УДК 004.056.53 Вестник УрФО № 2(24) / 2017, с. 9–17

Дик Д. И.

МЕТОДЫ ОБНАРУЖЕНИЯ АНОМАЛИЙ В СИСТЕМАХ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ ДЛЯ ВЕБ-ПРИЛОЖЕНИЙ

С ростом популярности веб-сервисов все большая доля уязвимостей приходится на Веб-приложения. Обеспечение безопасности приложений может осуществляться как на этапе их разработки, так и на этапе эксплуатации. На этапе эксплуатации наиболее эффективными являются средства фильтрации трафика прикладного уровня, специально ориентированные на веб-приложения (Web Application Firewall, или сокращенно WAF). К сожалению являясь, по сути, сигнатурной системой обнаружения и предотвращения вторжений WAF требует создания и поддержания в актуальном состоянии большого количества правил. Кроме того как показывает практика правила детектирования часто требуют настройки на конкретное приложение.

Поэтому для более полной защиты WAF должны дополняться системами обнаружения аномалий.

К настоящему времени предложено ряд различных подходов к выявлению аномалий поведения пользователей веб-приложений, некоторые из которых описывается в данной статье.

Ключевые слова: *веб-безопасность, обнаружение вторжений, обнаружение аномалий, машинное обучение.*

Dik D. I.

METHODS ANOMALY DETECTION IN INTRUSION DETECTION SYSTEMS FOR WEB APPLICATIONS

With the growing popularity of Web services, an increasing number of vulnerabilities are accounted for by Web applications. Security of applications can be achieved both at the stage of their development and at the operational stage. At the operational stage, the most effective are application-level traffic filtering tools that are specifically targeted to Web applications (Web Application Firewall, or WAF for short). Unfortunately, WAF is a signature intrusion detection and prevention system. So WAF requires the creation and maintenance of a large number

of rules in the actual state. In addition, as practice shows, detection rules often require tuning to a specific application.

Therefore, for more complete protection, WAFs must be supplemented with anomaly detection systems.

To date, a number of different approaches have been proposed to identify anomalies in the behavior of users of web applications, some of which are described in this article.

Keywords: *web security, intrusion detection, anomaly detection, machine learning.*

Всемирная паутина фундаментальным образом изменила взаимодействие между людьми. В настоящее время веб технологии охватили практически все сферы человеческой жизни от коммерческой деятельности до частной жизни.

Как и любые другие виды информационных систем, веб-приложения подвержены большому количеству различных угроз.

Обеспечение безопасности веб приложений может осуществляться как на этапе их разработки, так и на этапе их эксплуатации. На этапе разработки безопасность приложений осуществляется путем использования специальных подходов к проектированию, конструированию и тестированию систем. На этапе эксплуатации путем использования дополнительных средств защиты, таких как межсетевые экраны, системы обнаружения и предотвращения вторжений, а также средств фильтрации трафика прикладного уровня, специально ориентированные на веб-приложения (Web Application Firewall, или сокращенно WAF).

Являясь специализированным средством защиты, WAF являются одним из самых эффективным подходов обеспечения безопасности веб приложений на этапе их эксплуатации. К сожалению, в классическом виде WAF по своей сути являются фильтрами на основе правил (сигнатур). Серьезным недостатком сигнатурных систем является значительная сложность создания и поддержания в актуальном состоянии фильтрующих правил, поскольку на практике часто наблюдаются значительные задержки в «поставке» разработчиками WAF новых правил для актуальных векторов атак и кроме того часто требуется адаптация правил под конкретные веб приложения. Как следствие практическое использование WAF требует очень высокой квалификации обслуживающих его специалистов.

Для обеспечения более полной защиты веб приложений WAF могут дополняться системами обнаружения аномалий.

Существуют различные подходы к выявлению аномального поведения в таких системах, которые могут быть разбиты на следующие группы:

– обнаружение аномалий в параметрах HTTP запроса,

– обнаружение аномалий пользовательских сеансов,

– обнаружение аномалий в SQL-запросах,

– контроль состояния веб-приложения.

Ниже приводятся варианты реализации методов выявления аномалий в таких системах.

1 Обнаружение аномалий в параметрах HTTP запроса

Данный подход[1,2,3] в качестве источника данных для обнаружения аномалий использует записи успешно выполненных HTTP запросов (код возврата больше либо равен 200 и меньше 300) в форме лог записей HTTP сервера.

Для каждого запроса осуществляется разбиение URI запроса на путь и набор параметров запроса.

Модель обнаружения аномалий предполагает вычисление оценок аномальности для каждого атрибута модели в отдельности, а также последующее вычисление комплексной оценки аномальности. Полученные оценки сравниваются с порогами определенными в процессе обучения. Если любая из полученных оценок принимает значение меньшее соответствующего ей порога в шаблоне аномальностей, то данный запрос рассматривается как потенциальная атака (чем ближе значение оценки к 0, тем выше аномальность).

Комплексная оценка аномальности выполняется по формуле

$$\text{Оценка аномальности} = \sum w_m (1 - p_m),$$

где wm – вес атрибута модели,

pm – вероятностная оценка атрибута модели.

При оценке аномальности предлагается использовать следующие атрибуты модели:

1) **Длина параметра запроса**

Использование данного атрибута при оценке аномальности основано на предположении, что значения параметров в нормальных запросах обычно не сильно различаются по длине. В тоже время вредоносные данные, часто нарушают это предположение.

В процессе обучения оцениваются выборочное среднее μ и дисперсия σ^2 длины параметра HTTP запроса. В процессе детектирования вероятность $p(l)$, того что параметр принимает значение l определяется по формуле основанной на неравенстве Чебышева

$$p(|x-\mu| > |l-\mu|) < p(l) = \frac{\sigma^2}{(l-\mu)^2}$$

Как возможная атака рассматриваются только строки с длинной большей, либо равной среднему значению, поэтому для строк короче среднего, $p(l) = 1$.

2) Распределение символов в параметре

Учет распределения символов мотивируется тем соображением, что значения параметров запроса обычно имеют регулярную структуру, являются удобочитабельными и содержат только печатные символы. Таким образом, можно ожидать, что значения для конкретного параметры будет иметь определенное распределения символов.

Под распределением символов понимаются относительные частоты для каждого из 256 символов ASCII-символов, отсортированных в порядке убывания. Например, для строки *passwd* символ *s* встречается два раза, символы *p, a, w, d* единожды, остальные символы не встречаются. В результате распределение символов принимает вид 0.33, 0.17, 0.17, 0.17, 0.17 и 0 оставшееся 251 значение.

В процессе обучения для каждого параметра запроса определяется идеализированное распределение символов, как среднее всех распределений для значений параметра.

Во время фазы обнаружения оценивается гипотеза, что распределение символов запроса соответствует идеализированному распределению. Оценка выполняется на основе критерия $\chi 2$ Пирсона. Для вычисления критерия распределения символов разбиваются на интервалы (предлагается использовать шесть интервалов: {[0], [1, 3], [4, 6], [7, 11], [12, 15], [16, 255]}).

2) Структура параметра

Атрибут модели обнаружения, оценивающий структуру параметров, вводится из соображения, что нормальные значения параметров часто можно рассматривать как строки, генерируемые с помощью регулярной грамматики. Например, подмножество имен логических путей может быть сгенерировано из регулярного выражения (/|[a-zA-Z0-9])+, то есть серии буквенно-цифровых символов, чередующихся с разделителями путей. Поскольку в реальности генерирующая грамматика для параметра неизвестна, требуется сформировать ее «разумное» приближение.

Для этого в процессе обучения набор параметров запросов рассматривается как результат вероятностной грамматики. Вероятностная грамматика – это грамматика, в которой каждому из генерируемых значений присваивается определенная вероятность, т.е. некоторые слова более вероятно будут сгенерированы граматикой, чем другие.

Вероятностная регулярная грамматика может быть преобразована в недетерминированный конечный автомат (НКА). Каждое состояние *S* автомата имеет набор n_s возможных выходных символов о, которые генерируются с вероятностью $p_s(o)$. Каждый переход *t* помечается вероятностью *p(t)*, характеризующей вероятность перехода (рис. 1). Такой автомат можно рассматривать как марковскую модель.

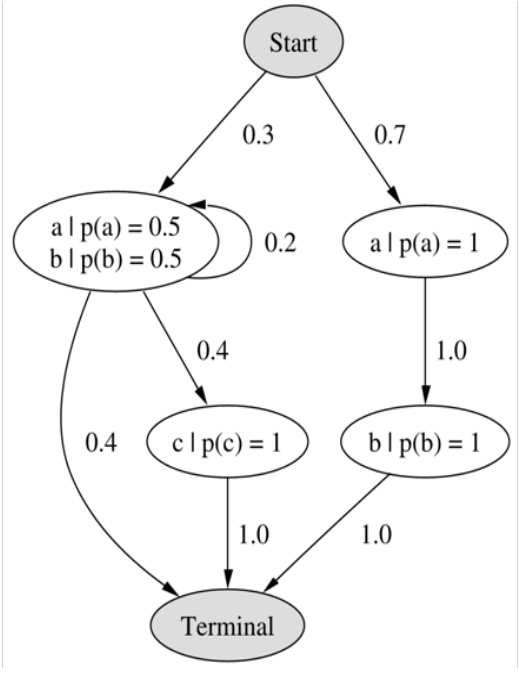

Рис. 1. Пример недетерминированного конечного автомата оценки структуры параметра

Выход марковской модели состоит из всех путей от начального до терминального состояний. Каждому результирующему слову

w (состоящему из цепочки символов o_1, o_2, …, o_k) присваивается значение его вероятности. Эта вероятностная величина вычисляется как сумма вероятностей всех различных путей через автомат, которые генерируют на выходе w. Вероятность одного пути является произведением вероятностей генерации символов $p_{Si}(o)$ и выбора переходов $p(t_i)$.

Целью процесса вывода структуры является поиск НКА, который имеет наибольшую вероятность для заданной обучающей выборки. Для формирования марковской модели из эмпирических данных предлагается использовать метод на основе теоремы Байеса[4].

После того, как модель Маркова построена, ее можно использовать на этапе обнаружения, чтобы определить вероятность параметров запроса. Вероятность параметра вычисляется по схеме, используемой на этапе обучения. Проблема в том, что даже легитимный ввод, который регулярно присутствовал на этапе обучения, может получить очень малое значение вероятности. Поэтому принимается, что модель возвращает значение 1, если значение параметра может быть результатом марковской модели и значение 0, когда значение параметра не может быть получено из данной грамматики.

3) Параметр запроса как идентификатор перечисления

Целью данного атрибута модели обнаружения является определение того, извлекаются ли значения некоторого параметра запроса из ограниченного набора возможных альтернатив (т.е. они являются идентификаторами перечисления).

Классификация параметра как перечисления или случайного значения основано на предположении, что если значение параметра случайно, то количество различных значений растет пропорционально общему количеству экземпляров параметра. В случае если такого роста не наблюдается, то значение параметра является перечислением.

С формальной точки зрения для определения является ли параметр запроса перечислением, вычисляется статистическая корреляция ρ между значениями функций f и g:

$$f(x) = x,$$

$$g(x) = \begin{cases} g(x-1)+1, & \text{если } x\text{-е значение параметра ранее не встречалось,} \\ g(x-1)-1, & \text{если } x\text{-е значение параметра встречалось ранее,} \\ 0, & \text{если } x = 0, \end{cases}$$

$$x = 1, \ldots, i,$$

где i – количество экземпляров параметра, участвующих в обучении.

Коэффициент корреляции ρ, вычисляется по формуле

$$\rho = \frac{cov(f, g)}{\sqrt{D[f]D[g]}},$$

где $cov(\cdot)$ – ковариация, $D[\cdot]$ – дисперсия.

Если ρ меньше 0, то f и g имеют отрицательную корреляцию и предполагается, что имеет место перечисление.

В случае если параметр классифицирован как перечисление, сохраняются значения идентификаторов параметра.

В процессе детектирования, если параметр является случайным, то всегда возвращается 1, если параметр – перечисление, то если значение параметра совпадает с одним из сохраненных значений, то возвращается 1, в противном случае 0.

4) Присутствие/отсутствие параметра

В данном случае предполагается, что отсутствие или аномальное присутствие одного или нескольких параметров в строке запроса может указывать на вредоносное поведение.

Для обнаружения таких ситуация в процессе обучения формируется таблица, содержащая перечень всех встречающихся подмножеств параметров запроса.

На этапе обнаружения алгоритм для каждого запроса выполняется поиск текущего набора параметров. Если искомый набор параметров встречался во время фазы обучения, возвращается 1, в противном случае – 0.

5) Контроль порядка следования параметров

Из-за внутренней логики работы программ легальные вызовы часто содержат одни и те же параметры в одинаковом порядке. При этом относительный порядок атрибутов обычно сохраняется даже тогда, когда некоторые параметры опускаются. Таким образом, нарушение порядка следования параметров может указывать на наличие вредоносного запроса.

Ограничения на порядок следования параметров запроса определяются во время фазы обучения.

Считается, что параметр a_s является предшествующим по отношению к параметру a_t, если оба параметра встречаются вместе, по крайней мере, в одном запросе и параметр a_s появляется раньше параметра a_t во всех обучающих запросах.

В процессе обучения для каждого запроса формируется ориентированный граф, вершинами которого являются параметры, а

дуги отражают отношения предшествования между параметрами. Полученный граф может иметь циклы, которые удаляются с помощью алгоритма Тарьяна. На основании полученного графа формируется список предшествований параметров, содержащий пары (a_s, a_t).

На этапе обнаружения для всех пар параметров запроса проверяется, удовлетворяют ли они ограничению предшествования, если ограничение выполняется, то возвращается 1, в противном случае 0.

6) Частота поступления запросов

Данный атрибут модели обнаружения учитывает частоту генерации запросов от всех пользователей сайта и от отдельного пользователя.

В процессе обучения обучающие запросы разбиваются на смежные интервалы одинаковой длительности (например, 10 секунд). Затем подсчитывается количество запросов определенного вида поступивших от всех пользователей и от каждого отдельного пользователя в пределах интервала. В результате формируются два распределения, для которых вычисляется среднее и дисперсия.

В фазе обнаружения время разбивается на интервалы той же длительности, которая использовалась в процессе обучения. Для каждого интервала подсчитывается количество поступивших запросов от всех пользователей и от каждого отдельного пользователя (с каждого IP адреса). Далее, также как при оценке длины параметра, с помощью неравенства Чебышева вычисляются две вероятности, оценивающие нормальность частоты поступления запросов. Далее найденные вероятности усредняются.

7) Временная задержка между запросами

Данный атрибут модели обнаружения направлен на выявление регулярных интервалов между поступающими запросами, что может свидетельствовать о факте зондирования или атаки, поскольку действия пользователей по своей природе характеризуются значительной не регулярностью.

На этапе обучения формируется распределение «нормальных» временных задержек между последовательными клиентскими запросами. Для этого вычисляются интервалы между запросами каждого пользователя.

Далее формируется шкала временных интервалов и подсчитывается количество попаданий временных задержек в каждый интервал (формируется гистограмма распределения).

На этапе обнаружения определяются временные задержки между последовательными запросами от каждого пользователя. И с помощью критерия $\chi 2$ Пирсона оценивается, удовлетворяет ли распределение задержек запросов от пользователя эталонному распределению задержек, полученному на этапе обучения.

Поскольку надежность результата оценки зависит от размера выборки (количества запросов) результат оценки аномальности масштабируется путем умножения его на весовой коэффициент в пределе, с увеличением количества наблюдаемых запросов, принимающий значение 1.

8) Порядок обращений в сеансе

Данный атрибут модели обнаружения пытается оценить регулярность структуры сеанса, который может быть связан с конкретными учетными данными (например, cookie). Анализ порядка обращений подобен оценке структуры параметра. Разница в том, что этот атрибут модель анализирует структуру последовательности запросов, а не синтаксис параметров запроса. Данный атрибут предназначен для обнаружения атак на логику приложения, например, когда злоумышленник пытается обойти авторизацию и получить доступ к привилегированной области системы напрямую.

На этапе обучения запросы к системе группируются в соответствии с IP-адресом источника запроса, идентифицируя, таким образом, сеансы взаимодействия пользователей с системой. В сеанс агрегируют только близкие по времени запросы. Далее формируется недетерминированного конечный автомат, представляющий все «законные» последовательности запросов.

На этапе обнаружения полученный запрос *q* связывается с соответствующим сеансом *S*. Если запрос *q* может быть частью сеанса *S* (является допустимым с точки зрения конечного автомата построенного на этапе обучения и предыдущих полученных запросов), то в результате возвращается 1, противном случае возвращается 0.

2 Обнаружение аномалий пользовательских сеансов

Обнаружение аномалий сеансов5 основано на предположении, что последователь-

ность посещаемых пользователями страниц веб-приложения будет иметь типовую структуру.

Метод обнаружение аномалий сеанса, извлекает веб-сеансы из журналов веб-сервера и в фазе обучения формирует профили «нормальных» последовательностей страниц, на основе которых вычисляется вероятность выполнения определенных последовательностей запросов. В фазе детектирования вычисляется оценка аномальности, и сеансы, для которых превышаются пороговые значения, обозначаются как ненормальные (пороги аномальности зависят от конкретного сайта).

Размеры веб-сессий, различаются по длине, поэтому каждая сессия дополнительно разбивается на подсессии с фиксированной длиной (рис. 2).

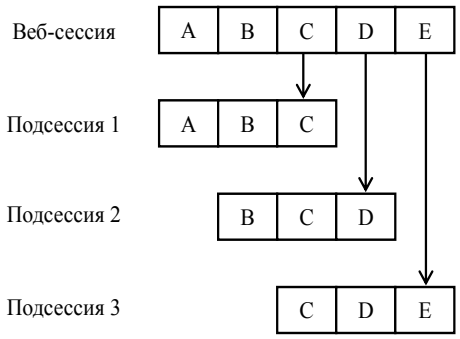

Рис. 2. Разбиение содержащей 5 страниц веб-сессии на подсессии по 3 страницы

Каждая подссессия разбивается на две части: префикс S_p, содержащий все страницы подсессии кроме последней, и суффикс S_p, содержащий последнюю страницу подсессии. Подсессии с одинаковыми префиксами в процессе обучения группируются в обучающие выборки.

Вероятность нормальности сеанса вычисляется с помощью байесовского метода оценки[6]:

$$C(D,L) = \sum_{k=k^0}^{L} \frac{k^0\alpha + N}{k\alpha + N} P(k|D),$$

$$P(X^{N+1}|D) = \begin{cases} \dfrac{\alpha + N_i}{k^0\alpha + N} C(D|L), & \text{если } i \in \Sigma^0, \\ \dfrac{1}{n - k^0}(1 - C(D|L)), & \text{если } i \notin \Sigma^0, \end{cases}$$

$$k^0 = |\Sigma^0|,$$

где $P(X^{N+1}|D)$ – оценка вероятности посещения заданной цепочки страниц, основанная на обучающей выборке;

X – случайная переменная, принимающая значение из множества возможных подсессий с заданным префиксом (или иначе говоря принимающая значение из множества возможных суффиксов при заданном префиксе);

D – обучающая выборка, содержащая N (возможно повторяющихся) подсессий X^1, …, X^N (с заданным префиксом);

L – количество возможных уникальных подсессий (при заданном префиксе);

N_i – сколько раз встречается i-я уникальная подсессия в обучающей выборке;

Σ^0 – множество наблюдаемых подсессий (при заданном префиксе);

$C(D, L)$ – масштабирующий фактор, который можно рассматривать как вероятность того, что в обучающей выборке присутствуют все возможные подсессии, $(1 – C(D, L))$ – вероятность появления новых подсессий.

3 Обнаружение аномалий в SQL-запросах

В настоящее время предложено большое количество подходов к обнаружению аномалий в SQL-запросах, которые можно разбить на три группы[7]:

– методы, при которых профиль формируется путём синтаксического анализа текста SQL запроса;

– методы, при которых профиль нормального поведения определяется в результате семантического анализа SQL запроса;

– методы, учитывающие различные характеристики запроса (лексические, темпоральные, ресурсные и др.).

В качестве примера рассмотрим один из методов, предложенный специально для веб-приложений[8].

В данном методе система обнаружения вторжений встраивается в канал связи между веб-приложением и сервером базы данных. Схема системы приведена на рис. 3.

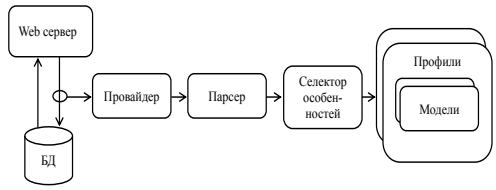

Рис. 3. Схема системы обнаружения аномалий в SQL-запросах

Провайдер событий отвечает за перехват SQL-запросов от веб-приложения и перенаправление их системе обнаружения вторжений. Провайдер событий также извлекает другую важную информацию о процессе, такую как, например, имя файла сгенерировавшего SQL запрос скрипта.

Перехваченные SQL-запросы перенаправляются в парсер, который анализирует

входящий SQL-запрос и создает его высокоуровневое представление. Запрос представляется как последовательность лексем. Каждой лексема присваивается флаг, который указывает, является ли лексема литералом или нет. Литералы – это единственные элементы SQL-запроса, которые могут содержать вводимые пользователем данные. Лексемы, представляющие имена полей базы данных, дополняются атрибутом типа данных, который автоматически извлекается из базы данных и в дальнейшем может быть дополнительно скорректирован пользователем (например, поле в базе данных может иметь тип varchar, что подразумевает произвольные строковые значения, однако пользователь может уточнить этот тип как XML).

Для литералов также выводятся их типы с использованием следующих правил:

– литералу, который сравнивается с полем базы банных, присваивается тип данных поля базы данных, с которым он сравнивается;

– литерал, вставляемый в таблицу базы данных, получает типа поля данных, в которое он вставляется.

После этого селектор особенностей преобразует запрос в форму подходящую для дальнейшей обработки и определяет, к какому профилю относится запрос. Для этого создается список всех являющихся литералами лексем в порядке их появления в запросе и формируется скелет запроса путем замены всех вхождений литеральных лексем в запросе маркерами (при этом значение литерала заменяется на пустое). Поскольку пользовательский запрос должен изменять только значения литеральных лексем, скелет SQL-запроса полностью описывает его структуру, и пользовательские запросы с разными данными должны приводиться к одному и тому же скелету.

Дальнейшие действия зависят от состояния системы обнаружения вторжений. Система может находиться в одном из трех состояний: обучения, настройки порогов обнаружения аномалий и детектирования аномалий.

Если система находится в режиме обучения, имя генерирующего запрос файла скрипта и скелетный запрос, используются в качестве ключей для поиска профиля. Профиль представляет собой набор статистических моделей выявления аномалий. Если профиль не найден, создается новый профиль. Если профиль уже существует, то происходит обновление моделей выявления аномалий.

Статистические модели выявления аномалий предназначены для контроля значений литеральных лексем и зависят от их типа. В качестве моделей выявлений аномалий для строковых литералов, по аналогии с приведенным ранее описанием методов анализа параметров входного запроса, предлагаются следующие статистические модели:

– длина строки,

– распределения символов в строке,

– проверка совпадение на с префиксом и суффиксом строки,

– контроль структуры строки.

Для строковых и не строковых типов предлагается также контроль литерала как идентификатора перечисления.

Если система находится в режиме настройки порогов обнаружения аномалий, соответствующий профиль просматривается так же, как и в режиме обучения, но обновления моделей не происходит. Вместо этого модели используются для вычисления оценки аномальности, которая определяет, насколько хорошо список литеральных лексем соответствует моделям. Совокупный балл рассчитывается как сумма отрицательного логарифма каждой индивидуальной модели. Для каждого профиля регистрируется наивысшая совокупная оценка аномальности, наблюдаемая в течение фазы настройки порогов. Если для события не найден профиль, выводится предупреждение, указывающее, что фаза обучения не была завершена.

В режиме детектирования рассчитывается оценка аномальности. Сигнал тревоги генерируется, если оценка аномальности превышает пороговое значение. Сигналы тревоги также генерируются, если не найден профиль события, или если SQL-запрос вызывает ошибку синтаксического анализа.

4 Контроль состояния веб-приложения

К методу контроля состояния веб-приложения относится подход получивший название *Swaddler*[9].

В данном подходе под состоянием веб-приложения в определенный момент времени понимается информация, связанная с сеансом пользователя. Часть этой информации хранится на сервере, а часть ее может передаваться между браузером пользователя и сервером в виде файлов *cookie*, скрытых полей формы и параметров запроса. В пилотной реализации *Swaddler* в качестве параметров состояния приложения используется

переменная *PHP $_SESSION*, представляющая собой ассоциативный массив, содержащий переменные сессии, доступные для текущего скрипта.

Предлагается ассоциировать каждую инструкцию приложения с моделью состояния, в котором эта инструкция обычно выполняется. Например, код, содержащийся в каталоге администратора приложения, может предполагать, что при его выполнении переменная *$_SESSION['username']* всегда должна быть равна admin. Любое нарушение во время выполнения этого требования означает, что пользователь с низким уровнем привилегий получил доступ к административной функции, минуя ограничения, реализованные разработчиком приложения.

В идеале, полный набор этих взаимосвязей между точками выполнения кода и переменными должен предоставляться разработчиками в рамках спецификации приложения. Однако, в реальности данное требование невыполнимо, поэтому модели нормального состояния для каждой программной инструкции должны быть выведены путем анализа трасс выполнения приложения. Для этого предлагается автоматически привязать веб-приложение к трассировщику коду, позволяющему извлекать значений переменных состояния во время выполнения приложения.

Из соображений оптимизации состояние веб-приложения может контролироваться не для каждой инструкции, а для первой инструкции каждого блока, представляющего собой последовательность операторов без внутренних ветвлений. Фактически, поскольку поток управления внутри блока представляет собой простую последовательность команд без ветвей, состояние приложения в начале базового блока однозначно определяет состояние каждой команды внутри блока.

Swaddler состоит из двух основных компонентов: датчика и анализатора. Датчик собирает данные состояния приложения (то есть значения переменных состояния) в начале каждого блока и инкапсулирует их в события, которые отправляются в анализатор. Событие, созданное датчиком, сопоставляет имена переменных и их текущие значения.

Для каждого блока приложения анализатор поддерживает профиль, т.е. набор статистических моделей, используемых для контроля некоторых характеристик переменных состояния. Эти модели позволяют, как контролировать свойства отдельных переменных, так и описывать сложные отношения между несколькими переменными состояния.

Формирование профилей блоков происходит в режиме обучения. В режиме обнаружения полученные профили используются для идентификации аномальных состояний приложения. Когда обнаружено аномальное состояние, анализатор генерирует предупреждающее сообщение и может, необязательно, остановить выполнение приложения.

Для контроля состояния отдельных переменных предлагается использовать следующие статистические модели:

– модель идентификатора перечисления: проверка того, что значение переменной извлекается из небольшого количества альтернатив,

– модель длины значения строковой переменной,

– модель распределения символов в значении строковой переменной.

Для контроля связей между переменных предлагается использовать следующие статистические модели:

– наличие или отсутствие переменных,

– вероятностные инварианты.

Цель модели контролирующей наличие или отсутствие переменных заключается в том, чтобы определить, какие переменные должны всегда присутствовать при доступе к блоку приложения.

На этапе обучения модель отслеживает, какие переменные всегда устанавливаются при доступе к определенному блоку кода. Основываясь на этой информации, каждой переменной состояния, связанной с блоком, присваивается вес, где переменные, которые всегда присутствуют, получают вес 1, а переменные, которые иногда отсутствовали, получают вес в диапазоне от 0 до 1, в зависимости от количества раз, когда переменная присутствовала. Общий балл для блока вычисляется как сумма всех оценок переменных, деленная на количество переменных в блоке. Эта оценка всегда находится между 0 и 1.

На этапе обнаружения общий балл блока рассчитывается на основе установленных весов. Следовательно, отсутствие переменной с более высоким весом приводит к более низкой оценке состояния, связанного с блоком.

Вероятностный инвариант – это утверждение (инвариант), для которого не гарантируется, что оно всегда истинно во всех воз-

можных исполнениях программы. Для автоматического обнаружения и извлечения вероятностных инвариантов, связанных с состоянием, используется система динамического детектирования вероятностных инвариантов *Daikon*[10].

Примечания

1. Kruegel, C. Anomaly Detection of Webbased Attacks [Text] / C. Kruegel, G. Vigna // Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS '03). – Washington, 2003. – P. 251–261.

2. Kruegel, C. A multi-model approach to the detection of web-based attacks [Text] / C. Kruegel, G. Vigna, W. Robertson // Computer Networks: The International Journal of Computer and Telecommunications Networking. – 2005. – Vol. 48, № 5. – P. 717–738.

3. Robertson, W. Using generalization and characterization techniques in the anomaly-based detection of web attacks [Text] / W. Robertson [et al.] // Proceedings of the 13th Symposium on Network and Distributed System Security (NDSS), 2006.

4. Stolcke, A Hidden Markov model induction by Bayesian model merging [Text] / A. Stolcke, S. Omohundro // Advances in Neural Information Processing Systems. – 1993. – Vol. 5

5. Cho, S. SAD: web session anomaly detection based on parameter estimation [Text] / S. Cho, S. Cha. // Computers & Security. – 2004. – Vol. 23. – P. 312–319.

6. Friedman, N. Efficient Bayesian parameter estimation in large discrete domains [Text] / N. Friedman, Y. Singer // Advances in neural information processing systems. – Cambridge, 1999. – Vol. 11.

7. Григоров, А.С. Обзор методов обнаружения аномалий в SQL-запросах к базам данных [Текст] / А.С. Григоров // Современные тенденции технических наук: материалы Междунар. науч. конф. (г. Уфа, октябрь 2011 г.). – Уфа: Лето, 2011. – С. 13–17.

8. Valeur, F. A learning-based approach to the detection of SQL attacks [Text] / F. Valeur, D. Mutz, G. Vigna // Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA). – Vienna, 2005. – P. 123–140.

9. Cova, M. Swaddler: An Approach for the Anomaly-based Detection of State Violations in Web Applications [Text] / M. Cova [et al.] // Recent Advances in Intrusion Detection (RAID). – Australia, 2007. – Vol. 4637. – P. 63–86.

10. Ernst, M.D. The Daikon system for dynamic detection of likely invariants [Text] / M.D. Ernst [et al.] // Science of Computer Programming Science of Computer Programming. –2007. – Vol.69, № 1-3. – P. 35–45.

ДИК Дмитрий Иванович, кандидат технических наук, доцент кафедры «Безопасность информационных и автоматизированных систем» Курганского государственного университета. E-mail: ddi@kgsu.ru

DIK Dmirty Ivanovich, Candidate of Engineering Science, Associate Professor of the Chair «Security of Information and Automation Systems», Kurgan State University. E-mail: ddi@kgsu.ru