УДК 004.056

И.А. Лубкин

Метрики защищенности приложений при использовании средств противодействия уязвимостям, основанных на возвратно-ориентированном программировании

Уязвимости, использующие возвратно-ориентированное программирование, создают угрозу функционирования информационных систем. Для противодействия им создано множество средств защиты, основанных на различных принципах функционирования. При этом отсутствуют общепризнанные подходы к оценке защищенности применяемых решений. В работе предложены метрики защищенности, позволяющие получить объективные данные об эффективности средств защиты от RoP-уязвимостей. Формируемые показатели защищенности отражают возможность проведения атаки при получении атакующим контроля над графом потока управления.
Ключевые слова: уязвимость, удаленное исполнение кода, RoP, защита кода, метрики.
DOI: 10.21293/1818-0442-2021-24-4-46-51

Потребность защиты от атак, основанных на возвратно-ориентированном программировании (RoP-атак), привела к созданию множества подходов, применяемых для решения проблемы. Атаки данного типа позволяют реализовать класс уязвимостей удаленного исполнения кода. Актуальность защиты связана с тем, что злоумышленник может получить как минимум возможность исполнения произвольного алгоритма на атакуемой системе. Это в дальнейшем может привести к повышению привилегий, получению удаленного доступа и иным негативным последствиям.

С началом выпуска процессоров, поддерживающих технологию теневого стека (например, реализация CET [1], предложенная Intel), наметился прогресс в решении проблемы RoP-атак. При этом информационные системы, аппаратное обеспечение которых не поддерживает указанную технологию и экономически неприемлема его замена, остаются уязвимы. Уязвимыми также остаются приложения, для которых нет возможности проведения перекомпиляции. Технология теневого стека требует модификации машинного кода программы.

Для повышения защищенности в системах, не поддерживающих аппаратные средства защиты, для противодействия RoP-атакам применяются программные меры защиты. Нужно оценивать эффективность таких мер защиты, потому что не удаётся в общем случае доказать её теоретически. Это обусловлено невозможностью создания программного обеспечения с доказанно корректным алгоритмом. Также проблема оценки эффективности зачастую связана с тем, что средства защиты основаны на неизвестности для атакующего сведений об образе исполняемой программы или о месте его размещения в памяти.

Автором была предложена система защиты от RoP-атак. Принцип ее работы заключается в защите эпилогов подпрограмм и снижении числа пригодных для использования в рамках атаки участков (далее – гаджетов) вне эпилогов. Описание приведено в [2]. Особенностью подхода является отсутствие необходимости в исходных текстах и перекомпиляции, так как выполняется вставка кода непосредственно в бинарный образ. Средство вставки описано в [3].

Общепринятые подходы к оценке подобных систем защиты отсутствуют. Для такой оценки нужны метрики, по которым можно сравнить предлагаемое решение и аналогичные ему. Сравнивать подобные решения тяжело, потому что их авторы редко приводят детали программной реализации и результатов работы. Более того, решения могут не иметь общих характеристик для сравнения.

Рассматриваются операционные системы семейства GNU/Linux, функционирующие на аппаратной платформе с архитектурой AMD64 и Intel64.

Обзор существующих подходов

Рассмотрим принципы работы аналогов предлагаемого решения. По ним можно определить, пригодны существующие подходы к оценке эффективности или нет. Показатели качества средства защиты зависят от этих принципов. Анализ систем защиты от RoP-уязвимостей показал следующие основные направления защиты:

1. Рандомизация размещения (периодически [4], при старте [5], в родительских процессах [6]). Для данного подхода безопасность основана на неизвестности атакующему сведений о содержимом исполнимой памяти атакуемой программы. При наличии примитива чтения у атакующего программа становится уязвима перед атаками от момента получения достаточной информации и до следующего переразмещения, если таковое предусмотрено системой защиты. Метрики защищенности могут быть выражены как оценка вероятности наличия у атакующего примитива чтения и определение статистических показателей временно́го «окна», когда возможна атака. Такие метрики неприменимы для предлагаемой системы защиты, так как она построена на предположении известности атакующему информации о содержимом исполнимой памяти.

2. Снижение числа пригодных для атаки гаджетов (на этапе компиляции [7, 8], в процессе работы за счет оверлеев [9]). Предлагаемая система защиты

включает меры, соответствующие данному направлению. Анализ подходов, применяемых для оценки защищенности, рассмотрен подробнее далее.

3. Затруднение получения контроля над ГПУ (путем контроля целостности адреса возврата [10], сигнатуры в стеке [11], косвенного указания адресов переходов [12]). Предлагаемая система защиты включает меры, соответствующие данному направлению, поэтому анализ подходов, применяемых для оценки защищенности, рассмотрен подробнее далее.

Проанализируем подходы, используемые во втором и третьем направлениях оценки защищенности для определения возможности применения в данной работе:

– качественный анализ подверженности атакам. Основан на экспертной оценке путем соотнесения предлагаемых мер и модели атаки. Его недостатком является возможная неполнота рассмотрения, что может приводить к последующему выявлению недостатков защиты, пропущенных при авторском анализе [13] (возможным улучшением ситуации могло бы стать проведение аудита безопасности, но такая практика не встречается в проанализированном материале);

– подсчет числа гаджетов. Арифметический подсчет является корректным, когда целью ставится устранение всех инструкций возврата. При синонимическом преобразовании опасных участков с целью сделать их непригодными для атаки такой подход не может показать корректного результата без доработки с целью учета не только количества, но и качества гаджетов;

– применение существующих эксплойтов (с адаптацией или без). Не является показательным в общем случае при изменении конфигурации фреймов стека (как, например, в предлагаемом принципе защиты), так как применимость зависит от их структуры. При этом потенциально возможно создание иного эксплойта, применимого к защищенной программе.

Для объективной оценки защищенности программ при применении предлагаемого и подобных ему решений необходимо определить количество реально пригодных для атак гаджетов. Такой подход позволит обойтись без субъективных экспертных оценок и экстраполяции существующих уязвимостей на защищаемую программу. Качественная оценка является неполноценной, так как оперирует предположениями о действиях атакующего, а не объективными данными, показывающими возможность атаки.

Постановка задачи

Ставится задача формирования набора метрик, показывающих возможность проведения RoP-атак.

Исходные посылки, определяющие состояние системы на начало атаки: невыполнение любого из перечисленных условий делает атаку невозможной. Атакующий:

– преодолел ASLR за счет примитива чтения (наличие которого предполагается);

– разместил в стеке или куче произвольные данные, которые могут быть использованы как управляющие данные и аргументы гаджетов;

– имеет возможность передачи управления на произвольный адрес (т.е. рассматривается ситуация, когда существует возможность запустить исполнение RoP-цепочки при наличии набора подходящих гаджетов);

– обладает полной информацией о принципе работы системы защиты и образе атакуемой программы.

Рассмотрим детальнее гаджеты, используемые для реализации уязвимостей. Структурно они состоят из двух частей: алгоритмической и управляющей. Первая содержит последовательность обрабатывающих данные инструкций, вторая – управляющую инструкцию, которой он оканчивается. В зависимости от вида управляющей инструкции гаджеты могут быть разделены на:

– RoP – оканчивается инструкциями «RETN», «RETN imm16», «RETF» и «RETF imm16». Для их использования в составе гаджета в стеке (со смещением или без) должен быть размещен адрес следующего гаджета или он же, но с указанием сегментного регистра соответственно. Их классификация приведена в [14];

– JoP – оканчивается инструкцией вида «JMP reg». Для её использования в составе гаджета в регистр-аргумент атакующий должен поместить адрес следующего гаджета;

– CoP – оканчивается инструкцией вида «CALL reg». Для её использования в составе гаджета в регистр-аргумент атакующий должен поместить адрес следующего гаджета, а в рамках его исполнения извлечь адрес возврата, записанный в стек ранее.

Применимость гаджетов в рамках атак определяется целью злоумышленника. Например, если атакующему необходимо вызвать функцию, принимающую аргументы через регистры *xmm*, то при отсутствии инструкций работы с ними в составе гаджетов конечная цель не может быть достигнута. Были проанализированы существующие эксплойты для определения конечных состояний, достижение которых является критерием успешности атаки. В качестве них были определены критерии:

– вызов подпрограммы из системной библиотеки, спроецированной на адресное пространство атакуемой программы. В стек атакующим помещаются данные, в регистрах передаются аргументы, в финале управление передается на целевую системную функцию (например, установки атрибута исполнимости на необходимую атакующему страницу памяти). Порядок передачи аргументов определяется БИП [15];

– для уязвимостей кода в привилегированном режиме – отключение аппаратных систем защиты процессора путем изменения режима работы процессора вследствие записи в управляющие регистры;

– простой передачи управления недостаточно для атаки, так как это соответствовало бы ситуации,

когда в программе содержится участок кода, выполняющий все необходимые для злоумышленника операции (а это означает наличие программной закладки или НДВ).

Из анализа целей следует, что атакующему необходимо конкретное состояние системы для успешности атаки. Рассмотрим в качестве примера ситуацию, когда ему необходимо передать один аргумент в подпрограмму с указанием некоторого числа. Если атакующий может влиять на ГПУ, но не может записать в регистр rdi, используемый для передачи первого аргумента, необходимое значение, то атака не может быть проведена успешно. Соответственно, разрабатываемые метрики должны учитывать это.

Выбранные метрики оценки должны учитывать, что аргументы работы с памятью имеют различную разрядность. Вследствие этого атакующий может располагать, например, средствами взаимодействия в ОЗУ или передачи управления в диапазоне адресов от 0 до 2^{32}–1. Данный диапазон штатно не используется ОС, вследствие чего все такие средства становятся бесполезными. Отсутствие учета таких аспектов ведет к неверной оценке результатов применения защиты.

Предлагаемые метрики защищенности

В качестве исходных данных для метрик целесообразно использовать перечни гаджетов, формируемые существующими средствами их поиска. Это позволяет работать с тем же списком, что и атакующий. Корректность инструментария для поиска обеспечивается распространенностью использования, вследствие чего их ошибки устраняются.

При этом использовать напрямую результаты работы средств поиска гаджетов некорректно, так как не все выявляемые ими гаджеты применимы и полезны. Необходимо учитывать только те гаджеты, которые приближают атакующего к состоянию, соответствующему критерию успешной атаки. Найденные гаджеты подлежат фильтрации, а лишь затем анализу.

Для учета специфики предлагаемой методики защиты гаджеты должны быть проанализированы на наличие штатных инструкций, модифицирующих вершину стека, которая используется как адрес возврата, и известность атакующему аргумента этих инструкций. Это отражает принцип рассматриваемой методики защиты, когда перед инструкцией возврата производится вставка инструкции, выполняющей над адресом возврата операцию XOR. При этом аргумент операции генерируется случайно в прологе каждой подпрограммы. Если в рамках гаджета выполняется неизвестная операция, то вместо следующего гаджета управление будет передано на неопределенный адрес, что с близкой к 100% вероятностью вызовет нештатное прекращение работы защищенного приложения. В результате уязвимость удаленного кода не будет реализована.

При выборе метрик должен быть учтен вопрос возможности транслировать численные показатели в возможность проведения атаки. Например, если атакующий имеет в распоряжении набор гаджетов, который не позволяет модифицировать память или регистры, но позволяет передавать управление, то такой набор не обеспечивает достижения цели атакующего. Таким образом, гаджеты, которые не позволяют менять состояние регистров или памяти, исключаются из рассмотрения и не должны влиять на метрики.

Изложенные в предыдущем разделе требования позволяют сформировать следующие метрики:

– число уникальных гаджетов после исключения из рассмотрения неприменимых гаджетов (критерии описаны далее, так как они зависят от программного средства выявления гаджетов). Данная метрика показывает общее количество потенциальных гаджетов. Её расчет важен для определения относительного снижения числа гаджетов при сравнении оригинальной и защищенной программ, а также исключения ложноположительного вывода о повышении защищенности для приложений, сформированных средствами защиты, интегрированными в компилятор (такими, как G-free);

– перечень регистров, значение которых может задавать атакующий напрямую (например, в результате извлечения данных из стека). Может быть определен как аргумент-назначение таких команд, как «MOV», «POP» и т.п. Для краткости аргументы группируются по принципу вложенности (например, если атакующий может определять значение части регистра *ch* и регистра *ecx*, то достаточно указать значение *ecx*). Данная метрика показывает возможности атакующего задавать произвольные значения в регистрах программы. Без её расчета невозможно определение способности атакующего задавать значения регистров и параметры алгоритма, исполняемого в ходе атаки;

– перечень регистров, значение которых может задавать атакующий косвенно, – определяются регистры, на значение которых атакующий может влиять через выполнение операций или транзитных передач (из этого множества исключаются регистры, значение которых атакующий может задавать напрямую). Необходимость введения данной метрики обусловлена тем, что предыдущая метрика не отражает полного множества регистров, доступных атакующему;

– показатель доступных подпрограмм – при наличии возможности задать значение первого аргумента показатель равен единице, первого и второго – он равен двойке и т.д. Первые шесть аргументов передаются через регистры (указаны в порядке возрастания номера аргумента): *rdi*, *rsi*, *rdx*, *rcx*, *r*8, *r*9. Данный показатель отражает разнообразие возможных конечных подпрограмм, которые может вызывать атакующий (указывается для полноценно вызываемых подпрограмм и для частично определяемых аргументов, когда атакующий не может полностью определять значение регистра). При этом вследствие последнего критерия успешной атаки конечным состоянием не может быть вызов подпрограммы без

аргументов. Так как атакующий имеет возможность управлять ГПУ (и, следовательно, вызывать любую подпрограмму), то возможность атаки сводится к возможности указания аргументов необходимой подпрограммы.

Указанные метрики позволяют определить принципиальную невозможность проведения атаки. То есть если атакующий не имеет средств для достижения конечного состояния атаки, то атака удаленного исполнения кода не может быть проведена. Иначе атака потенциально может быть реализована.

Программная реализация

Средство реализации вычисления метрик представляет собой программу на языке python, выполняющую первичное определение состава гаджетов, их фильтрацию и определение метрик. Анализируемая программа обрабатывается как «черный ящик». Для получения первичных данных используется программное средство ROPgadget [16]. Его результаты работы записываются в файл. Для сформированных данных проводится фильтрация, в ходе которой устраняются ложно обнаруженные гаджеты. Удаляются из перечня при фильтрации гаджеты:

– содержащие инструкцию «RETF», если перед ней отсутствует REX-префикс «0×48», который задает 64-разрядный режим обработки. Это связано с тем, что при отсутствии префикса может быть задан только адрес в пределах 32 бит с расширением нулями старших 32 бит, но ОС не использует данный диапазон, что делает их ложно пригодными;

– состоящие только из инструкций передачи управления, так как без модификации содержимого регистров и/или памяти атакующий не может достигнуть цели атаки. Возможна ситуация, когда без некоторой специфичной инструкции передачи управления построение RoP-цепочки станет невозможным. Определение таких ситуаций требует анализа конкретного сценария проведения атаки. Вследствие того, что он не может быть определен без рассмотрения конкретной уязвимости, то для отсутствия сужения применимости метрик будем считать, что у атакующего присутствуют в составе гаджетов все необходимые инструкции для передачи управления, а гаджеты, состоящие только из таких инструкций, исключаются из рассмотрения;

– содержащие операции с неканоническими адресами (старшие два байта адреса должны быть равны 0 или 0xFFFF) или неиспользуемыми диапазонами адресов (первые 2^{32} байт);

– выполняющие операции с неизвестным для атакующего и не модифицируемым им аргументом над адресом возврата. Например, в рамках предлагаемой системы защиты над адресом возврата проводится операция XOR с регистром *r*11, содержащим неизвестное атакующему значение. Если у злоумышленника нет возможности определить содержимое регистра, то исполнение такой инструкции в составе гаджета приведет к аварийному завершению программы.

Для отфильтрованных данных проводится обработка, в ходе которой вычисляются значения метрик, характеризующих защищенность от RoP-атак, и делается вывод о возможности или невозможности атаки.

Результаты экспериментальной проверки

Для проверки корректности оценки защищенности была сформирована выборка программ. Для каждой программы из выборки применяется методика защиты. Для оригинальной и защищенной программы проводится получение метрик защиты. Полученные результаты анализируются.

В состав выборки программ включены: синтетические тесты (содержащие намеренно внесенные дефекты, позволяющие проводить RoP-атаки), заведомо уязвимые приложения (используемые в различных соревнованиях по информационной безопасности) и приложения из состава ОС для оценки защищенности до и после применения системы противодействия RoP-атакам.

Примеры результатов определения метрик представлены в таблице. Из приведенных данных следует, что состав регистров, на которые может влиять атакующий, радикально уменьшается после применения предлагаемой системы защиты. Это приводит к тому, что атакующий хотя и может передать управление на целевую подпрограмму, но не имеет возможности указать ни один её аргумент. Из этого можно сделать вывод о том, что RoP-атака для удаленного исполнения кода нереализуема, а защита обеспечена. Для выборки в целом состав гаджетов после применения системы защиты соответствует приведенным примерам и не позволяет злоумышленнику проводить атаки.

Метрики защищенности, полученные для приложений до и после защиты

Приложение	Уникальные гаджеты		Регистры, определяемые атакующим		Оперируемые регистры		Дост. арг. (полностью / частично)		Защита обеспечена
	Ориг.	Защ.	Ориг.	Защ.	Ориг.	Защ.	Ориг.	Защ.	
Синтетический тест, нет оптимиз.	389	3	*rax, ch, rbx, edx, rsi, rdi, rbp, rsp, r8d, r12, r13, r14, r15*	Нет	*rcx, rdx, fr7*	*al*	4/5	0	Да
Синтетический тест, ур. оптимиз. 2	122	3	*eax, rbx, rsi, rdi, rbp, rsp, r12, r13, r14, r15*	Нет	*rax, ecx, edx*	*al*	2/4	0	Да
Учебное уязвимое приложение [17]	85	3	*eax, rbx, rsi, rdi, rbp, rsp*	Нет	*rax, edx*	*al*	2/3	0	Да
coremark	232	4	*rax, rbx, rsi, rdi, rbp, rsp, r8b, r12, r13, r14, r15, xmm0*	*bh*	*ecx, edx, r9d*	*eax*	2/6	0	Да
sshd	4713	23	*eax, rbx, rsi, rdi, rbp, rsp, r8, r11, r12, r13, r14, r15*	*eax, edx*	*r9, r10*	*rax*	6/6	0	Да

Для синтетических тестовых и для заведомо уязвимых приложений было осуществлено формирование эксплойтов для оригинального и защищенного варианта. Целью являлся вызов подпрограммы «system», принимающей один аргумент – имя и аргументы приложения, запускаемого в командной оболочке. Для оригинальных приложений эксплойты корректно сработали и позволили выполнить запуск целевой подпрограммы. Для защищенного варианта реализация эксплойта оказалась невозможна вследствие отсутствия гаджетов, помещающих в регистр *rdi* первый аргумент вызывающей целевой подпрограммы. Эксплойт, выполняющий только передачу управления на целевую подпрограмму, приводит к аварийному завершению работы атакуемого приложения. Данный результат свидетельствует о корректности выбранных метрик.

Данный подход, в отличие от аналогов, основанных на экспертных оценках, оперирует объективными данными. Это устраняет человеческий фактор при анализе защищенности.

Заключение

Предлагаемые метрики позволяют получить объективные данные об изменении подверженности RoP-атакам в результате применения системы защиты. При изменении модели атак предложенные метрики могут быть адаптированы для новых условий. Фильтрация неприменимых в атаках гаджетов позволяет исключить ложноотрицательные результаты. Анализ применимости гаджетов для достижения атакующим его цели обеспечивает определение возможности проведения RoP-атак в принципе.

Исследование выполнено при финансовой поддержке Минобрнауки России (грант ИБ). Проект № 21/2020.

Литература

1. Shanbhogue V. Security Analysis of Processor Instruction Set Architecture for Enforcing Control-Flow Integrity / V. Shanbhogue, D. Gupta, R. Sahita // Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy. – New York: Association for Computing Machinery, 2019. – P. 1–11.

2. Лубкин И.А. Комплексная система защиты от уязвимостей, основанных на возвратно-ориентированном программировании / И.А. Лубкин, В.В. Золотарев // Труды СПИИРАН. – 2022. – № 21 (1) – в печати.

3. Lubkin I.A. Automatic equivalency restoration after software patching / I.A. Lubkin, V.V. Zolotarev // Proceedings of the 2021 IEEE International Conference «Quality Management, Transport and Information Security, Information Technologies». – Yaroslavl: Saint Petersburg Electrotechnical University «LETI», 2021. – P. 217–222.

4. Shuffler: Fast and deployable continuous code re-randomization / D. Williams-King, G. Gobieski, K. Williams-King, J.P. Blake // Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. – Savannah, GA, USA: USENIX Association, 2016. – P. 367–382.

5. Selfrando: Securing the Tor Browser against De-anonymization Exploits/ M. Conti, S. Crane, T. Frassetto, A. Homescu // Proceedings on Privacy Enhancing Technologies. – 2016. – № 4. – P. 454–469.

6. How to Make ASLR Win the Clone Wars: Runtime Re-Randomization. K. Lu, S. Nürnberger, M. Backes, W. Lee [Электронный ресурс]. – Режим доступа: http://dx.doi.org/10.14722/ndss.2016.23173, свободный (дата обращения: 01.09.2021).

7. G-Free: Defeating return-oriented programming through gadget-less binaries / K. Onarlioglu, L. Bilge, A. Lanzi, D. Balzarotti // Proceedings of ACSAC. – Austin, Texas, USA: ACM Press, 2010. – P. 49–58.

8. Defeating return-oriented rootkits with «return-less» kernels / J. Li, Z. Wang, X. Jiang, M. Grace // Proceedings of EuroSys. – Paris, France: ACM Press, 2010. – P. 195–208.

9. Execution Integrity with In-Place Encryption / D. Sullivan, O. Arias, D. Gens, L. Davi, A. Sadeghi, Y. Jin [Электронный ресурс]. – Режим доступа: https://arxiv.org/pdf/1703.02698.pdf, свободный (дата обращения: 01.09.2021).

10. RAP:RIP ROP 2015 [Электронный ресурс]. – Режим доступа: https://pax.grsecurity.net/docs/PaXTeam-H2HC15-RAP-RIP-ROP.pdf, свободный (дата обращения: 01.09.2021).

11. Perry Wagle, Crispin Cowan, StackGuard: Simple Stack Smash Protection for GCC, GCC Developers Summit, 2003 [Электронный ресурс]. – Режим доступа: https://gcc.gnu.org/pub/gcc/summit/2003/Stackguard.pdf, свободный (дата обращения: 01.09.2021).

12. Применение компиляторных преобразований для противодействия эксплуатации уязвимостей программного обеспечения / А.Р. Нурмухаметов, Ш.Ф. Курмангалеев, В.В.Каушан, С.С. Гайсарян // Труды ИСП РАН. – М.: ИСП РАН, 2014. – Т. 26, вып. 3. – С. 113–126.

13. Koo Z.Z. Analysis of ROP attack on grsecurity. PaX linux kernel security variables / Z.Z. Koo, Z. Ayop, Z.Z. Abidin // International Journal of Applied Engineering Research. – 2017. – Vol. 12, Iss. 23. – C. 13179–13185.

14. Вишняков А.В. Классификация ROP гаджетов // Труды ИСП РАН. – М.: ИСП РАН, 2016. – Т. 28, № 6. – С. 27–36.

15. AMD64 Architecture Processor Supplement Draft Version 0.99.7 [Электронный ресурс]. – Режим доступа: https://www.uclibc.org/docs/psABI-x86_64.pdf, свободный (дата обращения: 01.09.2021).

16. Инструмент ROPgadget. Репозиторий с исходным кодом [Электронный ресурс]. – Режим доступа: https://github.com/JonathanSalwan/ROPgadget, свободный (дата обращения: 01.09.2021).

17. Return oriented programming. Собираем exploit по кусочкам [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/255519/, свободный (дата обращения: 01.09.2021).

Лубкин Иван Александрович
Ст. преп. каф. безопасности информационных технологий (БИТ) Сибирского государственного ун-та науки и технологий (СибГУ) им. акад. М.Ф. Решетнева
Имени газеты «Красноярский рабочий» пр-т, 48б,
г. Красноярск, Россия, 660037
ORCID: 0000-0002-9657-0440
Тел.: +7 (391-2) 22-76-39
Эл. почта: lubkin@rambler.ru

Lubkin I.A.

Application security metrics when using defense system against vulnerabilities based on return-oriented programming

The vulnerabilities using return-oriented programming pose threats to the functioning of information systems. There are many protection systems to counteract them. They are based on various principles of functioning. At the same time, there are no generally accepted approaches to assess the security of applied solutions. The paper proposes security metrics that allow obtaining objective data on the efficiency of protection against RoP vulnerabilities. Proposed security metrics show ability to perform attack by gain control over control flow graph.

Keywords: vulnerability, remote code execution, RoP, code protection, metrics.
DOI: 10.21293/1818-0442-2021-24-4-46-51

References

1. Shanbhogue V., Gupta D., Sahita R. Security Analysis of Processor Instruction Set Architecture for Enforcing Control-Flow Integrity. *Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy*. New York, Association for Computing Machinery, 2019, pp. 1–11.

2. Lubkin I.A., Zolotarev V.V. Comprehensive defense system against vulnerabilities based on return-oriented programming. *SPIIRAS Proceedings*, 2022, no. 21 (1) (in Russ.). In press.

3. Lubkin I.A. Automatic equivalency restoration after software patching. *Proceedings of the 2021 IEEE International Conference «Quality Management, Transport and Information Security, Information Technologies»*, Yaroslavl, Saint Petersburg Electrotechnical University «LETI», 2021, pp. 217–222.

4. Williams-King D., Gobieski G., Williams-King K., Blake J. P. Shuffler: Fast and deployable continuous code re-randomization. *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, Savannah, GA, USA, USENIX Association, 2016, pp. 367–382.

5. Conti M., Crane S., Frassetto T., Homescu A. Selfrando: Securing the tor browser against de-anonymization exploits. *Proceedings on Privacy Enhancing Technologies*, 2016, no. 4, pp. 454–469.

6. How to Make ASLR Win the Clone Wars: Runtime Re-Randomization. Lu K., Nürnberger S., Backes M., Lee W. Available at: http://dx.doi.org/10.14722/ndss.2016.23173, free (Accessed: September 1, 2021).

7. Onarlioglu K., Bilge L., Lanzi A., Balzarotti D., Kidra E. G-Free: Defeating return-oriented programming through gadget-less binaries. *Proceedings of ACSAC*. Austin, Texas, USA, ACM Press, 2010, pp. 49–58.

8. Li J., Wang Z., Jiang X., Grace M., Bahram S. Defeating return-oriented rootkits with «return-less» kernels. *Proceedings of EuroSys*, Paris, France, ACM Press, 2010, pp. 195–208.

9. Execution Integrity with In-Place Encryption. Sullivan D., Arias O., Gens D., Davi L., Sadeghi A., Jin Y. Available at: https://arxiv.org/pdf/1703.02698.pdf, free (Accessed: September 1, 2021).

10. RAP:RIP ROP 2015. Available at: https://pax.grsecurity.net/docs/PaXTeam-H2HC15-RAP-RIP-ROP.pdf, free (Accessed: September 1, 2021).

11. StackGuard: Simple Stack Smash Protection for GCC. Wagle P., Cowan C. Available at: https://gcc.gnu.org/pub/gcc/summit/2003/Stackguard.pdf, free (Accessed: September 1, 2021).

12. Nurmukhametov A.R., Kurmangaleev S.F., Kaushan V.V., Gaissaryan S.S. Compiler protection techniques against software vulnerabilities exploitation. *Proceedings of the Institute for System Programming of the RAS*, 2014, no. 26(3), pp. 113–126 (in Russ.).

13. Koo Z.Z., Ayop, Z., Abidin Z.Z. Analysis of ROP attack on grsecurity. PaX linux kernel security variables. *International Journal of Applied Engineering Research*, 2017, no. 12, pp. 13179–13185.

14. Vishnjakov A.V. Classification of ROP-gadgets. *Proceedings of the ISP RAS*, 2016, vol. 28, issue 6, pp. 27–36 (in Russ.).

15. AMD64 Architecture Processor Supplement Draft Version 0.99.7 Available at: https://www.uclibc.org/docs/psABI-x86_64.pdf, free (Accessed: September 1, 2021).

16. ROPgadget tool. Source code repository. Available at: https://github.com/JonathanSalwan/ROPgadget, free. (Accessed: September 1, 2021).

17. *Return oriented programming. Sobiraem exploit po kusochkam* [Return oriented programming. Exploit construction by pieces]. Available at: https://habr.com/ru/post/255519/, free (Accessed: September 1, 2021) (in Russ.).

Ivan A. Lubkin
Senior Lecturer, Department of Information Technologies Security, Reshetnev Siberian State University of Science and Technology
48b, Imeni gazety «Krasnoyarskiy rabochiy» pr., Krasnoyarsk, Russia, 660037
ORCID: 0000-0002-9657-0440
Phone: +7 (391-2) 22-76-39
Email: lubkin@rambler.ru