – Т. 13. – №. 1. – С. 68-79.

18. Изотова Т. Ю. Обзор алгоритмов поиска кратчайшего пути в графе // Новые информационные технологии в автоматизированных системах. – 2016. – №. 19. – С. 341-344.

19. Algorithms on Graphs: Let's talk Depth-First Search (DFS) and Breadth-First Search (BFS).

20. Малинин Л. И., Малинина Н. Л. Изоморфизм графов в теоремах и алгоритмах //М.: URSS. – 2009. – 249с.

21. Зеленов А. А., Тарасов А. А. Алгоритм Форда-Фалкерсона //Инновации. – №. 48. – С. 1864-1868.

22. Alon U. Network motifs: theory and experimental approaches //Nature Reviews Genetics. – 2007. – V. 8. – №. 6. – pp. 450-461.

23. Spielman D. Spectral graph theory // Combinatorial scientific computing. – 2012. – V. 18. – P. 18.

References

1. Ye Tao, Ying Li, Su Zhang, Zhirong Hou, Zhonghai Wu. Proceedings of the ACM Web Conference 2022. p. 2830-2838.

2. Kumar S., Mallik A., Panda B. S. Expert Systems with Applications. 2023. V. 212. P. 118770.

3. Zhichun Guo, Kehan Guo, Bozhao Nan, Yijun Tian, Roshni G. Iyer, Yihong Ma, Olaf Wiest, Xiangliang Zhang, Wei Wang, Chuxu Zhang, Nitesh V. Chawla. arXiv preprint arXiv:2207.04869. 2022.

4. Rocío Mercado, T. Rastemo, Edvard Lindelöf, G. Klambauer, O. Engkvist, Hongming Chen, Esben Jannik Bjerrum. Machine Learning: Science and Technology. 2021. V. 2. №. 2. P. 025023.

5. Wang C., An J., Mu G. Frontiers in Energy Research. 2021. T. 8. P. 613331.

6. Zhang L., Long C. ACM Transactions on Knowledge Discovery from Data. 2023. V. 17. №. 9. pp. 1-25.

7. Zhuo W., Tan G. Advances in Neural Information Processing Systems. 2022. V. 35. pp. 30181-30193.

8. Smirnov A. V. Modeling and Analysis of Information Systems. 2023. V. 30. № 1. pp. 6-15.

9. Minmin Zhou, Jinfu Chen, Yisong Liu, Hilary Ackah-Arthur, Shujie Chen, Qingchen Zhang & Zhifeng Zeng. Wuhan University Journal of Natural Sciences. 2019. V. 24. №. 2. pp. 149-160.

10. Filippo Contro; Marco Crosara; Mariano Ceccato; Mila Dalla Preda. 2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC). – IEEE, 2021. pp. 127-137.

11. Dana Warmsley, Alex Waagen, Jiejun Xu, Zhining Liu, and Hanghang Tong. 2022 IEEE International Conference on Big Data (Big Data). 2022. pp. 2932-2939.

12. Rui Zhu, Chenglin Li, Di Niu, Hongwen Zhang, and Husam Kinawi. Android malware detection using large-scale network representation learning. arXiv preprint arXiv:1806.04847, 2018.

13. Pengbin Feng, Jianfeng Ma, Teng Li, Xindi Ma, Ning Xi, and Di Lu. Android malware detection based on call graph via graph neural network. 2020 International Conference on Networking and Network Applications (NaNA). 2020. pp. 368-374.

14. Qian Wang, Zhengdao Li, Hetong Liang, Xiaowei Pan, Hui Li, Tingting Li, Xiaochen Li, Chenchen Li, Shikai Guo. Engineering Applications of Artificial Intelligence. 2024. V. 133. P. 108296.

15. Huijiang Liu, Shuirou Jiang, Xuexin Q, Yang Qu, Hui Li, Tingting Li, Cheng Guo, Shikai Guo. Expert Systems with Applications. 2024. V. 238. P. 121764.

16. Samarin, N. N. Metody i tekhnicheskie sredstva obespecheniya bezopasnosti informacii. 2023. № 32. P. 75. EDN VVDPIG.

17. Samarin N. N. Naukoemkie tekhnologii v kosmicheskih issledovaniyah Zemli. 2021. V. 13. №. 1. pp. 68-79.

18. Izotova T. YU. Novye informacionnye tekhnologii v avtomatizirovannyh sistemah. 2016. №. 19. pp. 341-344.

19. Algorithms on Graphs: Let's talk Depth-First Search (DFS) and Breadth-First Search (BFS).

20. Malinin L. I., Malinina N. L. Izomorfizm grafov v teoremah i algoritmah [Graph isomorphism in theorems and algorithms]. M.: URSS. 2009. 249 P.

21. Zelenov A. A., Tarasov A. A. Innovacii. №. 48. pp. 1864-1868.

22. Alon U. Nature Reviews Genetics. 2007. V. 8. № 6. pp. 450-461.

23. Spielman D. Combinatorial scientific computing. 2012. V. 18. P. 18.

Дата поступления: 8.09.2024

Дата публикации: 24.10.2024