УДК 004.49

Н.Г. Булахов, В.Т. Калайда

Методы обнаружения и обезвреживания саморазмножающихся вирусов

Представлен обзор современных методов выявления атак вредоносных программ.
Произведен анализ преимуществ и недостатков данных методов.

Вирусные атаки являются одной из первостепенных угроз информационной безопасности [1]. Такие действия наносят финансовый ущерб, а также позволяют реализовать многие другие опасные угрозы. Для борьбы с такими вредителями разработаны методы обнаружения и обезвреживания, которые рассмотрены в данной статье. Выделяется три основных класса методов: это сигнатурный, статистический и эвристический.

Сигнатурный метод анализа

Сигнатурные методы описывают каждую атаку особой моделью или сигнатурой [2], в качестве которой могут применяться строка символов, семантическое выражение на специальном языке, формальная математическая модель и т.д. Суть сигнатурного метода в следующем: в исходных данных, собранных сетевыми и хостовыми датчиками, с использованием специализированной базы данных сигнатур атак выполняется процедура поиска сигнатуры атаки. Преимущество данных методов – высокая точность определения факта атаки, а очевидный недостаток – невозможность обнаружения атак, сигнатуры которых пока не определены. Типичными представителями, реализующими данную идею, являются системы обнаружения сетевых атак [1], которые работают с базой данных сигнатур удаленных атак. Необходимо отметить, что непосредственное сравнение сигнатуры вторжения с регистрируемой активностью малоэффективно в связи с тем, что регистрируемые данные, относящиеся к атаке, часто бывают зашумлены вследствие вариаций действий нарушителя во время атаки или мутаций сценария. Таким образом, сигнатурный метод вторжения предполагает использование методов искусственного интеллекта.

Рассмотрим различные методы, использующие сигнатуры вторжений.

Продукционные/экспертные системы обнаружении вторжения кодируют данные об атаках и правила импликации «если – то», а также подтверждают их, обращаясь к контрольным записям событий. Правила кодируются для определения условий, необходимых для атаки в части «если». Когда эти условия соблюдены, выполняется часть «то» правил.

Продукционные системы обнаружения вторжения предназначены для:

1) логического вывода факта вторжения на основании данных аудита. Существуют следующие проблемы, усложняющие решение данной задачи: трудность учета последовательности событий, что заставляет ввести дополнительные проверки данных, определяющие их последовательность; необходимость хорошего администратора системы, который мог бы настроить ее в соответствии со своими знаниями о безопасности системы, что может сделать базу знаний плохо переносимой; могут быть обнаружены только известные уязвимости;

2. объединения возможных атак для получения общей картины вторжения.

Преимуществом продукционных систем обнаружения вторжения является отделение управляющего решения от его формулировки. Кроме описанных выше, недостатками продукционных систем обнаружения вторжения являются управление огромным количеством данных, соответствующих вторжению, а также высокая квалификация, необходимая для поддержки базы знаний системы обнаружения вторжений.

Обнаружение вторжений, основанное на модели. Этот метод построения систем обнаружения вторжений является одним из вариантов, объединяющим модели вторжения и доказательств, поддерживающих вывод о вторжении. В системе обнаружения вторжений поддерживается база данных сценариев атак. Каждый из сценариев составляет последовательность поведений, описывающих атаку. Система обнаружения вторжений пытается выявлять эти сценарии по записям в контрольном журнале и таким образом доказывать или опровергать их присутствие. Такой процесс получил название «планировщик». Планировщик пытается определить последовательность поведений, основанную на текущих активных моделях вторжения, которые могут быть проверены в журнале по шаблонам. Метод основан на накоплении контрольной информации, и поэтому возврат к предыдущему результату проверки сценария не вызывает осложнений. Планировщик переводит описание поведения в активность в вычислительной системе и сравнивает его с данными системно-зависимого контрольного журнала.

Вероятность P (Activity/Behavior)/P (Activity/ШBehavior) при вторжении должна быть высока. Преимущества данного метода: математическая база поддержки вывода в условиях неопределенности; планировщик обеспечивает независимое представление данных аудита. Недостатки данного метода: дополнительная нагрузка на эксперта выражается в правильном присвоении имеющих смысл числовых значений элементам графа, представляющего модель; поведение описывается последовательным множеством событий, что затрудняет описание сложных атак.

Необходимо отметить, что обнаружение вторжений, основанное на сравнении с моделью, не заменяет, а дополняет метод определения по статистическим критериям.

Анализ перехода системы из состояния в состояние представлен в системах STAT и USTAT под ОС UNIX. В данных системах обнаружения вторжений атаки представляются как последовательность переходов контролируемой системы из состояния в состояние. Состояние шаблона атаки соответствует состоянию системы и связано с утверждениями, которые должны быть удовлетворены для последующего перехода из состояния в состояние. Возможные состояния связаны дугами, представляющими события, необходимые для перехода из состояния в состояние. Типы допустимых событий встроены в модель. Данная модель может определить только атаку состоящую из последовательных событий, не позволяя выразить атаки с более сложной их структурой. Более того, не существует общего целевого механизма в случае частичного распознавания атак.

Изменение состояний и сети Петри. Метод, использующий формализм сетей Петри для обнаружения вторжений, является развитием метода анализа изменений состояний. Данный подход был разработан в университете Purdue. Хотя данный метод позволяет получить более точные результаты в области обнаружения вторжений, он требует значительных вычислений и не может использоваться в системах обнаружения вторжений реального времени, что, впрочем, допустимо для исследовательских целей.

Сигнатурные методы очень надежны [3], однако новые вирусы с их помощью можно обнаружить лишь после соответствующего обновления системы, которое в лучшем случае оказывается возможным лишь через несколько часов, а нередко и спустя целый день – столько времени проходит до того момента, когда последний производитель выпустит новую сигнатуру и предоставит ее своим клиентам по Internet. В течение этого срока пользователь беззащитен перед новыми вредителями.

Временной лаг между появлением нового вируса и выходом обновления сигнатур делает компьютеры уязвимыми к атакам из Internet. Причем производители решений для обеспечения безопасности всегда хотя бы на шаг, но отстают от вирусописателей. К тому же перед публикацией любого вируса его создатель проверяет, как реагируют на него наиболее популярные антивирусные решения. В результате антивирусные продукты на базе сигнатур, с одной стороны, надежно защищают от известных вредителей, а с другой – совершенно не замечают нового вредоносного программного обеспечения. Они способны ограничить распространение вирусной эпидемии, но воспрепятствовать заражению множества компьютеров до выпуска обновления не в состоянии.

Сегодня киберпреступники способны доставить свои вредоносные программы на десятки тысяч компьютеров за очень короткое время. Кроме того, они получают дополнительное время для маневра благодаря модульной компоновке вирусов и программированию сразу нескольких различных вариантов своих кодов. Это вынуждает производителей систем безопасности создавать отдельные сигнатуры для каждого варианта, а вирусописатели между тем успевают без спешки установить разработанные ими инструменты системного уровня на зараженные компьютеры или открыть потайные ходы, чтобы с их помощью похитить важные данные или пароли.

Статистический метод анализа

Рассмотрим различные модели, которые могут применяться при статистическом анализе безопасности поведения программ и в системах обнаружения нарушителя [1].

Операционная модель основывается на том, что каждое новое наблюдение переменной должно укладываться в некоторых границах. Если этого не происходит, то мы имеем дело с отклонением. Допустимые границы определяются на основании анализа предыдущих значений переменной. Данная модель может использоваться, если некоторое значение метрики можно аргументированно связать с попыткой нарушения безопасности (например, количество попыток ввода пароля более 10).

Модель среднего значения и среднеквадратичного отклонения базируется на том, что все, что мы знаем о предыдущих наблюдениях некоторой величины x, это ее среднее значение $\mu = \sum x_i / n$ и среднеквадратичное отклонение $\sigma = \mathrm{sqrt}((x_1^2 + \ldots + x_n^2)/(n-1) - \mu^2$. Тогда новое наблюдение является анормальным, если оно не укладывается в границах доверительного интервала $\mu + d^*\sigma$. Модель применима для измерения счетчиков событий, временных интервалов и используемых ресурсов. Преимуществом модели, по сравнению с операционной, является независимость оценки анормальности поведения от априорных знаний. Кроме того, необходимо отметить, что анормальность поведения зависит от значения доверительного интервала и, как следствие, понятие анормальности для пользователей системы может отличаться.

Многовариационная модель аналогична модели среднего значения и среднеквадратичного отклонения, но учитывает корреляцию между двумя или большим количеством метрик (использование ЦПУ и количество операций ввода-вывода, количество выполненных процедур входа в систему и время сессии).

Модель Марковского процесса применима только к счетчикам событий, рассматривая каждый тип событий как переменную состояния и используя матрицу переходов для характеристики частот переходов между состояниями. Наблюдение является аномальным, если вероятность перехода, определенная предыдущим состоянием и матрицей перехода, очень мала. Модель применима в том случае, если рассматривается множество команд, последовательность которых важна.

Модель временных серий. Использует временные периоды вместе со счетчиками событий и измерениями ресурса, учитывая как значения наблюдений $x_1,\ldots,x_n$, так и временные интервалы между ними. Новое наблюдение является анормальным, если вероятность его появления с учетом времени низка. Преимуществом данной модели является учет временного сдвига между событиями, а недостатком – накладные расходы по вычислению по сравнению с моделью среднего значения и среднеквадратичного отклонения.

Информационный анализ. Метод состоит в выделении особенностей больших наборов данных, т.е. необходимо определить наиболее компактное описание нормального поведения на основе перебора всех шаблонов, отражающих нормальное поведение. На основе определенных особенностей необходимо сделать обобщение для определения нормальных шаблонов, пропущенных в тренировочных данных.

Для реализации данного метода использовалась система RIPPER (Repeated Incremental Pruning to Produce Error Reduction) – система, построенная на основании обучающих правил и используемая для решения задач классификации. Тренировка состоит в определении образцов, состоящих из множества атрибутов, описывающих классифицируемый объект, и целевого класса, к которому принадлежит объект. На основании многих примеров RIPPER выделяет правила в следующей форме:

ClassA:- attrib1 = x, attrib5 = y,

ClassB:- attrib2 = z,

ClassC:- true (по умолчанию).

Для задач обнаружения нарушителя данный метод применим только в том случае, если известно полное множество примеров анормальных классов, на основании которых тренируется система. Lee адаптировал систему RIPPER для обнаружения аномалий с использованием правил, предсказывающих системные вызовы, для коротких последовательностей трасс программы.

Для каждой программы был использован список уникальных последовательностей длины 10, на основании которых были созданы образцы для тренировки RIPPER. Каждая последовательность была преобразована в пример RIPPER преобразованием всех системных вызовов, за исключением последнего, в атрибуты. На основании последнего вызова определяется класс, соответствующий последовательности. Такие пары атрибут/цель создаются для тестовых трасс. При этом используются все последовательности системных вызовов, а не только примеры уникальных трасс. RIPPER создает гипотезы на основе тренировочных образцов – список правил, описывающих нормальное поведение системы.

Для каждого правила RIPPER определяет оценку нарушения на основе количества ситуаций, в которых правило было применено корректно в тренировочных данных. Например, для правила, чьи условия были удовлетворены M раз во время тренировки, и предсказание было сделано корректно T раз, оценку нарушения можно определить как $100 * M/T$. Среднее значение оценки нарушений может быть использовано для вычисления ранга трассы (это не применимо для online обнаружения вторжений). Следовательно, целесообразно использовать отклонение от вероятной трассы (использовалось значение 80%) как «пропуск» с использованием механизма, описанного для других методов.

Машина конечных состояний. Подход состоит в разработке машины конечных состояний для распознавания «языка» трассы программы (кода, находящегося на компьютере либо передаваемого через сеть). Для решения данной задачи существует много техник, основанных на использовании как детерминистских, так и вероятностных автоматов. При этом обычно определяется частота, с которой появляются отдельные символы (системные вызовы) в зависимости от некоторого числа предыдущих символов. Состояния соответствуют истории системы, а переходы показывают вероятность появления следующего символа. При этом алгоритмы базируются на стационарности данных.

Одной из основных моделей построения таких систем является скрытая модель Маркова (СММ). Состояния СММ представляют некоторые ненаблюдаемые условия моделируемой системы. В каждом состоянии существует вероятность некоторого наблюдаемого вывода системы и отдельная вероятность, определяющая следующее состояние. Наличие в модели отдельно распределения вероятности вывода для каждого состояния и возможности изменений состояний во времени позволяет представить в модели нестационарные последовательности. Модель является очень мощной, но требует большого объема вычислений.

Для применения СММ необходимо, в первую очередь, определить размер модели (СММ работает с фиксированным количеством состояний). На основании исследования анализируемого ПО было обнаружено, что большинство программ использует 40 уникальных системных вызовов (т.е. используется СММ с 40 состояниями). Состояния являются полностью связанными (переходы возможны из каждого состояния в каждое). Для каждого состояния необходимо хранить вероятность перехода в каждое состояние и вероятность выполнения каждого системного вызова. Таким образом, для программы, использующей S системных вызовов, требуется модель с S состояниями, хранящая $2S^2$ значений. Обычно все значения инициализируются случайным образом и тренируются с использованием алгоритма Baum-Welch (однако при инициализации желательно использовать априорные знания).

Во время тренировки вероятности итеративно изменяются для увеличения вероятности того, что автомат образует трассы для тренировочного множества (требуется несколько проходов через тренировочные данные). Для того, чтобы избежать излишних тренировок, поддерживается второе множест-

во нормальных трасс, которое периодически оценивается. Если вторая вероятность больше не улучшается, то тренировку следует прекратить.

Вычисления для каждой трассы для каждого прохода при тренировке занимают $O(TS^2)$, где T – длина трассы. Промежуточные переменные при этом занимают $T*(2*S+1)$ значений с плавающей точкой. Тестирование является более эффективным. Стандартным способом тестирования является проверка вероятности неизвестной трассы.

Более целесообразным (не зависящим от длины трассы) является следующий метод, использующий граф, лежащий в основе СММ как недетерминированный конечный автомат. При этом «считывается» один системный вызов за момент времени с определением требуемых переходов состояний и выводов для того, чтобы СММ обработала данный вызов. Если СММ хорошо описывает программу, то для нормальных трасс будут требоваться только вероятные переходы и выводы, а при вторжении – необычные переходы и выводимые символы.

В некоторый момент времени t существует список возможных состояний. Выбор самого вероятного состояния для любого системного вызова не совместим с лучшим путем для СММ для последовательности системных вызовов, так что необходимо хранить все возможные пути. Управляемым параметром является уровень (в экспериментах варьируется от 0,01 до 0,0000001), свидетельствующий о нормальном переходе и выходной вероятности. Если в трассе встречается системный вызов, на основании которого образуются системные переходы ниже определенного уровня, то он рассматривается как «пропуск». Необходимо отметить, что СММ делает оценку для отдельных системных вызовов, а не для их последовательностей. Для каждого системного вызова оценка осуществляется за время $O(S^2)$.

Эвристические методы

Эвристический анализатор (эвристик) – это программа, которая анализирует код проверяемого объекта и по косвенным признакам определяет, является ли объект вредоносным [4]. Причем в отличие от сигнатурного метода эвристик может детектировать как известные, так и неизвестные вирусы (т.е. те, которые были созданы после эвристика).

Работа анализатора, как правило, начинается с поиска в коде подозрительных признаков (команд), характерных для вредоносных программ. Этот метод называется статичным анализом. Например, многие вредоносные коды ищут исполняемые программы, открывают найденные файлы и изменяют их. Эвристик просматривает код приложения и, встретив подозрительную команду, увеличивает некий «счетчик подозрительности» для данного приложения. Если после просмотра всего кода значение счетчика превышает заданное пороговое значение, то объект признается подозрительным. Разумеется это применимо не только к программам хранящимся на компьютере [3] , но и к другим файлам передающимся по сети, хранящимся на почтовых серверах, и т.д. К примеру, как потенциально опасные маркируются вложения с отсутствующими расширениями файлов. Команды, без разрешения открывающие адресную книгу Outlook Express, создающие ключи реестра или активирующие сетевые порты, тоже идентифицируются как потенциальные вредители.

Эвристические методы оказываются чрезвычайно успешными в случае очень коротких программных частей в загрузочном секторе: если, к примеру, программа производит запись в сектор 1, дорожку 0, сторону 0, то это приводит к изменению раздела накопителя. Но кроме вспомогательной программы FDISK эта команда больше нигде не используется, и потому в случае ее неожиданного появления речь (с большой вероятностью) идет о загрузочном вирусе.

Эвристические алгоритмы способны обнаружить новых или мутировших вредителей, для которых пока нет вирусных определений. Поскольку этот метод поиска базируется на эмпирических предположениях, полностью исключить ложные срабатывания (false positive) нельзя.

Преимуществом этого метода являются простота реализации и высокая скорость работы, однако уровень обнаружения новых вредоносных кодов остается довольно низким, а вероятность ложных срабатываний высокой [4].

Поэтому в современных антивирусах статический анализ используются в сочетании с динамическим. Идея такого комбинированного подхода состоит в том, чтобы до того как приложение будет запущено на компьютере пользователя, эмулировать его запуск в безопасном виртуальном окружении, которое называется также буфером эмуляции, или «песочницей». В маркетинговых материалах поставщики используют еще один термин – «эмуляция виртуального компьютера в компьютере».

Технология «песочниц» такова: подозрительный файл помещается в изолированную от остальной системы «песочницу», где проверяется его поведение [3]. При этом каждый исполняемый файл запускается в виртуальной среде, которую стандартный программный интерфейс приложений Windows (Application Programming Interface, API) организует в так называемой «тюрьме» (jail). Потенциально вредоносное приложение не может воздействовать на операционную систему и нанести ей вред. Если же оно отдает, к примеру, команду об изменении системных настроек, то классифицируется как вредитель и может быть либо удалено, либо переведено в карантин. Вне всяких сомнений, методы на основе «песочницы» являются надежными и достаточно точно распознают и устраняют неизвестное до сих пор вредоносное программное обеспечение. Тем не менее для практического использования они пригодны лишь условно: поскольку новые вирусы многочисленны, а их структура довольно сложна,

эмуляция же на шлюзе длится долго и отнимает слишком много ресурсов. Кроме того, «песочницы» приносят мало пользы, когда вирусы выполняют свою программу с временной задержкой и на момент проверки не распознаются как вредоносные коды.

По новому пути идет *технология блокировки поведения*, где комбинируются оба упомянутых метода. При ее применении благодаря иерархической обработке трафика данных сводится к минимуму высокая потребность «песочницы» в ресурсах и уменьшается вероятность ложных срабатываний эвристических методов: в то время как в «песочницах» подробно изучается каждый входящий файл, блокираторы поведения подвергают анализу поведение лишь тех файлов, которые эвристическими методами были отмечены как подозрительные по причине своего происхождения или присутствия известных составных частей кода. Тем самым значительно сокращается количество файлов, что, в свою очередь, положительно отражается на системной нагрузке. Одновременно снижается риск ложных срабатываний, поскольку подозрительные файлы исследуются с точки зрения их реального поведения. Для полноты картины рассмотренные методы представлены в таблице, где проведена их классификация по критичным к обнаружению атак параметрам.

Классификация методов по критичным к обнаружению атак параметрам

Класс	Метод	Ресурсоемкость	Время выявления	Эффективность на ранней стадии	Эффективность на поздней стадии	Ложные срабатывания	Универсальность	Наличие обновляемых баз	Простота критериев оценки	Самостоятельность метода	Необходимость обучения системы
Сигнатурный метод анализа	Продукционные / экспертные системы обнаружении вторжения	+/–	+	–	+	+	–	–	–	+	+
	Обнаружение вторжений, основанное на модели	+/–	+	–	+	+	–	–	–	–	+
	Анализ перехода системы из состояния в состояние	+	+	–	+	+	–	–	–	+	+
	Изменение состояний и сети Петри	–	–	–	+	+	–	–	+	+	+
Статистический метод анализа	Статистический анализ последовательности системных вызовов	+	+	+	–	–	+	+	–	+	–
	Машина конечных состояний	–	–	+	–	–	+	+	–	+	–
Эвристический метод	Анализ поведения системы	+	+	+/–	+	+/–	–	+	–	+	+

Работа выполнена при поддержке РФФИ, проект № 06–08–00751.

Литература

1. Корт С.С. Методы выявления нарушений безопасности – http://www.kiev-secu-rity.org.ua/box/12/113.shtml

2. Сердюк В. Вы атакованы – защищайтесь! // Байт Россия № 9/2003 – http://www.w3c.org/TR/1999/REC-html401- 19991224/loose.dtd

3. Матиас Р. Анализ поведения и эвристические методы выявления вирусов – http://www.osp.ru/lan/2006/10/3474604/

4. Гудилин О. Проактивность как средство борьбы с вирусами – http://www.viruslist.com/ru/ analysis?pubid= 189544544

Булахов Николай Георгиевич
Томский государственный университет
Аспирант кафедры квантовой электроники и фотоники радиофизического факультета
Тел.: (3822) 41 38 25.
Эл. адрес: nboolahov@yandex.ru.

Калайда Владимир Тимофеевич
Томский государственный университет систем управления и радиоэлектроники
Д.т.н., профессор кафедры АСУ
Тел.: (3822) 49 22 42.
Эл. адрес: kvt@iao.ru.

N.G. Bulakhov, V.T. Kalaida

Self-propagating viruses detection and neutralization methods

Paper introduce statistical model of digital information network applicable for detection of harmful network software propagation.