Анализ защищенности распределенных информационных систем

В. Г. Гукасян

В данной работе ставится задача анализа безопасности информационной сети. Применяется теоретико-автоматный подход для моделирования поведения злоумышленника, вводится отношение частичного порядка на множестве состояний — векторов, значение координат которых отражает уровень прав доступа к соответствующим службам на анализируемых узлах. Рассматривается понятие трассы атаки и изучаются соответствующие количественные характеристики. Задача обнаружения уязвимостей сети переформулируется в терминах нахождения состояний, достижимых из некоторого начального состояния, и описания возможных путей до них. На основе построенной математической модели предложены алгоритмы, решающие поставленные задачи, и их возможные модификации. Также рассмотрены подходы к уменьшению построенного автомата.

Ключевые слова: компьютерная безопасность, оценка защищенности, конечные автоматы, трассы атак.

1. Введение

В настоящее время компьютерные сети имеют огромные масштабы, это приводит и к увеличению количества уязвимостей в них, соответственно усложняется процесс анализа защищенности систем. Для решения задачи исследования безопасности можно эффективно использовать графы, так как они позволяют строить наглядные модели информационных систем. Помимо удобства нельзя не отметить продуктивность использования графовых моделей, позволяющих с достаточной точностью описать реальные объекты [1, 2, 3, 4, 5, 6, 7, 8, 9].

В данной работе осуществляется попытка решения проблемы анализа безопасности распределенной информационной сети. Предлагается математическая модель, отражающая уязвимости системы и воз-

можные действия злоумышленника. На этой модели построены алгоритмы решающие следующие задачи:

1) формирование наиболее полного списка небезопасных узлов сети;
2) возможность компрометации выделенного ресурса;
3) описание всевозможных атакующих действий на выделенный ресурс;
4) описание всевозможных атакующих действий из начального положения злоумышленника;
5) описание всевозможных атакующих действий, использующих выделенную уязвимость.

Структурно работа состоит из трех частей: в первой части строится математическая модель, в качестве нее используется недетерминированный автомат [10], затем описываются алгоритмы, решающие поставленные задачи, и их возможные модификации, в заключительной части предложены подходы по уменьшению математической модели, облегчающие её реализацию.

2. Математическая модель информационных систем

2.1. Построение модели

Предположим, дано m узлов информационной сети и логические связи между ними, определяющиеся топологией сети. В каждом из узлов анализируем n_i служб, где $i = 1, \ldots, m$. Считаем, что для служб существует некоторое количество уровней доступа, причем на них есть частичный порядок: одни права больше других. Так получим частично упорядоченное множество возможных прав доступа к службам. Таким образом, положение нарушителя в системе задается вектором размерности $\sum n_i, i = 1, \ldots, m$, где в каждой координате записан уровень прав доступа к соответствующей службе, на соответствующем узле (обозначим $\sum n_i$ за N). Иначе говоря, первые n_1 координат показывают доступ к службам на первом узле, следующие n_2 координаты — на втором и т. д. Обозначим множество возможных векторов через AV, а сами вектора — $AccessVect_i$, где $i \in \{1 \ldots |AV|\}$ Полагаем, что значение координаты увеличивается с

увеличением (расширением) прав использования мошенником службы, соответствующей данной координате. Таким образом, порядок на множестве прав доступа к каждой анализируемой службе индуцирует частичный порядок на множестве AV: $AccessVect_k \geqslant AccessVect_l$ тогда и только тогда, когда каждая координата $AccessVect_k$ не меньше соответствующей координаты $AccessVect_l$. Формально:

$$AccessVect_k \geqslant AccessVect_l \rightleftharpoons$$
$$\rightleftharpoons \forall i, 1 \leqslant i \leqslant N : AccessVect_k[i] \geqslant AccessVect_l[i].$$

Пример. Возможные права доступа. В системе Linux права доступа к файлу задаются строкой из 9 символов, где указываются права владельца, группы-владельца и остальных пользователей. Согласно сказанному выше, на множествах возможных прав доступа каждой категории можно ввести порядок: `rwx` $\geqslant$ `-wx`, `r-x` $\geqslant$ `--x`, `-w-` $\geqslant$ `---`. А, в свою очередь, `rw-` и `--x`, `r-x` и `-wx`, `r--` и `--x` будут не сравнимыми элементами.

Замечание. Возможность установить полный порядок. В данном примере возможно установить полный порядок на координатах, отвечающих за отдельные службы, посредством увеличения размерности векторов из множества AV. Рассматривая параметры `rwx` по отдельности, закодируем присутствие буквы единицей, а отсутствие соответствующих прав — нулем (теперь за доступ к файлу определенной группой пользователей будет отвечать не одна координата вектора-положения, а три). Так `---` будет иметь код 000, а `rwx` — 111. Как видно, по каждой из этих 3 координат имеет место точный порядок (возможные значения только 0 или 1), но порядок на множестве AV частичный: получившиеся тройки из 0 и 1 сравниваются, как 3-наборы.

Перейдем непосредственно к построению недетерминированного автомата V. Состояния автомата — это вектора-положения: если Q — множество состояний автомата, то $|Q| = |AV|$ и каждому $q_i \in Q$ соответствует ровно один $v_j \in AV$, причем различным состояниям соответствуют различные вектора-состояния (то есть определена и зафиксирована биекция между Q и AV). Также далее будем полагать, что ассоциированные друг с другом элементы множеств Q и AV имеют одинаковые номера: q_k состоянию автомата соответствует вектор-состояние $AccessVect_k$. Таким образом, корректно равенство: $Q = AV$.

Теперь опишем переходы автомата. Под *атомарной атакой* будем понимать совокупность действий, направленных на использование нарушителем отдельной уязвимости. Например, переполнение буфера службы SSH, позволяющее удаленно получить права администратора системы [11]. Каждый переход в автомате задаётся одной из рассматриваемых атомарных атак. Атомарная атака представляет из себя эксплуатирование уязвимости на одной машине сети и в результате этого расширение/изменение прав доступа на другой (возможно, той же). Для корректного построения автомата необходима обширная база уязвимостей и эффективное обнаружение их в системе. Это можно сделать используя сканеры безопасности, например, Nessus [12]. Если состоянии системы (прав доступа злоумышленника) задавалось вектором $AccessVect_i$, а в случае успеха атомарной атаки A вектор-положение изменится на $AccessVect_j$, то будем говорить, что атомарная атака A задает переход из состояния q_i в состояние q_j. Перебирая таким образом сгенерированные атомарные атаки, расставим переходы в автомате (все переходы считаем по пустому слову, Λ-переходами).

Пример. Переходы автомата. Чтобы получить некоторое представление о возможных переходах в полученном автомате, можно внимательнее изучить параметры известных уязвимостей. Например, по версиям многих организаций, таких как OWASP [13], Mitre [14], SANS [15], одними из наиболее используемых угроз безопасности являются: инъекции (SQL-инъекции [18], инъекции при выполнении команд ОС [19]), XSS (Cross Site Scripting [20]), отсутствие аутентификации при выполнении критических действий [21, 16, 17].

Замечание. Структура переходов. Стоит отметить, что в результате подобного построения между двумя различными состояниями могут возникать несколько переходов. Иными словами, если рассматривать диаграмму Мура автомата как направленный граф, то в нем могут существовать кратные ребра.

Замечание. Входной и выходной алфавиты. Построенный автомат является автоматом без входа. Единственное, что нужно указывать в качестве входных данных — это начальное состояние. Также отсутствует и выходной алфавит: на данном этапе первостепенную роль играет наличие либо отсутствие перехода между состояниями, поэтому можно говорить об автомате, как о графе.

Пример. Моделирование реальных задач. Построенная модель дает возможность исследовать информационную сеть на такие свойства, как:

- Конфиденциальность данных. Допустим необходимо исследовать возможность нарушения конфиденциальности информации, содержащейся в файле ConfFile.data. Для этого достаточно в множество координат векторов-состояний добавить координату, отвечающую за права чтения этого файла. Координата будет иметь 2 возможных состояния: 1 — если пользователь имеет доступ к содержимому файла, 0 — иначе. Если после некоторой последовательности атакующих действий правам доступа злоумышленника будет соответствовать вектор-состояние, в котором указанная выше координата равна 1, то конфиденциальность данных может быть нарушена.
- Целостность данных. В рамках описанной модели можно действовать аналогично предыдущему примеру, с той лишь разницей, что координата будет отвечать за возможность изменения интересующего файла. В остальном описание такое же.
- Защита важных процессов от несанкционированного отключения. Зачастую в информационных сетях работает немало критически важных служб, например, межсетевые экраны или антивирусы — граф атак позволяет проанализировать, может ли злоумышленник отключить их. Для этого достаточно в множество координат векторов-состояний добавить координату, отвечающую за возможность отключать процесс (если процесс запущен на всех машинах сети, и необходимо исследовать все машины, то в векторах-состояниях будет m координат показывающих доступ к интересующей службе — по одной на каждую машину).
- Возможность подключения внешних электронных носителей. Эта компонента необходима в учреждениях, работающих с информацией, представляющей государственную тайну. Может иметь более слабое ограничение: невозможность записи информации на внешние носители.
- Возможность самостоятельной установки программного обеспечения. Если злоумышленник получит права установки ПО и инсталирует вредоносный код, это может привести к самым разнообразным негативным последствиям от доступа к конфиденциальной информации до остановки работы сети в целом.

Помимо возможностей, описанных выше, нельзя не отметить еще одно важное свойство построенной модели, а именно

- Возможность задавать начальное положение. Исходное положение моделируемого нарушителя задается его исходными правами доступа. В терминах модели это означает различное задание начального состояние построенного автомата V. Изменяя начальное состояние, можно варьировать уровень доступа нарушителя и машину, с которой начинается атака. Таким образом, модель дает возможность исследовать защищенность сети от различных классов нарушителей.

Построенную модель — недетерминированный автомат, представляющий всевозможные последовательности действий нарушителя для достижения им поставленных целей — будем называть *графом атак*. Такие последовательности действий будем назвать *трассами атак*. Выше было определено понятие атомарной атаки, тогда трасса атаки — суть последовательность атомарных атак.

Достоинства и недостатки модели.

Начнем с недостатков:

- Малая гибкость: потенциально, уязвимостей бесконечно много. При добавлении новой уязвимости в рассматриваемые, потребуется для каждого узла, использующего небезопасную службу, сгенерировать новый список атомарных атак и добавить в автомат соответствующие переходы, что громоздко.
- Достаточно большое число состояний и переходов. Современные информационные сети довольно обширны, а политики безопасности разнообразны, что влечет необходимость рассмотрения автомата с объемным множеством состояний .
- Использование недерменированного автомата. При переходе к детерменированному автомату возникнет проблема экспоненциального роста количества состояний (так как в модели используются только Λ-переходы).

Плюсами же модели являются:

- Подробное описание возможных атак.
- Возможность моделировать различные положения и уровень начального доступа злоумышленника.

- Широкая применимость. Модель позволяет выбирать исследуемые ресурсы, что дает возможность выбирать классы атак, для анализа защищенности от которых она применяется.

2.2. Количественные характеристики трасс атак

Для атомарных атак полезно рассматривать следующие параметры:

- Опасность в случае реализации.
  Возможно несколько подходов: поделить все угрозы по уровню опасности для компонентов сети на 3 типа (по возрастанию): Low, Medium и High — либо использовать более подробную оценку — числовые значения, присвоенные угрозам одной из известных баз уязвимостей, например, CVSS [22]. Под опасностью можно подразумевать не только уровень полученных привилегий, но и, к примеру, время или стоимость, необходимые на исправление последствий. Также можно ввести более подробную классификацию в зависимости от конкретной информационной сети и поставленных задач, например, для финансовой организации могут быть следующие уровни:
  1) раскрытие информации принесет ничтожный моральный и финансовый ущерб фирме;
  2) ущерб от атаки есть, но он незначителен, основные финансовые операции и положение фирмы на рынке не затронуты;
  3) финансовые операции не ведутся в течение некоторого времени, за это время фирма терпит убытки, но ее положение на рынке и количество клиентов изменяются минимально;
  4) значительные потери на рынке и в прибыли; от фирмы уходит ощутимая часть клиентов;
  5) потери очень значительны, фирма на период до года теряет положение на рынке; для восстановления положения требуются крупные финансовые займы;
  6) фирма прекращает существование.
- Сложность реализации.
  Присваиваем каждой уязвимости (и соответствующей атомарной атаке) в зависимости от сложности её использования уровень Low, Medium или High.
- Вероятность реализации.
  Здесь также возможно 2 подхода:

а) Рассматривать вероятность удачной реализации атомарной атаки, то есть с какой вероятностью *можно* эксплуатировать уязвимость. Это чрезвычайно полезно при оценке защищенности ресурса, но получение точных значений данной характеристики сложно и зависит от множества факторов (ПО узла, используемых аппаратных и программных средств защиты, возможностей злоумышленника, используемых алгоритмов и т. д.)

б) Каждому автоматному переходу (и соответствующей атомарной атаке) приписать вероятность того, что реализовываться будет именно он, то есть с какой вероятностью злоумышленник *будет* эксплуатировать именно эту уязвимость. Тогда сумма вероятностей на переходах из одного состояния будет равна 1 — можно перейти уже к вероятностному автомату. Но так как цели и модель поведения нарушителя априори не известна, сложно подобрать параметры, соответствующие действительности.

- Стоимость реализации.
  Эта характеристика похожа на первую, но здесь можно рассматривать уже более конкретные параметры и их числовые значения: в частности, время, необходимое на использование уязвимости, или, скажем, денежную составляющую, например, цену аппаратного и программного обеспечения, необходимых для реализации атомарной атаки.

Для трасс атак рассматриваем схожие параметры, приписывая значение по следующим правилам:

- опасность в случае реализации трассы равна опасности реализации последней атомарной атаки;
- сложность реализации трассы равна наибольшему значению из сложностей реализации составляющих её атомарных атак;
- вероятность реализации трассы в случае обоих подходов равна произведению вероятностей, приписанных каждой составляющей её атомарной атаке;
- стоимость трассы равна сумме стоимостей атомарных атак, образующих её;
- и для трасс можно рассмотреть еще один параметр — их длины, количество атомарных атак, составляющих трассу.

2.3. Оценка количества состояний

Попытаемся оценить количество состояний получившегося автомата или, что то же, объем множества AV. Для этого рассмотрим 2 основные характеристики векторов из AV: их длины и размерность по каждой координате.

- *Длины векторов.* Длина каждого вектора напрямую зависит от размера рассматриваемой информационной сети и количества интересующих служб/файлов. Каждая координата вектора из AV отождествляется с некоторым действием (иногда набором действий), которые доступны (или не доступны, в зависимости от значения) нарушителю. Эти действия совершаются над объектами (файлами/службами), соответственно, чем больше объектов рассматривается и чем разнообразнее операции с ними, тем больше будет размерность AV.
  К важным службам относят: сетевой обмен данными, ssh серверы, системные процессы, процессы аутентификации, почтовые клиенты, месседжеры, клиенты, имеющие отношение к электронным деньгам и т. д. В координаты векторов множества AV могут включатся такие действия с подобными службами, как запуск, остановка, перенаправление потока данных.
  Важными же файлами в первую очередь являются различные базы данных (БД), как то: клиентские БД, БД паролей, БД сотрудников и так далее — а также аккаунты (соцсетей, электронной почты, онлайн игр, используемые для удаленной работы), финансовая данные, управление банковскими вкладами, документы (SSN, паспортные данные), данные электронных денег (CCN, пароли и номера пластиковых карт, аккаунты торговых площадок таких, как Ebay) [23]. Рассматривать можно такие действия с данными, как открытие, копирование, изменение, удаление, создание, передача по электронной почте, запуск (например, исполняемых файлов, сценариев) [24].
- *Размерности координат.* Чаще всего в координате, отвечающей определенному действию над соответственным файлом/службой, возможны лишь значения 1 (если в таком состоянии системы нарушитель имеет права на выполнение операции) или 0 (иначе). В таком случае AV будет состоять из 2^N бинарных векторов длины N.
  Однако, учитывая объем современных информационных сетей и хранящихся в них данных, зачастую удобнее говорить не о

каждом действии в отдельности, а классифицировать данные по конфиденциальности и пользователей по уровням доступа. Таким образом, один параметр (равно, координата в векторах из AV) может отвечать сразу за несколько возможных операций над множеством файлов/служб. Например, по европейским стандартам информация подразделяется на открытую, конфиденциальную и строго конфиденциальную [25]. Согласно же действующему законодательству Российской Федерации можно применить следующее разграничение информации по грифу конфиденциальности: открытая информация, для внутреннего использования, конфиденциальная информация [26]. Также возможно группировать информацию исходя из имеющейся типизации пользователей. Пользователей, в свою очередь, по уровню доступа можно классифицировать подобно тому, как это делается в различных ОС (например: guest,user, admin, root), или исходя из структуры управления защищаемой организации (например, карьерная лестница, а, следовательно, и уровни доступа, компании может быть следующей: стажер, консультант, старший консультант, младший менеджер, менеджер, старший менеджер, директор, партнер) [27]. Подробнее можно ознакомиться в [43].

2.4. Альтернативные программные средства и базы уязвимостей

В данный момент помимо CVSS существует немало баз и предлагаемых методов оценки показателей уязвимости. Вот далеко не полный список:

1) *CVE* [28]. По сути, это словарь известных уязвимостей, имеющий строгую характеристику по описательным критериям. CVE можно отыскать в Национальной Базе Уязвимостей США [29] или на официальном сайте.
2) *BID* [30]. Одна из отличительных особенностей совместимость с CVE [31].
3) *OSVDB* [32]. Присутствуют дополнительные характеристики: локация эксплуатации (сетевой доступ/локальный доступ) и импакт (ущерб от уязвимости, воздействие на какую-либо часть целевой информационной системы).

4) *Secunia* [33]. Как и в случае Nessus, база уязвимостей одноименного сканера [34].
5) *ISS X-Force* [35]. Помимо прочих параметров описывает материальный ущерб, который может повлечь за собой угроза эксплуатации.

Также и Nessus далеко не единственный сканер безопасности. Вот некоторые из часто используемых:

1) *Acunetix*. Включает в себя веб-сканер, паук, инструменты анализа отчетов и баз данных проверяет всех ведущие веб-серверные платформы.
2) *Cenzic Hailstorm* [37]. Сканер может анализировать безопасность веб-приложений.
3) *GFI LANguard* [38]. Сетевой сканер безопасности включает в себя модули сетевого поиска уязвимостей, управления патчами и модуль аудита.
4) *Nmap* [39]. Сканер портов. Бесплатный и открытый. Утилита для проверки сетевой безопасности или аудита безопасности.
5) *Retina* [40]. Сканер обнаруживает известные и потенциальные уязвимости. Продукт также оценивает риски, предлагает лучшие решения в области безопасности, политики и аудита.
6) *SAINT 8* [41]. Сетевой сканер уязвимостей с интегрированным инструментом тестирования возможности проникновения, позволяющего пользователям использовать найденные уязвимости.
7) *XSpider* [42]. Средство сетевого аудита, предназначенное для поиска уязвимостей в различных сетевых операционных системах и аппаратных устройствах.

3. Алгоритмы решения поставленных задач

3.1. Формирование наиболее полного списка небезопасных узлов сети

В терминах построенной математической модели, задача переформулируется, как описание всех достижимых состояний из начального состояния q_0 и возможных путей до них. Предлагается отождествлять состояния по принципу достижимости: если состояние q

достижимо из состояния q_0, то отождествляем их. Таким образом, пока возможно, перебирая все вершины, соседние отождествленным, и запоминая пути к ним, решим поставленную задачу. Делать это можно, например, известными алгоритмами поиска в глубину либо в ширину. Тогда за время, линейно зависящее от суммы количества ребер и вершин, построим необходимый список.

3.2. Возможность компрометации выделенного ресурса

В терминах построенного автомата V задача звучит как: «Достижимо ли хотя бы одно из выделенных состояний (обозначим множество выделенных состояний как Q') из состояния q_0?». Эту проблему можно рассматривать как подзадачу предыдущей и использовать схожие алгоритмы. С одной стороны, меньше чем за линейное время мы не сможет решить задачу, так как необходимо, все же, просмотреть всю компоненту связанности, в которой находится состояние q_0, с другой же реальные сети достаточно объемны, поэтому даже линейное время работы — это, в каком-то смысле, долго. Но имеет значение порядок, в котором будут перебираться вершины. Вспомним, что на состояниях введен частичный порядок. Предлагается 2 противоположных подхода:

1) Если допустить достаточно правдоподобное предположение, что чем больше привилегий получено, тем больше времени на это понадобилось (тем больше промежуточных состояний пришлось пройти), то можно предложить альтернативный порядок обхода. Поставим вспомогательный вопрос: можно ли из состояния q_0 достигнуть множества Q' (суть какой-либо вершины из Q'), двигаясь по «меньшим» состояниям? Будем сначала рассматривать состояния, которые строго меньше Q' (каждого состояния из Q') и только потом — несравнимые.
Заведем 2 очереди: S_1 и S_2. В S_1 будем хранить только те состояния, которые меньше Q', а в S_2 — несравнимые с q_1. Изначально помещаем q_0 в соответствующее множество, другое оставляем пустым. Начинать каждый шаг поиска в ширину будем с вершин, лежащих в S_1: текущая вершина выбирается первой нерассмотренной ранее в S_1, в случае, если таких не оказывается, то первой нерассмотренной ранее из S_2. На каждом шаге всех соседей текущей вершины записываем в соответствующую очередь (если же нашли соседнюю вершину больше либо равную

какой-нибудь вершины из Q', то завершаем алгоритм и выдаем положительный результат — Q достижимо). Таким образом переходим к рассмотрению вершин из S_2 только тогда, когда S_1 становится пустым. Если же оказывается так, что и $S_1 = \varnothing$, и $S_2 = \varnothing$, то завершаем алгоритм и выдаем отрицательный ответ: Q' недостижимо из состояния q_0.

2) С другой стороны, возможна ситуация, когда получение широких прав доступа не затруднительно и они позволяют скомпрометировать немало ресурсов, в том числе и выделенный. Суть алгоритма заключается в том, чтобы при обходе в ширину двигаться в направлении наибольшего роста привилегий, то есть получать как можно бóльшие привилегий за 1 шаг.

3.3. Описание всевозможных атакующих действий на выделенный ресурс

В терминах модели задача звучит как: «Вывести всевозможные трассы из начального состояния q_0 до выделенного множества состояний Q'».

Будем решать задачу, используя рекурсивную функцию, выводящую все искомые трассы, назовем её $Track(tr, Q')$, где tr — уже построенная часть трассы. Эта функция будет удлинять трассу, дописывая в конец поочередно каждое ребро, ведущее из последней вершины трассы в «новые» (через которые не проходила трасса или, что тоже, в которую не входит ни одно ребро из трассы) , и запускать себя, передавая в качестве аргументов уже эти удлиненные трассы и Q'. И так, пока не получим трассу, в которой последняя вершина принадлежит Q' (тогда выводим эту трассу), либо не сможем удлинить трассу, это будет означать, что она проходит по всем вершинам, соседним конечной.

Чтобы получить ответ для начального состояния q_0, нужно запустить функцию $Track()$ для каждого перехода, выходящего из q_0 и Q'.

3.4. Описание всевозможных атакующих действий из начального положения злоумышленника

В терминах модели задача звучит как: «Вывести состояния, достижимые из начального q_0 и всевозможные трассы до них».

Можно решать эту задачу используя уже указанные алгоритмы: сначала построить множество Q' достижимых из q_0 состояний, а потом запустить функцию $Track()$, передавая ей в качестве второго аргумента последовательно каждое состояние из Q' (первым аргументом, очевидно, будет q_0).

Можно же поступить немного иначе: запустить $Track(q_0, \varnothing)$, которая выведет всевозможные трассы, а потом сгруппировать эти трассы по конечной вершине.

3.5. Описание всевозможных атакующих действий, использующих выделенную уязвимость

В терминах модели задача формулируется как: «Вывести все трассы атак, содержащие переход a».

Можно, используя описанные выше алгоритмы, найти всевозможные трассы, а потом оставить только содержащие a.

Но можно привести и более эффективный алгоритм: Сначала воспользуемся функцией $BackTrack(tr, Q')$ — она действует, как $Track(tr, Q')$, но идёт в обратном направлении, то есть дописывает переходы в начало трассы, пока не получим трассу, выходящую из Q'. Запустим функцию $BackTrack(a, \{q_0\})$ затем $Track(tr, \varnothing)$, передавая ей в качестве первого аргумента последовательно каждую из трасс, выданных функцией $BackTrack(tr, Q')$. Можно использовать похожий алгоритм, не прибегая к описанию дополнительных алгоритмов: вместо $BackTrack(a, \{q_0\})$ можно сгенерировать все трассы от q_0 до вершины, из которой выходит a (см. п. 3.3), затем дописать в конец каждой a, далее действуем так же.

4. Подходы к уменьшению автомата

4.1. Введение входного алфавита

Попытаемся изменить автомат так, чтобы переходы были не Λ-переходами. Это будет полезно при сопоставлении каждой возможной трассе атаки слова в некотором алфавите и уменьшит количество состояний в детерминированном автомате, моделирующем работу автомата V. Для этого необходимо более тонко определить понятие атомарной атаки. Будем считать *видом атомарной атаки* последовательность действий, которая может быть ассоциирована с несколькими (возможно, одним) узлами информационной сети, а *атомарной*

атакой — вид атомарной атаки вместе с конкретными узлами сети, которые он затрагивает. Или другими словами: атомарная атака — это вид атаки от двух аргументов — соответствующих узлов сети. Поясним на примере.

Пример. Пусть сеть состоит из двух компьютеров: Comp1 и Comp2. Тогда Buffer overflow — это вид атаки, который порождает 2 атомарные атаки: Buffer overflow(Comp1,Comp1) и Buffer overflow(Comp2,Comp2). Обе эти атаки принадлежат одному виду, так как по сути выполняются одни и те же действия, только на разных машинах.

Пусть $A = \{a_1, a_2, \ldots, a_m\}$ множество видов атомарных атак. Мы будем рассматривать в основном виды атак, ассоциированные с одной или двумя машинами. В последнем случае чаще всего это атакующая и атакуемая машины. Каждому переходу, соответствующему какой-либо атомарной атаке, припишем вид этой атаки и номер атакуемого узла. Назовем эти переходы A-переходами, тогда в автомате будут присутствовать A- и Λ-переходы (но от меньших состояний к большим ведут только A-переходы, так как для расширения привилегий необходимо выполнить некие атакующие действия).

Замечание. Номер узла дописывается для того, чтобы исключить возможность выхода из одного состояния 2 переходов с одинаковыми подписями. Например, пусть состоит из 3 узлов, злоумышленник имеет права администратора на 1-ой машине, тогда атака Port-forward (пусть ей соответствует буква p в алфавите A) может быть осуществлена на узел 2, либо на узел 3. Если не приписывать номер атакуемой машины, то из начального состояния автомата V будут выходить 2 стрелки с подписью p.

4.2. Использование монотонности и классификации уязвимостей

Теперь можно произвести следующим модификации, которые помогут уменьшить автомат, не ущемляя при этом его применимости:

- Монотонность.
  Предполагаем, что никакие действия злоумышленника не мешают ему выполнять другие его действия. Другими словами, что получение новых прав доступа не уменьшает текущие. То есть если в какой-то момент времени злоумышленник имел определенные права доступа, то в результате использования им рас-

сматриваемых атомарных атак видов A эти права могут только расшириться. Об обосновании использования монотонности можно подробнее ознакомиться в [44]. Тогда:

1) В построенном автомате отсутствуют A-переходы от бóльших состояний к меньшим.
2) В построенном автомате отсутствуют A-переходы между несравнимыми состояниями.
3) Можно исключить Λ-переходы от больших состояний к меньшим, так как злоумышленник не будет двигаться в таких направлениях.
4) По этой же причине исключаются Λ-переходы между несравнимыми состояниями.

Таким образом, в автомате останутся только A-переходы (и вести они будут только от меньшего состояния к большему). Теперь зафиксировав начальное состояние автомата V, можно задать каждую трассу атаки словом из алфавита $A \times \{1 \dots m\}$ (m — номер атакуемого узла): просто последовательно переписываем подписи на ребрах.

Также автомат, построенный с использованием принципа монотонности, будет обладать одним важным свойством: при решении задачи обнаружения всех достижимых состояний, как мы помним, ответом служило множество несравнимых вершин (все достижимые из начальной были меньше либо равны какой-то вершины из ответа), теперь же в искомом множестве будет только одна вершина.

- Разделение уязвимостей на классы.
  Полагаем, что злоумышленник во время атаки перемещается последовательно от одной машины к другой. Иными словами, в каждый момент времени выполняет атакующие действия только на одном узле. Таким образом, можно разделить все рассматриваемые виды уязвимостей на 2 класса: для которых атакующая и атакуемая машины являются различными узлами сети (назовем их *сетевыми*, а класс — NV(Network vulnerabilities)) и атакующие машину, на которой они выполняются (назовем их *внутренними*, а класс — IV (Internal vulnerabilities)). Для атак IV-класса можно считать, что они зависят только от одного аргумента.

  Для данной информационной сети, состоящей из m узлов, назовем автоматы V_i ($i = 1 \dots m$) *узловыми*, если переходы автоматов V_i суть внутренние атомарные атаки, возможные на уз-

ле с номером i. Благодаря классификации уязвимостей, можно рассматривать автомат V, как композицию узловых автоматов $V_1, V_2, \ldots, V_m$. Соединяются же автоматы V_i посредством переходов, соответствующих сетевым атомарным атакам, то есть переход между состояниями, принадлежащими различным узловым автоматам может быть только класса NV.
Так как мы рассматриваем атомарные атаки NV-класса, ассоциированные только с 2 узлами, разумное предположение, что для реализации атомарной атаки достаточно обладать правами администратора на атакующей машине, позволяет существенно уменьшить количество *состояний*: теперь каждому состоянию будет соответствовать вектор-состояние не всей системы, а только права доступа на узле, которому принадлежит состояние.

В таком автомате будет уже не $2^N = 2^{\sum n_i}$состояний, а $\sum_{i=1}^{m} 2^{n_i}$.

Замечание. Подобным образом уменьшить количество состояний можно, сделав, например, предположение о том, что злоумышленник не возвращается на узел, из которого уже выходил. Во введенных терминах это означает следующее: для каждой трассы атаки каждый узел сети в качестве атакующей машины в её атомарных атак NV-класс может встретиться не более одного раза.

Вместе с тем классификация атомарных атак позволяет значительно уменьшить и количество *ребер*. Пусть при построении автомата V рассматривалась атомарная атака IV-класса, изменяющая r-ю координату в векторе-состоянии с 0 на 1, причем для её выполнения достаточно было, лишь чтобы одна другая координата, пусть s-я была равна 1. Тогда в автомате V было 2^{N-2} переходов, по сути соответствующих одной этой атомарной атаке (даже не классу атак, а атаке). В предложенной альтернативе переходов, отвечающих данной атомарной атаке, будет уже 2^{n_i-2} (считаем, что она выполняется на i-м узле) Если же рассматривать подобный вид атомарных атак и считать, что этот вид порождает m атомарных атак (то есть такая атака возможна на всех узлах сети), то переходов, отвечающих данному виду атомарных атак, в автомате V будет уже $m \cdot 2^{N-2}$, а для другого подхода: $\sum_{i=1}^{m} 2^{n_i-2}$.

Описанные выше подходы существенно уменьшают объем графа атак, что упрощает практическую реализацию. При этом область применимости автомата не уменьшается. Помимо облегчения реализации введение входного алфавита и декомпозиция автомата дает возможность выделять самые распространенные способы проникновения в систему, наименее защищенные узлы и принимать меры по защите информационной сети.

Список литературы

[1] Blum A. L., Furst M. L. Fast Planning Through Planning Graph Analysis // Artificial Intelligence — AI. — 1997. Vol. 90, no. 1–2. — P. 281–300.

[2] Sheyner O., Wing J. M. Tools for Generating and Analyzing Attack Graphs // Proceedings of the conference: Formal Methods for Components and Objects. — 2003. — P. 344–372.

[3] Phillips C. A., Swiler L. P. A graph-based system for network vulnerability analysis // New Security Paradigms Workshop. — 1998. — P. 71-–79.

[4] Sheyner O. Scenario Graphs and Attack Graphs / PhD thesis. — Carnegie Mellon University, 2004.

[5] Sheyner O., Haines J., Jha S., Lippmann R., Wing J. Automated generation and analysis of attack graphs // Proceedings of the IEEE Symposium on Security and Privacy. — 2002. — P. 273–284.

[6] Swiler L. P., Phillips C., Ellis D., Chakerian S. Computer-attack graph generation tool // Proceedings of the DARPA Information Survivability Conference and Exposition. — 2001.

[7] Ghosh N., Ghosh S. K. A planner-based approach to generate and analyze minimal attack graph // Applied Intelligence — APIN. — 2012. Vol. 36, iss. 2. — P. 369–390.

[8] Jha S., Sheyner O., Wing J. M. Two Formal Analyses of Attack Graphs // Proceedings of the conference: Computer Security Foundations Workshop — CSFW. — 2002. — P. 49–63.

[9] Ghosh N., Ghosh S. K. An Intelligent Technique for Generating Minimal Attack Graph. — Indian Institute of Technology, Kharagpur–721302, 2009.

[10] Кудрявцев В. Б., Алешин С. В., Подколзин А. С. Введение в теорию автоматов. — М.: Наука, 1985.

[11] Kotenko I., Stepashkin M. Analyzing Vulnerabilities and Measuring Security Level at Design and Exploitation Stages of Computer Network Life Cycle // Third International Workshop on Mathematical Methods, Models and Architectures for Computer Network Security, MMM–ACNS 2005. — St. Petersburg, Russia, September 24–28, 2005. — P. 311–324.

[12] Материалы для ознакомления со сканнером Nessus. [http://www.nessus.org].

[13] The Open Web Application Security Project. [https://www.owasp.org].

[14] The MITRE Corporation. [http://www.mitre.org].

[15] The SANS Institute. [http://www.sans.org/about].

[16] The ten most critical web application security risks. The Open Web Application Security Project / Ежегодный отчет OWASP. — 12.06.2013. [https://www.owasp.org/index.php/OWASP_Top_10].

[17] CWE/SANS TOP 25 Most Dangerous Software Errors. — 27.06.2011. [http://www.sans.org/top25-software-errors].

[18] Improper Neutralization of Special Elements used in an SQL Command. [http://cwe.mitre.org/top25/index.html#CWE-89].

[19] Improper Neutralization of Special Elements used in an OS Command. [http://cwe.mitre.org/top25/index.html#CWE-78].

[20] Improper Neutralization of Input During Web Page Generation. [http://cwe.mitre.org/top25/index.html#CWE-79].

[21] Missing Authentication for Critical Function. [http://cwe.mitre.org/top25/index.html#CWE-306].

[22] Информация о базе уязвимостей CVSS. [http://nvd.nist.gov/cvss.cfm].

[23] Панасенко А. Какую информацию воруют и что реально нужно защищать? [http://www.anti-malware.ru/node/2646].

[24] ГОСТ Р ИСО/МЭК 17799–2005. Практические правила управления информационной безопасностью.

[25] ISO/IEC 17799:2000. Information technology — Code of practice for information security managment.

[26] Костров Д. Защита информации. Конфидент # 2. — 2004. [http://bre.ru/security/21477.html].

[27] Карьерная лестница в компании PwC. [http://www.pwc.ru/ru/careers/career-path.jhtml].

[28] Common Vulnerabilities and Exposures. [http://cve.mitre.org].

[29] National Vulnerability Database. [http://nvd.nist.gov].

[30] Security Focus Bugtraq ID database entry. [http://www.securityfocus.com/bid].

[31] Список соответствий между базами уязвимостей BID и CVE. [http://cve.mitre.org/data/refs/refmap/source-BID.html].

[32] Open Sourced Vulnerability Database. [http://osvdb.com].

[33] Secunia Vulnerability Database. [http://secunia.com/community/advisories/search].

[34] Материалы для ознакомления со сканером Secunia. [http://secunia.com].

[35] Internet Security Systems X-Force Vulnerability Database. [http://xforce.iss.net].

[36] Acunetix Web Vulnerability Scanner. [http://www.acunetix.com/vulnerability-scanner].

[37] Cenzic Hailstorm. [http://www.cenzic.com].

[38] GFI LanGuard network security scanner and patch management. [http://www.gfi.com/products-and-solutions/network-security-solutions/gfi-languard].

[39] Network Mapper. [http://nmap.org].

[40] Retina Network Security Scanner. [http://www.beyondtrust.com/Products/RetinaNetworkSecurityScanner].

[41] SAINT Network Vulnerability Scanner. [http://www.saintcorporation.com/products/SAINT8.html].

[42] Сканер XSpider. [http://www.ptsecurity.ru/xs7].

[43] Материалы для ознакомления с управлением доступом на основе ролей. — National Institute of Standards and Technology. [http://csrc.nist.gov/groups/SNS/rbac/index.html].

[44] Ammann P., Wijesekera D., Kaushik S. Scalable, Graph-Based Network Vulnerability Analysis // ACM Conference on Computer and Communications Security — CCS. — 2002. — P. 217–224.