УДК 004.056.53.001.3

Соколов Артем Михайлович

Современные модели обнаружения аномалий в компьютерных системах

Введение

В последнее время мир убедился, что даже самые надежные системы защиты не способны защитить от атак на компьютерные системы государственных и коммерческих учреждений. Одна из причин — в том, что в большинстве систем безопасности применяют стандартные механизмы защиты: идентификацию и аутентификацию, механизмы ограничения доступа к информации согласно прав субъекта и криптографические механизмы. Это традиционный подход со своими недостатками: незащищенностью от собственных пользователей-злоумышленников, размытостью разделения субъектов системы на "своих" и "чужих" из-за глобализации информационных ресурсов, сравнительной легкостью подбора паролей вследствие использования их смысловой разновидности, снижением производительности и усложнением информационных коммуникаций вследствие ограничения доступа к ресурсам организации. Отсюда потребность в механизмах, которые, дополняя традиционные, делали ли бы возможным обнаружение попыток несанкционированного доступа и информировали об этом ответственных за безопасность или реагировали в ответ. Важно, чтобы такие системы могли противостоять атакам, даже если злоумышленник уже был аутентифицирован, авторизован и с формальной точки зрения соблюдения прав доступа имел необходимые полномочия на свои действия.

Эти функции и выполняют системы обнаружения вторжений (*intrusion detection systems*, *IDS*). Поскольку предусмотреть все сценарии развертывания событий в системе с активным "чужим" субъектом невозможно, надо или как можно детальнее описать возможные "злоумышленные" сценарии или же, наоборот, — "нормальные" и постулировать, что любая активность, не подпадающая под принятое понимание "нормальности", является опасной. Согласно этому, IDS подразделяются на системы, реагирующие на известные атаки — системы

обнаружения злоупотреблений (*misuse detection systems, MDS*) и системы обнаружения аномалий (*anomaly detection systems, ADS*), регистрирующие отклонение эволюции системы от нормального хода.

Системы обнаружения злоупотреблений. Атака является многошаговым процессом, и его осуществление требует высокой квалификации хакера, поэтому простейшим путем взлома или приведения системы в недееспособное состояние является применение "эксплойтов", то есть уже написанных модулей, реализующих необходимые этапы атаки. Огромное количество эксплойтов доступно в Интернет, что привело к распространению именно такого способа атаки. Вследствие этого последовательность аудит-событий, реализующая атаку, часто является фиксированной.

Работа MDS базируется на составлении шаблонов таких атак. Их уровень абстракции может быть простым, например наличие определенных значений в заголовке сетевого пакета или последовательности команд в файле аудита, так и сложным, например прохождение траектории системы в пространстве состояний через определенные опасные состояния. Защитные системы такого типа эффективны на известных схемах атак, однако в случаях неизвестной атаки или отклонений хода атаки от шаблона возникают проблемы, поэтому надо поддерживать большую базу данных на каждую атаку и ее вариации, а также организовывать непрерывное пополнение баз шаблонов.

Системы обнаружения аномалий. Основным предположением ADS является то, что действия злоумышленника (события в атакованной системе) обязательно чем-то отличаются от поведения обычного пользователя (от событий в нормальном состоянии), то есть являются аномалиями. Потому такие системы способны регистрировать и неизвестные атаки. Работе ADS предшествует период накопления информации, когда строится концепция нормальной активности системы, процесса или пользователя. Она становится эталоном, относительно которого оцениваются последующие данные. Здесь определяется оптимальное количество факторов, учитывая которые, будут вестись наблюдения. Их совокупность не должна быть слишком большой, чтобы не снизить производительность работы в целом, или слишком ограниченной, потому, что по недостаточно исчерпывающим характеристикам невозможно будет построить профиль нормального поведения.

Возможны два общих вида ошибок: а) нормальное поведение системы или пользователя ошибочно принимается за злоумышленное (*false positives*); б) попытка злоумышленного проникновения в систему принимается за нормальную активность (*false negatives*). Хотя ни одна из этих ситуаций нежелательна, вторая — более опасна, а потому одной из основных задач построения ADS есть четкое определение условий, при которых ситуация воспринимается как аномальная, так, чтобы ни одна из перечисленных ситуаций не возникала слишком часто [1].

Модели обнаружения аномалий

Пионерской работой в этой отрасли была работа [2], в которой приводились основные понятия и подходы к проблеме. Позднее в литературе появилось много попыток построения ADS. Большинство из них — концептуальные модели, цель которых — проверить возможность применения математической модели или подхода. Коммерческих продуктов в области IDS очень мало, а те, что есть, почти никогда не выходят за рамки MDS.

Практически все описанные в литературе модели для обнаружения аномалий можно разделить на: а) базирующиеся на хранении примеров поведения; б) частотные; в) нейросетевые; г) строящие конечные автоматы; д) другие, специальные.

Методы, базирующиеся на хранении примеров поведения. Простейшим подходом есть прямое запоминание примеров действий, последовательностей команд пользователей или вообще любых параметров доступных регистрации (*instance-based learning*). Несмотря на невозможность применения этого подхода в других случаях моделирования человеческого поведения, в задачах обнаружения вторжений он является достаточно действенным, что обусловлено ограниченным количеством возможных действий субъектов компьютерной системы, значительной детерминированостью задач, которые можно выполнять, и самой структурой операционной системы.

В работе [3] проводится аналогия с гомеостазом — процессом в живых организмах, который поддерживает их в жизнедеятельном состоянии. Реакцией "организма" (компьютерной системы) на аномальное поведение процесса есть его принудительное замедление. Таким образом, процессы, демонстрирующие активное аномальное поведение, будут почти остановлены и автоматически уничтожены системой как не реагирующие на запросы. Зафиксированные ранее подпоследовательности системных вызовов, которые встречались в тренировочном множестве, запоминаются, а во время работы проверяется их наличие в текущей сессии. Как и в [4], утверждается, что поскольку наблюдения ведутся за системными вызовами программы, то из-за их высокой регулярности (последовательность вызовов и их типы значительно детерминированы исходным кодом программы) размеры базы подпоследовательностей не будут большими. Подпоследовательность из текущей сессии, которой не было среди тренировочных, считается аномальной. Подход нуждается в дописывании ядра операционной системы, что является нелегкой и не всегда возможной задачей. Кроме того, постоянное присутствие такого мониторингового компонента приводит к общему замедлению работы всей системы, которое составляет 4% – 50% [3].

Другим примером instance-based системы есть [5]. Вводится специальная метрика сходства символьных последовательностей. Так, для последовательностей X и Y их сходства

$Sim(X,Y) = \sum_{i=0}^{l-1} w(X,Y,i)$, где $w(X, Y, i) = 1+w(X, Y, i-1)$, если $x_i=y_i$, и $w(X, Y, i) = 0$, если $x_i \neq y_i$ или

i<0. Для каждого пользователя хранится множество ***D*** последовательностей, которые он обычно выполняет. Степенью сходства множества ***D*** и последовательности, поступившей для проверки, принимается величина $Sim_D(X) = \max_{Y \in D}\{Sim(X,Y)\}$. Из-за необходимости сохранения большого количества данных на каждого пользователя возникает потребность в применении специальных методов уменьшения объема баз последовательностей. Авторы рассматривают два таких метода: а) выборочной селекции примеров, когда сохраняются не все примеры последовательностей, а только последние *n* или все, кроме *n* с наименьшими вероятностями, и б) преобразования базы на базу представителей классов элементов.

Из-за высоких требований к ресурсам, instance-based системы могут применяться только в задачах обнаружения аномалий с незначительным количеством возможных сигналов и в основном со статическим поведением субъектов наблюдения.

Частотные модели. Развитием идей instance-based систем есть принятие во внимание частотного распределения параметров системы. Уже в [2], с целью регистрации, предлагалось сохранять информацию о субъектах в шаблонах активности — выраженных в статистических терминах наборах характеристик поведения субъекта относительно определенного объекта, (вхождение в систему, запуски программ, доступы к файлам и устройствам). Затем проверяется, попадает ли относительное количество определенных событий в заданный экспертом интервал.

Модификацией частотного подхода есть работа [6], где предлагается метод, базирующийся на т.н. структурных нулях. Он заключается в использовании информации о командах, которые встречаются очень редко или совсем не встречаются (соответствующие им ячейки в таблице вероятностей равняются нулю, то есть являются структурными нулями). Вводится индекс уникальности $\frac{1}{N_i}\sum_{c \in \Sigma} W_{ic} U_c N_{ic}$, который вычисляется для каждой сессии и каждого пользователя *i*. Здесь N_i — количество команд в сессии, W_{ic} — индикатор {1,-1} присутствия команды *c* в тренировочных данных пользователя, N_{ic} — количество вхождений команды *c* в текущую сессию и U_c — параметр, изменяющийся от 0 для команды, которой использовались все

пользователи, до 1 для команды, которой не пользовался ни один пользователь. Этот индекс получает положительный прирост для частых команд в пределах текущей сессии, который тем меньше, чем чаще эта команда используется вообще. Так, широкое использование в общем редких команд обусловит большие значения индекса уникальности. В случае появления команд, не свойственных пользователю, индекс уменьшается. Предположив, что значение индекса является стабильным для данного субъекта, авторы стараются различать их за значениями индекса.

Распространенными недостатками частотных методов является неадаптивность, поскольку часто эталонные значения частот определяются одноразово, по тренировочным множествам или по экспертным данным, и неучитывание последовательностей выполнения команд.

Нейросетевые модели. Применение нейронных сетей обусловливается самой неформальной постановкой задачи — обнаружить аномальное поведение. Идея состоит в том, чтобы, получив некоторое "тренировочное" множество параметров вход-выход, характеризующее поведение системы, дать сети "привыкнуть" к ним. Выходом может быть некоторый коэффициент "нормальности" поведения или один из параметров системы. Если исходные данные имеют закономерности, то делается предположение, что сеть способна "научиться" на них. Если в процессе работы предложенный нейронной сетью выход, приусловии что он является некоторым коэффициентом, попадает в опасную область или отличается от имеющегося в реальной системе, если это один из параметров системы, то делается вывод, что в системе имеется аномалия.

В работе [7] для построения шаблона поведения пользователя используются такие параметры: время, когда он обычно работает, набор узлов, с которых он начинает рабочую сессию, характеристики использования ресурсов системы и т. п. Эти параметры оцифровываются и служат входом в нейронную сеть обратного распространения ошибки (*backpropagation neural network*, *BPNN*), а выходом является коэффициент, равный нулю для пользователя с нормальным поведением и равный единице — с аномальным. Иными словами, сеть тренируется на парах типа ("нормальные" параметры, 0) и ("аномальные" параметры, 1). Поскольку для получения "ненормального" поведения надо было бы вынудить пользователя вести себя не так, как он

привык, то аномальные данные генерируются случайно, что осложняет интерпретацию результатов относительно работы на реальных данных.

В работе [8] предложенная система идентифицирует пользователей на основе ограниченного количества команд (всего 100). Учитывается количество запусков каждой команды, а не их последовательность. Эти определенным способом закодированные количества служат входом в BPNN, на которых она тренируется с учителем. Выход — идентификатор пользователя (параметр, который, хотя и имеет в UNIX числовое значение, создает искусственную близость пользователей с близкими идентификаторами). Если пользователь идентифицирован неправильно — на это обращается внимание администратора. Были получены неплохие результаты, но только с 10 пользователями. Такие условия являются "тепличными", поскольку в реальных системах количество пользователей достигает нескольких тысяч, причем большинство выполняет однотипные действия, что осложняет их различение предложенным способом.

В нашей предыдущей работе [9] тоже был предложен подход на основе BPNN. В отличие от [7,8], мы не ограничивались узким кругом пользователей, количество команд было большим (до 512 команд) и не использовали искусственных данных. Сеть применялась для предсказания следующей ($n+1$)-ой команды в сессии пользователя, используя предыдущие n команд. Из-за неупорядоченного характера множества команд было предложено их компромиссное кодирование: каждой команде ставился в соответствие вектор — двоичная запись номера команды после произвольного перенумерования всех команд. Таким образом, мы отчасти избавлялись от проблемы искусственной упорядоченности, когда командам ставили в соответствие скалярные значения и получали довольно экономный метод кодирования (в отличие от метода, когда команде присваивается вектор с одной единицей в фиксированной позиции). После тренировки на входах, отвечающий контексту предыдущих нескольких команд, которые попадали в окно ширины n и с выходом — следующей командой, сеть должна была на реальных данных по имеющемуся контексту определять следующую. Если относительное количество правильно определенных команд в сессии не превышало заданный уровень, это сигнализировало об аномалии. Для

обеспечения адаптивности нейронная сеть после каждой сессии дотренировывалась. Результаты свидетельствуют о возможности применения такого подхода и об эффективности выбранного метода кодирования команд, однако нерегулярность поведения пользователей значительно повышает уровень ошибок типа false positives.

Мы считаем, что лучшие результаты могут быть получены на данных от более регулярных источников, таких как системные процессы. Дополнительные примеры применения BPNN и рекуррентных нейронных сетей для обнаружения аномалий приведено в [10-12]. Неисследованными остаются возможности применения нейронных сетей неперсептронного типа, например ансамблевых.

Недостатком многих нейронних сетей является их плохая приспособленность для работы с неупорядоченными величинами. Введение искусственного порядка на множестве значений элементов только исказит картину, поскольку нейронная сеть будет "учитывать" близость числовых величин.

Модели, строящие конечные автоматы. В этом подходе достигается бóльшая моделирующая способность, чем при использовании тривиальных частотных и instance-based методов. Исходные данные рассматриваются как поток дискретных событий, например системных вызовов или идентификаторов процессов. Цель — получить автомат, который моделирует указанную последовательность событий. Для многих последовательностей характерно то, что вероятность следующего символа, элемента или сигнала зависит от предыдущих элементов. Часто они зависят только от небольшого количества предыдущих. Это наталкивает на мысль моделировать их при помощи марковских цепей [2,4,13-15]. Однако при увеличении порядка цепи, которое может существенно увеличить точность модели, количество состояний соответствующего автомата ведет себя как $O(\Sigma^L)$, где Σ — размер алфавита символов, а L — порядок цепи. Это выдвигает большие требования к ресурсам и увеличивает время обработки.

Работы [13,15] примеры простейшего применения марковских цепей в моделировании процессов в компьютерной системе. По исходным данным строится матрица переходов цепи

первого порядка, а вероятность сессии определяется как произведение вероятностей перехода между состояниями, отвечающими элементарным событиям в файле аудита.

Более мощной моделью есть т.н. скрытые марковские модели (HMM). От марковских цепей они отличаются тем, что выходные символы автомата не детерминируются его состояниями, а зависят от них стохастично. Есть теоретические результаты, свидетельствующие о сложности обучения HMM. Кроме того, часто необходим подбор количества скрытых состояний, который обычно проводится вручную и после тренировки только некоторые из них являются реально задействованными (такими, которые имеют отличные от нуля вероятности переходов). Подход к обнаружению аномалий на основе HMM предлагается, например, в работе [16]. Для обеспечения адаптивности необходимо периодически перенастраивать HMM на новых данных, что являются значительным недостатком модели.

В работе [17] конечный автомат строится по *n*-граммам событий из аудит-файла, который не содержит аномальных данных. В первом варианте построения каждому состоянию отвечает одна или несколько *n*-грамм. Несколько *n*-граммы могут быть присвоены одному состоянию, если при последовательной обработке данных, т.е. окно за окном, имеем ситуацию, когда следующая *n*-грамма еще не имеет своего состояния и одновременно не существует ребра, который исходит из текущей *n*-граммы в следующую. Тогда следующая *n*-грамма присваивается тому же состоянию, что и текущая. Построенный таким образом недерминированный автомат может находить ранее не встречавшиеся события, однако он не принимает во внимание статистические параметры последовательности. Согласно второму способу строится вероятностный автомат со множеством входов — событиями из аудит-потока и множеством выходов — последовательностью чисел, указывающей на степень аномальности исходных событий. Эти числа показывают, как вероятность имеющегося перехода из текущего состояния в следующее соотносится с вектором переходных вероятностей из этого состояния. Несмотря на некоторую необоснованность выбора именно такой меры аномальности, подход при экспериментальной проверке оказался эффективным.

Совершенствуя модель для обнаружения аномалий по системным вызовами (вспомним фиксированность исходного кода программы и, из-за этого, определенную статичность картины генерирующихся системных вызовов), можно не учиться на примерах, а попытаться, используя исходный код, задать статическую грамматику, описывающую некоторый язык вызовов. В работе [18] такую грамматику задано путем анализа исходных текстов программ. Предложено три варианта моделей: графа вызовов (недетерминированный конечный автомат), стековую модель (контекстно-свободную грамматику) и модель, похожую на описанную в работе [4] (для данной k-граммы проверяется, принадлежит ли она к языку, порожденному определенной грамматикой). Преимуществом такого подхода есть отсутствие ошибок типа false positives из-за зависимости построенных моделей непосредственно от исходного кода программы, т.е. несовместимая с моделью последовательность вызовов сразу сигнализирует об отклонении работы программы от ее исходного кода, и только такие отклонения будут сигнализировать об аномалии, а все, что предусмотрено текстом программы, будет принято. Основным недостатком подобного подхода является сложный процесс построения модели анализируя тексты программы. Кроме того, есть ограничения на виды программ, которые могут быть описаны статическими грамматиками [18].

Модель, не имеющая недостатков HMM и марковских цепей с фиксированным порядком, была предложена нами в работе [19]. Ее основная идея — использовать модель марковских цепей переменного порядка [20], поскольку следующий символ или сигнал в последовательностях, генерирующихся в компьютерных системах, редко определяется предыдущим контекстом постоянной длины. Учет только существенных контекстов позволяет обойти экспоненциальный рост требований к ресурсам, не ухудшая точность модели. Кроме того, большинство предыдущих подходов к моделированию поведения является неадаптивными. Наилучшее, что предлагается, — периодическая перенастройка модели, а отсюда — затраты времени и уменьшение ее надежности с течением времени, которое проходит после очередной перетренировки. Для поддержки реальной адаптивности мы применили определенную процедуру изменения параметров модели так, что последняя полученная информация становилась более весомой [19]. При этом сохраняются

аппроксимирующие свойства исходной модели. Единственной известной нам работой, где был применен похожий на наш метод адаптивной настройки вероятностей, является [13]. Однако там в качестве основы служила простейшую марковская модель первого порядка, а поэтому не учитывался контекст длины большей, чем одна команда.

Другие методы. Как пример подхода на основе data-mining техник вспомним работу [21], где их применяют для выделения из массива данных ассоциативных правил и т.н. частых эпизодов. Ассоциативные правила — это выражения вида $[\mathrm{X} \rightarrow \mathrm{Y}, \mathrm{c}, \mathrm{s}]$, где X, Y — подмножества множества значений на множестве аттрибутов, $s = Support(X \cup Y)$ — процент записей в базе, которые содержат $X \cup Y$, $c = \frac{Support(X \cup Y)}{Support(X)}$. Тогда правило вида $[\mathrm{lynx} \rightarrow \mathrm{www.google.com}, 0.2, 0.4]$ означает, что пользователь в 40% случаев, когда он пользуется интернет-браузером `lynx`, заходит на сайт `www.google.com`, и посещение этого сайта составляет 20% от общей его активности. Авторы применили метод частых эпизодов и некоторые специальные модификации общих data-mining техник, чтобы приспособить их для обнаружения аномалий. В процессе работы новые правила добавляются, пока множество правил не стабилизируется, то есть по исходным данным уже нельзя получить новых данных, и обучение прекращается.

В работе [22] для обнаружения аномалий используются баессовские сети — графические модели представления в собственной структуре зависимостей между объектами и распределений вероятностей. По множеству тренировочных данных оцениваются корреляции между состояниями и строится сеть с соединенными, зависимыми между собою, узлами, а также вероятности переходов между состояниями, которые заполняются, и связанные с узлами таблицы распределения вероятностей по данным состояниям узлов-предков. Из-за жесткой зависимости построенной структуры от предоставленных тренировочных данных полученная модель не является адаптивной.

Теоретико-информационный подход предлагается в [23], где приводятся два методы выбора оптимальной ширины окна для обнаружения аномалий в последовательностях программных системных вызовов. Первый — вычислением условной энтропии $H(X|Y)$ следующего символа $X \in \Sigma$ при условии предыдущего контекста $Y \in \Sigma^n$. То значение n (отдельное для каждой программы), при котором энтропия минимальная, будет шириной окна. Второй метод — использованием ансамблей вероятностных деревьев, аналогичных тем, которые используются в [19]. Каждое учитывается с определенным весом при вычислении вероятности следующего системного вызова. Если какое-то дерево правильно "угадало" следующий вызов, его вес в ансамбле увеличивается, чем достигается адаптивность.

В работе [14] описан интересный подход, который, в отличие от традиционных методик обнаружения аномалий после обучения на неаномальних данных, предназначен для обнаружения аномалий без предшествующей тренировки на примерах нормальной активности. Для этого совокупность последовательных данных представлено как таковую, которая была сгенерирована "смешанной" стохастичной моделью, т.е. с (малой) вероятностью λ элемент последовательности является аномальным и с вероятностью $(1-\lambda)$ — нормальным. Распределение A аномальных элементов при отсутствии каких-то априорных данных о нем считается равномерным. Распределение нормальных элементов M может оцениваться любым методом из арсенала машинного обучения. Следовательно, имеем генерирующее распределение-смесь $D = (1-\lambda)M + \lambda A$. Далее, считая, что соответствующие распределения известны, можно вычислить функцию правдоподобности $L(D)$ для данной последовательности элементов. Тогда, если текущий элемент x_t переместить из нормального распределения M в аномальное A и при этом правдоподобность увеличится, то элемент там и остается, а если нет — ничего не изменяется. Применение этого подхода дает возможность не зависеть от "чистоты" тренировочных данных, которые могут содержать и аномальные включения (при условии их незначительного количества). В работе дается экспериментальное сравнение этого метода (где M — марковская цепь третьего порядка) с двумя другими.

Были попытки использования для обнаружения аномалий генетических алгоритмов — т. е. алгоритмов поиска оптимума, базирующихся на аналогиях с природным отбором в природе. В качестве "популяции" берется множество "особей" — бинарных строк фиксированной длины. Количество строк во множестве также фиксировано. В процессе эволюции, по определенному критерию — "приспособленности к окружающей среде" выбираются строки, из которых генерируется следующее поколение строк. Генерация следующего поколения происходит с использованием трех основных операторов над строками: селекции (выбор наиболее приспособленного представителя), репродукции (обмен частями строк между собою) и мутации (случайные изменения битов в строках во избежание необратимой утраты информации). Так, в [24] генетические алгоритмы используются для нахождения пессимистического сценария в гибридной задаче обнаружения злоупотреблений и аномалий. С одной стороны, обнаруживаются только известные атаки, с другой — применение генетических алгоритмов и нестрогих ограничений на гипотезы может позволить регистрировать также и вариации этих атак. Пусть N_e и N_a — количество аудит-событий, регистрирующихся в системе и количество известных атак. Тогда столбцы матрицы $\hat{A}_{N_e \times N_a}$ указывают количество событий каждого типа в аудите. Векторы R, O и H — соответственно вектор длины N_a рисков, связанных с атаками, вектор длины N_e количества событий, имеющихся в текущем аудите и вектор-гипотеза с одним единичным компонентом на месте, которое соответствует определенной атаке. Тогда задачей генетического поиска есть нахождение вектора H, который максимизирует произведение (R, H) (то есть атаки, которая максимизирует потери) при условии, что $(\hat{A}H)_i < O_i, 1 \le i \le N_a$. Подход был проверен на искусственных данных со смоделированными простыми атаками.

Другие обзоры. Среди ранних обзоров стоит вспомнить описание некоторых применений методов искусственного интеллекта к обнаружению вторжений, приведенный в [25]. Сравнительный анализ не исчерпывающего количества методов обнаружения аномалий, базирующегося на системных вызовах, описан в работе [26]. Рассматриваются примеры четырех основных подходов к задаче: простое запоминание подпоследовательностей, которые попадаются в тренировочных данных, частотные методы, data-mining и HMM. Все конкретные модели были проверены на больших массивах реальных данных как с аномальными включениями, так и без них, после чего был сделан вывод, что HMM-модель является наилучшей, хотя и не самой эффективной.

В нашем обзоре мы рассматривали в основном математический аспект проблемы моделирования поведения субъектов компьютерной системы. Важным является также вопрос сравнительной характеристики имеющихся IDS и их классификации. Так, в обзоре [27] рассмотрено и подробно классифицировано по многим параметрам 20 (от первых до тех, которые появились сравнительно недавно) IDS. В обзоре [28] детально описываются исследовательские, коммерческие и доступные для широкой публики системы обнаружения вторжений. Также перечислен проблемы, которые должны решать будущие IDS, и даны рекомендации по выбору, поддержке этих систем.

Выводы

Мы рассмотрели основные модели, применяющиеся в ADS, но ни одна из них не гарантирует обнаружение всех атак, поэтому в реальных IDS мы рекомендуем пользоваться совокупностью показателей, полученных по разным моделям и по ним делать окончательный вывод о наличии или отсутствии вторжений. их характере и т. д.

Модели не должны зависеть от конкретного типа аудит-данных, тогда их можно применять к любым аудит-последовательностям. Такими данными, кроме команд, которые непосредственно выполняются пользователем, могут быть последовательности системных вызовов, характерные для данной программы, значения интервалов между нажатиями клавиш на клавиатуре, интервалы

между рабочими сессиями, последовательные значения координат положения курсора мыши на экране, последовательности значений полей заголовка сетевых пакетов всех уровней или величины, которые в них не содержатся, но от них зависят и т.п. Иными словами любая последовательность сигналов, команд, элементов, хронологических или других значений, характерная для пользователя, процесса, системы или сетевого сегмента, может быть использована в ADS.

Пользователям свойственно изменять свое поведение (из-за изменения задач, приобретения новых привычек и т. п.), поэтому модели обязательно должны быть адаптивными.

Большая часть атак является атаками против системных программ, например, приобретения прав суперпользователя путем использования уязвимостей в SUID программах. Программа, как правило, демонстрирует кардинально отличный характер системных вызовов от тех, которые наблюдались до атаки [3,4,10-12,14,23,26], поэтому надо обратить внимание на обнаружение аномалий на уровне системных вызовов, который становится все более популярным, поскольку дает возможность абстрагироваться от нерегулярного человеческого поведения. Одновременно нельзя отказываться от анализа аудит-данных, которые поступают от пользователей, поскольку только анализ на этом уровне делает возможным обнаружение вторжений, которые на уровне системных вызовов не проявляются (например, полностью законное с точки зрения подсистемы аутентификации использование украденного пароля).

Полноценная IDS должна иметь также и компоненту по обнаружению злоупотреблений, поскольку на сегодняшний день атаки с использованием эксплойтов остаются одними из основных видов атак.

1. S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In ACM Conference on Computer and Communications Security, pages 1-7, 1999.
2. D. E. Denning. An intrusion-detection model. In Proc. IEEE Symposium on Security and Privacy, pages 118-131, 1986.

3. A. Somayaji. Automated response using system-call delays. In Proc. of USENIX Security Symposium 2000, pages 185-197, 2000.
4. S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for Unix processes. In Proc. of the 1996 IEEE Symposium on Research in Security and Privacy, pages 120-128. IEEE Computer Society Press, 1996.
5. T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. ACM Transactions on Information and System Security, 2(3): 295-331, 1999.
6. M. Theus and M. Schonlau. Intrusion detection based on structural zeroes. Statistical Computing and Graphics Newsletter, 9(1): 12-17, 1998.
7. K. Tan. The application of neural networks to unix computer security. In Proc. of the IEEE International Conference on Neural Networks, volume 1, pages 476-481, Perth, Australia, 1995.
8. J. Ryan, M. Lin, and R. Miikkulainen. Intrusion detection with neural networks. Advances in Neural Information Processing Systems, pages 254-272, 1998.
9. А. М. Резник, Н. Н. Куссуль, А. М. Соколов. Нейросетевая идентификация поведения пользователей компьютерных систем // Кибернетика и вычислительная техника, 1999. – Вып. 123. – С. 70-79.
10. D. Endler. Intrusion detection: Applying machine learning to solaris audit data. In Proc. Annual Computer Security Applications Conference (ACSAC'98), pages 268-279, Los Alamitos, CA, Dec. 1998. IEEE Computer Society Press. Scottsdale, AZ.
11. Ghosh, J. Wanken, and F. Charron. Detecting anomalous and unknown intrusions against programs. In Proc. of the 1998 Annual Computer Security Applications Conference (ACSAC'98), Dec. 1998. http://rstcorp.com/~anup/ACSAC98.pdf
12. K. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior profiles for intrusion detection. In Proc. 1-st USENIX Workshop on Intrusion Detection and Network Monitoring, pages 51-62, Santa Clara, California, Apr. 1999.
13. D. Davison and H. Hirsh. Predicting sequences of user actions. In Predicting the Future: AI

Approaches to Time-Series Problems, pages 5-12, Madison, WI, July 1998. AAAI Press. Proc. of AAAI-98/ICML-98 Workshop.

14. E. Eskin. Anomaly detection over noisy data using learned probability distributions. In Proc. 17th International Conf. on Machine Learning, pages 255-262. Morgan Kaufmann, San Francisco, CA, 2000.
15. N. Ye. A markov chain model of temporal behavior for anomaly detection. In Proc. of the 2000 IEEE Systems, Man, and Cybernetics, Information Assurance and Security Workshop, 2000. http://www.itoc.usma.edu/marin/Wshop/Papers2000/WA1_1.pdf
16. T. Lane. Hidden markov models for human/computer interface modeling. In IJCAI-99 Workshop on Learning About Users, pages 35-44, 1999.
17. C. C. Michael and A. Ghosh. Two state-based approaches to program-based anomaly detection. ACM Transactions on Information and System Security, 5(2), 2002. http://www.acsac.org/2000/papers/96.pdf
18. D. Wagner and R. Dean. Intrusion detection via static analysis. In Proc. of the 2001 IEEE Symposium on Security and Privacy, pages 156-169, Los Alamitos, CA, May 14-16 2001.
19. Н. Н. Куссуль, А. М. Соколов. Адаптивное обнаружение аномалий в поведении пользователей компьютерных систем с помощью марковских цепей изменяющегося порядка // Кибернетика и вычислительная техника (подано в печать).
20. D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. Machine Learning, 25(2-3): 117-149, 1996.
21. W. Lee and S. Stolfo. Data mining approaches for intrusion detection. In Proc. of the 7th USENIX Security Symposium, pages 79-94, San Antonio, TX, 1998.
22. N. Ye, M. Xu, and S. M. Emran. Probabilistic networks with undirected links for anomaly detection. In Proc. of the 2000 IEEE Workshop on Information Assurance and Security, West Point, NY, June 2000. http://www.itoc.usma.edu/marin/Wshop/Abstracts/WA1_2.pdf
23. E. Eskin, W. Lee, and S. J. Stolfo. Modeling system calls for intrusion detection with dynamic

window sizes. In Proc. of DARPA Information Survivability Conference and Exposition (DISCEX II), June 2001. http://www.cs.columbia.edu/ids/publications/smt-syscall-discex01.pdf

24. L. Me. GASSATA, a genetic algorithm as an alternative tool for security audit trails analysis. In First international workshop on the Recent Advances in Intrusion Detection, Louvain-la-Neuve, Belgium, Sept. 1998. http://www.raid-symposium.org/raid98/Prog_RAID98/Full_Papers/gassata.pdf
25. J. Frank. Artificial intelligence and intrusion detection: Current and future directions. In Proc. of the 17th National Computer Security Conference, pages 22-33, Baltimore, MD, 1994.
26. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In Proc. of the 1999 IEEE Symposium on Security and Privacy (SSP '99), pages 133-145, May 1999. IEEE.
27. J. Allen et al. State of the practice of intrusion detection technologies. TR CMU/SEI-99-TR-028, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Jan. 2000.
28. S. Axelsson. Research in intrusion-detection systems: A survey. TR 98-17 SE-412 96, Department of Computer Engineering, Chalmers University of Technology, Geteborg, Sweden, Dec. 1998.