фаза обнаружения, в противном случае выдается заключение о возможности наличия утечек. Оно содержит непредусмотренные адреса, а также инструкции, которые последними осуществляли в них запись. Использующий инструментальное средство аналитик должен определить, являются ли утечки реальными, или следуют из неточностей, возникших при абстрактной интерпретации.

Заметим, что используемые в работе абстрактные состояния построены таким образом, что исключают возможность возникновения пропусков, однако допускают ложные срабатывания. Таким образом, минимизация и адекватное ранжирование последних является важным направлением для дальнейшей работы.

СПИСОК ЛИТЕРАТУРЫ:

1. *Cousot P., Cousot R.* Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints // Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. P. 238–252. Los Angeles. California, 1977. ACM Press. New York. NY.

И. О. Леошкевич

АНАЛИЗ ИСПОЛНЯЕМОГО КОДА С ПОМОЩЬЮ СИМВОЛЬНОГО ВЫПОЛНЕНИЯ

Целями анализа программного обеспечения являются восстановление алгоритма его работы, например, при рассмотрении вредоносного ПО, а также поиск в нем ошибок и уязвимостей. Большинство современных инструментов статического анализа функционирует на уровне исходного кода. Такой подход имеет следующие недостатки: невозможность анализа программ без исходного кода, сложность реализации универсальных систем анализа, не ориентированных на конкретный диалект языка программирования, а также скрытость определенных аспектов в исходном коде [1]. Анализ исполняемого кода позволяет решить эти проблемы за счет того, что последний выполняется непосредственно процессором или виртуальной машиной и к нему сводятся программы на любом языке.

Получение модели программы по ее исполняемому коду связано с рядом специфических трудностей. В исполняемом коде отсутствует информация о переменных и типах данных. Для построения графа потока управления необходимо определение целей косвенных переходов, которые включают в себя вызовы виртуальных методов и возвраты из процедур. Существование различных архитектур процессоров и форматов исполняемых файлов, хотя и в меньшей степени, чем обилие языков программирования и их диалектов, затрудняет реализацию универсальной системы анализа. И все же применение тех же методов, что и для исходного кода, принципиально возможно, однако требуется их адаптация. В данной работе рассматривается совместное применение двух таких методов — абстрактной интерпретации (abstract interpretation) и символьного выполнения (symbolic execution).

Абстрактная интерпретация была введена в работе [2]. Каждой точке программы ставится в соответствие абстрактное состояние, представляющее собой надмножество всех возможных

в этой точке конкретных состояний. Множество абстрактных состояний с заданными на нем операциями называется доменом. Примерами доменов являются домены знаков, интервалов, мультиинтервалов и битовых векторов. Выполнение программы эмулируется по следующему правилу: все ее операторы перебираются в порядке, обратном завершению их рассмотрения при обходе в глубину (reverse postorder), и для каждого оператора производится обновление абстрактного состояния.

Для получения точного результата требуется несколько проходов, производимых, пока состояния не перестанут изменяться. Для ускорения сходимости используются операторы расширения и сужения, сопряженные, однако, с потерей точности.

Символьное выполнение представляет собой абстрактную интерпретацию с доменом формул. Для простых случаев состояния представляются парами имен переменных и их значений. Для поддержания компактности символьных состояний используется операция упрощения, реализацию которой можно найти в системах компьютерной алгебры, например Mathematica или Sage. В работе [3] показано, каким образом с помощью символьного выполнения можно точно анализировать сложные циклы. Символьное выполнение является дорогой операцией и потому должно быть использовано только там, где абстрактная интерпретация дала неудовлетворительные результаты. Такими точками могут быть неразрешенные косвенные переходы или места, где конкретный алгоритм анализа обнаружил потенциальную ошибку и желает произвести уточнение [4]. Для ускорения символьного выполнения можно предварительно производить слайсинг и удаление неиспользуемого кода, используя для определения зависимостей информацию, полученную абстрактной интерпретацией. На уровне исполняемого кода неиспользуемый код встречается достаточно часто. В качестве примера можно привести обновление флагов процессора при арифметических операциях, большинство из которых впоследствии не используются.

Память представляет собой большой массив, поэтому для описания состояний при символьном выполнении можно воспользоваться алгеброй массивов, в которой они представляют собой последовательности операций добавления элементов. При этом нужно уделять внимание псевдонимам переменных: адрес добавляемого значения может перекрывать любой из ранее добавленных. В случае с абстрактной интерпретацией «наивное» представление на основе массивов приведет к излишне пессимистичным результатам именно за счет большого количества потенциальных псевдонимов. Для их уменьшения можно воспользоваться понятием региона [1]. Регионы представляют собой непересекающиеся непрерывные участки памяти. Для отслеживания указателей на регионы все значения следует хранить в виде (регион, смещение), где для представления смещения можно выбрать любой домен. Значения внутри регионов можно хранить с использованием алгебры массивов или просто используя фиксированные смещения.

Предлагаемый алгоритм анализа выглядит следующим образом. На первом этапе производится один шаг абстрактной интерпретации, допускающий использование эвристик, направленных на максимальное покрытие кода, например допускается проход по ребрам, условия которых не выполняются, а также сохраняются перезаписанные указатели на код. На втором этапе строится и фиксируется граф потока управления и производится полноценная абстрактная интерпретация. На третьем этапе выявляются точки и участки памяти, значения которых нужно уточнить, производится символьное выполнение. Если уточнение возможно, то составляется список дополнительных ограничений для абстрактной интерпретации, и алгоритм повторяется заново с их учетом. В противном случае алгоритм завершается.

СПИСОК ЛИТЕРАТУРЫ:

1. *Balakrishnan G.* Wysinwyx: what you see is not what you execute. PhD thesis. Madison. WI. USA, 2007. Adviser-Reps Thomas.
2. *Cousot P., Cousot R.* Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints // Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. P. 238–252. Los Angeles. California, 1977. ACM Press. New York. NY.
3. *Fahringer T., Scholz B.* Advanced Symbolic Analysis for Compilers. Springer-Verlag, New York. Inc. Secaucus. NJ. USA, 2003.
4. *Станкевичус А.* Инструментальное средство проверки безопасности кода, предназначенного для исполнения в ГРИД-сетях // Безопасность информационных технологий. 2009. № 1. С. 58–61.

Е. Б. Маховенко, Н. Г. Сюсюгина

АНАЛИЗ КРИПТОСИСТЕМ НА СКРЫТЫХ ОТОБРАЖЕНИЯХ ПОЛЕЙ НЕЧЕТНЫХ ХАРАКТЕРИСТИК

Параметрами алгоритма являются: конечное поле $\mathbf{F} = \mathbf{F}_q$, где q — степень простого числа, n — степень расширения поля $\mathbf{F}$, d — степень секретного полинома $\rho(x)$, а также другие параметры для модификаций алгоритма: l — число исключаемых открытых полиномов для HFE- (добавляемых случайных полиномов для HFE+), v — число новых переменных для HFEv. В качестве характеристик алгоритма рассмотрим время генерации ключевой пары (t_{gen}), скорость шифрования (v_{enc}) и расшифрования (v_{dec}), а также сложность наилучшего алгоритма вскрытия ($diff$, F_4).

В работах [1–3], посвященных алгоритму HFE, рассматривается только зависимость стойкости алгоритма от параметров, причем выбор параметров алгоритма осуществляется исключительно эмпирически. Для нахождения оптимального набора параметров представляется целесообразным оценить все прямые зависимости и далее применить методы теории принятия решений. Выявленные прямые зависимости представлены в таблице 1.

Таблица 1.

y	x	$y(x)$	Параметры	Коэффициент корреляции
Генерация ключевой пары (t_{gen})	q	$y = a+b*\ln x$, $n = 25$, $d = 2$	$a = 2{,}5146$; $b = 0{,}8817$	0,9198
Зашифрование (v_{enc})		$y = a+b*\ln x$, $n = 25$, $d = 2$	$a = 0{,}1026$; $b = 0{,}0254$	0,9228
Расшифрование (v_{dec})		$y = a+b*\ln x$, $n = 25$, $d = 2$,	$a = -0{,}5758$; $b = 0{,}9045$;	0,9834
		$y = a*e^{b*x}$, $n = 25$, $d = 40$	$a = 2{,}1285$; $b = 3{,}2*10^{-3}$	0,9952
Генерация ключевой пары (t_{gen})	n	$y = a*x^{b}*e^{c*x}$, $q = 7$, $d = 2$	$a = 7*10^{-5}$; $b = 3{,}2088$; $c = 0{,}0278$	0,9988
Зашифрование (v_{enc})		$y = a+b*x^4$, $q = 7$, $d = 2$	$a = -0{,}0116$; $b = 4{,}1*10^{-5}$	0,9977
Расшифрование (v_{dec})		$y = a_i*x^{9-i}$, $i = 0$ч9, $q = 7$, $d = 2$	$a_0 = -0{,}47$, $a_1 = -0{,}106$, $a_2 = 1{,}03*10^{-5}$, $a_i = 0$, i=3ч9	0,9942
		$y = a*e^{b*x}$, $q = 7$, $d = 40$	$a = 4{,}4576$; $b = 6{,}3*10^{-3}$	0,9893