Секция 4

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

УДК 519.7 DOI 10.17223/2226308X/11/28

ЭФФЕКТИВНЫЙ АЛГОРИТМ ПОСТРОЕНИЯ МНОЖЕСТВА КРАТЧАЙШИХ АТАК В РАМКАХ ОДНОЙ МОДЕЛИ РАЗВИТИЯ АТАК В КОМПЬЮТЕРНОЙ СЕТИ[1]

Д. Е. Горбатенко, А. А. Семёнов

Рассматривается задача компактного представления множества всех кратчайших атак в компьютерной сети. Для одной модели развития атак показывается, что задача имеет эффективное решение. Описывается алгоритм с временной сложностью $O(n^2)$, где n — число хостов в сети, который строит множество всех кратчайших атак в форме графа специального вида.

Ключевые слова: *атаки в компьютерных сетях, графы атак, дискретные динамические системы.*

Впервые идея представлять множество атак в сети в виде графа специального вида высказана в [1], где были использованы программные средства решения задач символьной верификации. Однако довольно быстро стало ясно, что такой способ представления атак неэффективен с вычислительной точки зрения. В [2, 3] и ряде других работ предложены иные подходы к построению графов атак. Ценность работы [2] заключается в том, что в ней впервые представлена модель развития атак, в рамках которой граф, представляющий все атаки в сети, строится за полиномиальное от числа хостов сети время. Помимо собственно генерации графов атак, в [1, 2] исследованы задачи оптимизационного характера. Так, в [1] показано, что в рамках использованной в данной работе модели задача поиска «минимальной атаки» является NP-трудной. В [2] описан полиномиальный алгоритм поиска «локально минимальной атаки». По-видимому, наиболее эффективными на сегодняшний день способами представления атак являются модели, описанные в [3 – 5]. На наш взгляд, основной недостаток этих моделей заключается в том, что получаемые в них графы содержат петли, что не позволяет корректно определять длину атаки. В настоящей работе рассматривается модель развития атак в сети, при которой задача представления множества кратчайших атак имеет эффективное решение.

Следует отметить, что во многих ключевых работах по графам атак, к сожалению, недостаточно чётко прописана система базовых понятий — абстратные понятия часто путаются с конкретными. Такая ситуация имеет место, например, по отношению к понятию «доступ». Наилучшим образом в плане таксономии организована диссертация М. Данфорт [6], но и в ней имеется ряд упущений. Например, Данфорт выделяет два уровня доступа, которые злоумышленник может иметь на атакуемом им хосте: это «user» и «root». Эти формы доступа злоумышленник получает, эксплуатируя уязвимости, имеющиеся на хосте. Возникает вопрос об уровне доступа, который был у

[1]Работа выполнена при финансовой поддержке Российского научного фонда, проект № 16-11-10046.

злоумышленника в тот момент, когда он провёл первую элементарную атаку. Этот и другие похожие вопросы никак не акцентируются в подавляющем большинстве работ. Далее мы пытаемся исправить указанные недостатки: строим конкретный пример набора базовых понятий, в рамках которого доказываем основной результат о сложности представления множества кратчайших атак. Используемая система понятий и соответствующая ей модель являются весьма простыми, однако при необходимости их легко дополнить до уровня, достаточного для описания атак в реальных компьютерных сетях.

Будем рассматривать компьютерную сеть в виде простого неориентированного помеченного графа $G = (V, E, L_V)$. Здесь V — множество вершин, называемых далее хостами, $|V| = n$. Выделим в V две вершины: v_M (M от Malefactor) — хост злоумышленника, v_T — «целевой хост». Через E обозначено множество рёбер графа G. Произвольное ребро $(v_1, v_2) \in E$ отображает прямую физическую связь между хостами v_1 и v_2, простейшая конкретизация — проводное соединение компьютера с маршрутизатором. Окрестность вершины $v \in V$ — множество смежных с ней вершин. С хостом $v \in V$ связывается двоичное слово l_v, называемое вектором уязвимостей. Множество всех слов вида l_v обозначается через L_V.

Далее потребуется такое важнейшее понятие, как «доступ». В соответствии с [7] доступ объекта A (обычно называемого субъектом) к объекту B — это возможность A читать, а в некоторых случаях и модифицировать данные, которыми располагает B. В рамках рассматриваемой далее (базовой) модели развития атак целесообразно выделить следующие три уровня доступа:

1) Уровень доступа, обозначаемый user_0, хоста v' к хосту v'' предполагает возможность v' читать вектор $l_{v''}$. Данный уровень доступа — то, чего не хватает в упомянутых выше работах. Он соответствует ситуации, когда хост v' (v' может быть как хостом v_M, так и хостом, на котором M получил некие права) может физически связаться с хостом v'' (например, сканируя некоторые порты).
2) Уровень доступа, обозначаемый user, хоста v' к хосту v'' предполагает, что v' может вычислять все функции из некоторого списка S, аргументами которых являются координаты вектора $l_{v''}$.
3) Уровень доступа, обозначаемый root, хоста v' к хосту v'' предполагает, что v' получает право user_0 на все хосты, находящиеся в окрестности v'' в графе G.

Здесь необходимо оговориться, что в контексте представления атак на сеть G нас интересуют только действия злоумышленника. Везде далее предполагаем, что повышение прав доступа с хоста v' на хост v'' осуществляет M, который действует «от имени» v'. В таком контексте удобно рассмотреть развитие атак в сети как дискретный динамический процесс: например, получению M права user на v'' в момент $t + 1$ предшествует получение M права user_0 на v'' в момент t плюс некоторые дополнительные действия. При этом получению M права user_0 на v'' в момент t предшествует получение M в момент $t - 1$ права root на хосте v', таком, что $(v', v'') \in E$.

Далее понадобится ещё одно ключевое понятие — уязвимость. Уязвимости хоста v — это условия, наличие которых позволяет злоумышленнику, атакующему данный хост, поднять свой уровень доступа к v. Через U будем обозначать список всех возможных уязвимостей в сети. В произвольном векторе l_v координаты, равные 1, указывают на те уязвимости из U, которые присутствуют на хосте v.

Повышение прав доступа (*privilege escalation* [3, 4, 6]) M с хоста v' на хост v'' (с user_0 до user и root) происходит за счёт процесса, называемого эксплуатацией уяз-

вимостей. Эксплуатация уязвимостей — это последовательность вычислений значений функций из некоторого списка S. Каждое такое вычисление называется элементарной атакой [6] (или правилом взаимодействия [3, 4]). Тот факт, что при переходе от момента t к моменту $t+1$ M получает на v более высокий уровень доступа, будем отображать в специальном регистре (одномерном массиве), который связывается с M в моменты времени $t = 0, 1, \ldots$ Состояние регистра в момент t будем обозначать $R(t)$. В регистре R в момент t с каждым хостом v связано множество ячеек, разбитое на три подобласти. Первая подобласть, обозначаемая a_v^t, содержит один бит, который равен 1, если M в момент t имеет на v доступ user_0. Вторая подобласть b_v^t, в свою очередь, разбивается на две подобласти. Первая содержит один бит, который равен 1, если M в момент t имеет на v доступ с правом user. Во второй хранится множество пар вида (u, f), $u \in U$, $f \in S$: каждая такая пара отображает тот факт, что M в момент t реализовал элементарную атаку f, эксплуатирующую уязвимость u, и получил в результате право user на хосте v. Аналогично устроена область c_v^t, в которой содержится информация о получении M права root на v в момент t.

Атака на сеть состоит из элементарных атак, последовательно реализуя которые, M получает доступ на всё большее число хостов сети. Атака считается успешной, если существует такая последовательность элементарных атак, реализуя которую, M в некоторый момент времени получает право root на целевой хост v_T.

Далее постулируем следующие два факта (первый акцентируется во всех отмеченных выше работах, второй явно упоминается не всегда):

1) Требование монотонности: будем считать, что любое действие, совершённое злоумышленником в момент $t \geqslant 1$, не может быть отменено в последующие моменты времени. Это означает, что если в некоторый момент t в некоторую ячейку регистра R была записана 1, то в каждый момент вида $t+e$, $e \geqslant 1$, в данной ячейке в состоянии $R(t+e)$ также будет находиться 1.
2) Полагаем, что мощности множеств U и S — константы, не зависящие от n.

Учитывая второе свойство, отмечаем, что размер областей b_v^t и c_v^t ограничен константой, соответственно длина регистра R — $\mathrm{O}(n)$ бит.

Опишем алгоритм построения множества атак в сети G, основанный на процедуре обхода G (вариант обхода в ширину). Действуя в стиле [8], будем представлять развитие атак в G как дискретный динамический процесс, происходящий в моменты $t = 0, 1, \ldots$ Состояния соответствующей дискретной динамической системы (ДДС) — это $R(t)$. Все состояния регистра $R(t)$ сохраняются.

Алгоритм состоит из трёх стадий: начальной, инициализации и переходов в новые состояния. Начальная стадия соответствует моменту $t = 0$. Стадий инициализации и переходов в новые состояния может быть несколько. Приведём неформальное описание алгоритма.

B (начальная стадия). Строится состояние $R(0)$ — нулевой вектор.

I (стадия инициализации). Вход: множество хостов $W^i = \{v_{i_1}, \ldots, v_{i_p}\}$, выход: множество хостов $W^o = \{v_{o_1}, \ldots, v_{o_q}\}$ (детализация далее).

T (стадия переходов в новые состояния). Вход: множество W^o. Полагаем, что построены состояния $R(0), \ldots, R(t)$.

1) Рассматривается момент t. В состоянии $R(t)$ области a_v^t, b_v^t, c_v^t заполнены нулями для всех $v \in W^o$. В момент $t+1$ $a_v^{(t+1)} = 1$ для всех $v \in W^o$. Итог: состояние $R(t+1)$.

2) Рассматривается момент $t+1$. По всем вершинам $v \in W^o$ проверяется, может ли M получить на них право user. Выделяется множество $\widehat{W}_o \subseteq W^o$ вершин, на которых M получает право user, вычисляя некоторые функции из S. Соответствующая информация записывается в область $b_v^{(t+2)}$. Итог: состояние $R(t+2)$.
3) Рассматривается момент $t+2$. По всем вершинам $v \in \widehat{W}_o$ проверяется, может ли M получить на них право root. Выделяется множество $\widetilde{W}_o \subseteq \widehat{W}_o$ вершин, на которых M получает право root, вычисляя некоторые функции из S. Соответствующая информация записывается в область $c_v^{(t+3)}$. Итог: состояние $R(t+3)$.

Конец стадии переходов. Множество $\widetilde{W}_o$ является входом следующей стадии инициализации.

Алгоритм завершает работу в некоторый момент t_* тогда и только тогда, когда $R(t_*) = R(t_*+1)$.

Остановимся более детально на стадиях инициализации. Первая стадия инициализации — это выбор всех вершин из G, смежных с v_M. Обозначим данное множество W и пометим все вершины в нём как пройденные. Множество W — выход первой стадии инициализации (к W применяется стадия переходов в новые состояния). Пусть W_k^i — вход стадии инициализации с номером $k \geqslant 2$. Инициализация I_k заключается в следующем: рассматриваются окрестности всех вершин из W_k^i и выбираются только те вершины, которые не были помечены как пройденные. Объединяем полученные множества вершин. Результат — множество W_k^o, все его вершины помечаются как пройденные.

Отметим, что алгоритм формирования множеств W_k^i, W_k^o близок по смыслу алгоритму обхода графа в ширину (требуется помечать некоторые вершины как пройденные и исключать их из дальнейшего рассмотрения). Таким образом, сложность данного этапа составляет (в худшем случае) $\mathrm{O}(n^2)$.

Легко понять, что алгоритм всегда завершит работу: в силу свойства монотонности обязательно найдётся такое t_*, что $R(t_*+1) = R(t_*)$ ($R(t_*)$ — неподвижная точка рассматриваемой ДДС). В силу определения массива R и предположения о множествах U и S значение t_* ограничено сверху величиной $\mathrm{O}(n)$. Полагая, что вычислительное устройство имеет прямой доступ к памяти, заключаем, что сложность алгоритма равна $\mathrm{O}(n^2)$. Отметим, что граф (рис. 1) является графом перехода между состояниями рассматриваемой ДДС. Будем обозначать его Γ_G. Покажем, как по графу Γ_G представить множество кратчайших атак.

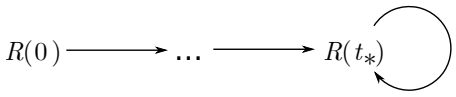

Рис. 1

Сначала построим специальный граф, обозначаемый $\Delta(G)$. Будем считать, что в состоянии $R(t_*)$ M имеет права root на хосте v_T, поскольку в противном случае множество успешных атак пусто. Обозначим через t_0, $t_0 \leqslant t_*$, момент, в который M получает право root на хосте v_T. Пусть $I_1, \ldots, I_d$ — стадии инициализации, пройденные алгоритмом до момента времени t_0. Будем считать, что выходом стадии I_d является множество вершин Ω_d^o, такое, что в окрестности некоторой вершины $w \in \Omega_d^o$ находится вершина v_T. Построим $(d+1)$-дольный граф $\Delta(G)$. В доле с номером $d+1$ находится одна вершина v_T. В доле с номером d находятся вершины из Ω_d^o, смежные с v_T в графе G. Данное множество вершин обозначим $\widehat{\Omega}_d^o$. В доле с номером r находятся

вершины, смежные с вершинами из $\widehat{\Omega}^o_{r+1}$, $r = d-1, \ldots, 2$. В доле с номером 1 графа $\Delta(G)$ находится одна вершина v_M.

Определение 1. Пусть A — произвольная успешная атака. Назовём длиной A число функций вида $f \in S$ (элементарных атак), значения которых были вычислены в процессе реализации A.

Лемма 1. В рамках рассмотренной модели развития атак в сети G длина кратчайшей атаки равна $2d$.

Построим граф $\Delta_*(G)$, который представляет все кратчайшие атаки в рамках рассматриваемой модели. Он строится на основе графа $\Delta(G)$: добавляем в граф $\Delta(G)$ дополнительные доли, в которых отображаются шаги из стадии переходов в новые состояния (стадия T), которые совершаются между соседними стадиями инициализации. Данная информация берётся из соответствующих состояний регистра R. На рис. 2 приведены две соседние доли графа $\Delta(G)$ и соответствующий им фрагмент графа $\Delta_*(G)$. Через S^u_j, $j \in \{l+1, \ldots, l+m\}$, обозначены списки элементарных атак, реализация которых (произвольной атаки из списка) приводит к получению M права user на хосте v_j. Аналогичный смысл (в отношении права root) имеет запись S^r_j ($j \in \{l+1, \ldots, l+m\}$).

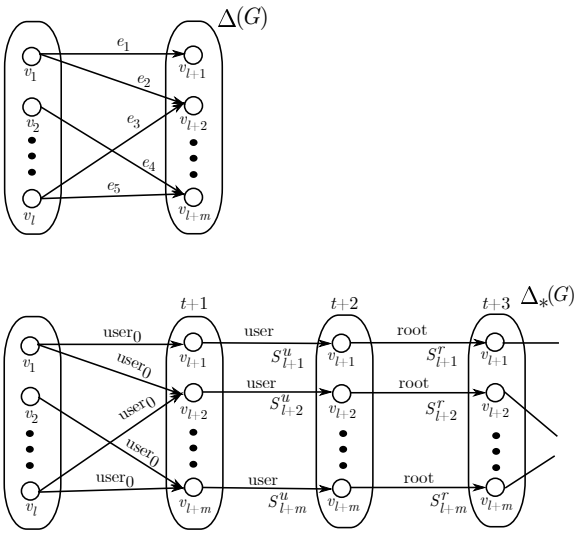

Рис. 2. Пример графов $\Delta(G)$ и $\Delta_*(G)$

Теорема 1. В рамках рассмотренной модели развития атак в сети G граф, представляющий все кратчайшие атаки в сети, строится за время $O(n^2)$.

Авторы благодарят рецензента за ряд конструктивных замечаний, учёт которых способствовал улучшению качества представляемого материала.

ЛИТЕРАТУРА

1. *Sheyner O., Haines J. W., Jha S., et al.* Automated generation and analysis of attack Graphs // Proc. 2002 IEEE Symp. Security and Privacy. 2002. P. 273–284.
2. *Amman P., Wijesekera D., and Kaushik S.* Scalable, graph-based network vulnerability analysis // Proc. CCS 2002: 9th ACM Conf. Computer and Communications Security. 2002. P. 217–224.
3. *Ou X., Wayne B., and Miles M.* A scalable approach to attack graph generation // Proc. 13th ACM Conf. Computer and Communications Security. 2006. P. 336–345.

4. *Ou X., Govindavajhala S., and Appel A.* MulVAL: A logic-based network security analyzer // SSYM'05 Proc. 14th Conf. USENIX Security Symp. 2005. V. 14. P. 113–128.
5. *Ingols K., Lippmann R., and Piwowarski K.* Practical attack graph generation for network defense // ACSAC'06: Proc. 22nd Ann. Computer Security Appl. Conf. 2006. P. 121–130.
6. *Danforth M.* Models for Threat Assessment in Networks. PhD Thesis, University of California-Davis, 2006.
7. *Девянин П. Н.* Модели безопасности компьютерных систем. М.: Academia, 2005.
8. *Горбатенко Д. Е., Кочемазов С. Е., Семёнов А. А.* О дискретно-автоматных моделях атак в компьютерных сетях // Прикладная дискретная математика. Приложение. 2016. № 9. С. 80–83.

УДК 004.94 DOI 10.17223/2226308X/11/29

ПОДХОДЫ К МОДЕЛИРОВАНИЮ УПРАВЛЕНИЯ ДОСТУПОМ В СУБД PostgreSQL В РАМКАХ МРОСЛ ДП-МОДЕЛИ

П. Н. Девянин

Широкое применение СУБД PostgreSQL в защищённых операционных системах, в том числе в операционной системе специального назначения (ОССН) Astra Linux Special Edition, требует разработки научно-обоснованных подходов к исследованию безопасности реализованного в этой СУБД механизма управления доступом. Для этого необходимо проанализировать как изначально используемое в СУБД PostgreSQL ролевое управление доступом, так и востребованные в практике создания защищённых систем мандатное управление доступом и мандатный контроль целостности. Известно, что научной основой при разработке механизма управления доступом в ОССН стала мандатная сущностно-ролевая ДП-модель (МРОСЛ ДП-модель), включающая перечисленные три вида управления доступом, имеющая иерархическое представление (позволяющее дополнять модель новыми элементами без её полной переработки), а также прошедшая проверку на корректность с применением инструментальных средств дедуктивной верификации. В связи с этим предлагаются подходы к построению в рамках иерархического представления МРОСЛ ДП-модели новых уровней, соответствующих механизму управления доступом в СУБД PostgreSQL. При этом из-за наличия существенных отличий этого механизма в ОССН и СУБД на первом этапе моделирования основное внимание уделено ролевому управлению доступом.

Ключевые слова: *компьютерная безопасность, формальная модель, управление доступом, PostgreSQL.*

С введением в действие ФСТЭК России с июня 2017 г. [1] профилей защиты операционных систем (ОС) общего назначения (типа «А»), включающих, начиная с третьего класса защиты, компоненты доверия ADV_SPM.1 «Формальная модель политики безопасности», AVA_CCA_EXT.1 «Анализ скрытых каналов» и ADV_FSP.5 «Полная полуформальная функциональная спецификация с дополнительной информацией об ошибках» [2], разработка научных подходов по формированию модели безопасности управления доступом, верификации её реализации, анализа на её основе безопасности информационных потоков (скрытых каналов) стала актуальной задачей для специалистов в области защиты информации. Можно ожидать, что в ближайшей перспективе ФСТЭК России будут введены аналогичные требования для систем управления базами данных (СУБД).