Е. П. Тумоян, Д. А. Кавчук

МЕТОД ОПТИМИЗАЦИИ АВТОМАТИЧЕСКОЙ ПРОВЕРКИ УЯЗВИМОСТЕЙ УДАЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Введение

Существующие в настоящее время сканеры уязвимостей обеспечивают обнаружение и идентификацию уязвимостей на основании косвенных признаков. Для проверки (валидации) обнаруженной уязвимости используются средства эксплуатации, такие как Immunity Canvas или Metasploit Framework. Одной из основных проблем при проведении проверки является необходимость массовой проверки уязвимостей. Ручная проверка большого количества уязвимостей требует от эксперта большого количества рутинных операций. С другой стороны, средства автоматической проверки (например, auto_pwn в Metasploit) выполняют последовательный запуск эксплойтов с учетом простых критериев, таких как сервис, целевая операционная система, ранг эксплойта. Вследствие этого процесс проверки занимает значительное время. Целью данной работы является разработка метода и программного средства оптимизации процесса автоматической проверки уязвимостей удаленной информационной системы. Предлагаемый в работе метод оптимизирует поиск средств эксплуатации уязвимостей на основе использования вероятностного дерева атак и искусственной нейронной сети с учетом специфики операционной системы, под управлением которой работает исследуемая ПЭВМ.

Разработанный метод позволяет провести проверку уязвимостей за меньшее количество попыток эксплуатации при заданной доверительной вероятности обнаружения всех известных уязвимостей. Разработанный метод также позволяет проводить проверку многостадийных атак (например, атак, связанных с эксплуатацией уязвимости в клиентском сервисе с последующим локальным повышением привилегий).

1. Существующие средства и академические разработки

Существуют два типа средств анализа безопасности: средства обнаружения уязвимостей (такие как Nessus Security Scanner, MaxPatrol, Nexpose) и средства тестирования на проникновение (Immunity Canvas, Metasploit Framework и другие). Средства тестирования на проникновение помимо основного назначения могут использоваться для проверки (валидации) уязвимостей, аудита паролей, модификации старых и создания новых эксплойтов и т. д.

Необходимость в проверке уязвимостей возникает, поскольку сканеры уязвимостей демонстрируют ложные срабатывания. Вероятность ложных срабатываний снижается при использовании агентов внутри анализируемой системы, которые предоставляют точную информацию о конфигурации и состоянии системы: такие средства есть в продуктах Retina или MaxPatrol. Однако в ряде случаев использование агентов невозможно по техническим или организационным причинам. Тогда актуальной становится задача оптимизации проверок уязвимостей с использованием информации, полученной сканером уязвимостей. В существующих системах данная проблема решается с учетом простых критериев, таких как ОС, сервис и ранг эксплойта. Соответственно, процесс проверки уязвимостей занимает значительное время и фактически плохо контролируем.

На данный момент существует ряд академических работ, позволяющих построить модель эксплуатации уязвимостей (или более широко — сетевой атаки) и оценить эффективность различных вариантов атаки. Данные модели используют разную математическую базу, но большинство из них основаны на конечных автоматах и представляют атаку как последовательность состояний автомата.

1) Деревья атак

Модель описана Б. Шнайером в работе [1] в 1999 г. Деревья атаки представляют собой концептуальные диаграммы, которые описывают угрозы системе и возможные атаки, направленные

на их реализацию. Деревья атак предоставляют возможность для введения оценок каждого шага по некоторым критериям, например по времени выполнения, числу операций, оценочной стоимости и т. д. При этом последовательность шагов может быть оценена на основании критериев для каждого шага. В работе [1] приведены примеры оценки деревьев на основе двух критериев, хотя нет принципиальных препятствий для разработки алгоритма многокритериальной оценки. Однако модель не обеспечивает средств для включения условий внешней среды — некоторого набора входных данных.

2) Улучшенные деревья атак

Модель описана в работе [2] в 2006 г. Модель опирается на использование деревьев и механизма конечных автоматов для моделирования уязвимостей протоколов и систем. Модель представляет собой расширение и уточнение модели деревьев атак. Авторами приводится пример анализа безопасности протокола IEEE 802.11. Данная модель имеет те же недостатки, что и предыдущая.

3) Графы атак

Модель предложена коллективом исследователей в работе [3] в 2002 г. Модель графов атак основана на расширении модели деревьев атак. Однако графы атак имеют следующие особенности:

• являются специализированным средством для описания сетевых атак;

• узлы графа представляют не концептуальные действия, а узлы сети, процессы программы, конфигурационные файлы, участки кода, переменные и т. д.

Моделирование систем на основе графов атак базируется на конечных автоматах. Переходы между узлами осуществляются на основе применения детерминированных правил. При этом может учитываться текущее значение некоторых параметров системы, переменных и т. д. Целью является достижение определенной вершины. Модель может предусматривать анализ условий, необходимых для достижения цели атаки, поэтому графы атак могут использоваться для оценки безопасности программной системы или компьютерной сети. Модель получила широкое распространение [3, 4], поскольку основана на простой и хорошо исследованной математической базе, сама достаточно проста и очевидна. Данная модель была реализована в нескольких инструментальных средствах, например в Lockheed Martin ANGI, а также в средстве AttackGraph Tool 0.5 [5].

Недостатком данной модели является то, что она представляет собой средство для оценки сложности нарушения безопасности информационной системы, а не моделирования и исследования атак. Другой существенный недостаток модели — применение аппарата конечных автоматов. Модель реальной информационной системы в большинстве случаев не ограничивается моделью автоматов, поскольку включает плохо контролируемые или скрытые факторы.

4) Interacting State Machines

Модель предложена коллективом исследователей в работе [6] в 2002 г. Модель основана на применении специального типа высокоуровневых конечных автоматов Interacting State Machines (ISM) для моделирования сложных систем. ISM применяются для моделирования атакуемого протокола с целью обнаружения ошибок, приводящих к уязвимостям системы, т. е. решаются задачи, существенно отличные от моделирования атак.

2. Метод оптимизации проверки уязвимостей

При разработке метода мы используем модель улучшенных деревьев атак, дополненную средствами учета условий внешней среды — входных данных.

Предлагаемый метод можно сформулировать следующим образом:

1) Получение информации об исследуемой системе — семейство, версия, service pack, список и баннеры сервисов.

$osInf = \{S, Ver, SP, serv, servB\}$,

где S — семейство ОС; $Ver = \{ver\_i\}$ — конечное множество (к.м.) предположительных версий ОС; $S\rho = \{s\rho\_i\}$ — к.м. предположительных service pack ОС; $serv = \{serv\_i\}$ — к.м. сервисов; $servB = \{servB\_i\}$ — к.м. баннеров сервисов.

2) Получение информации об уязвимостях.

$vulnInf = \{vuln\_i\}$,

где $vuln\_i$ — предположительная уязвимость системы.

3) Построение дерева атак для целевой системы. Узлы дерева представляют собой эксплойты, ребра — вероятности срабатывания листа дерева (эксплойта). Дерево включает все возможные эксплойты для данной операционной системы.

$G = (V, E)$,

где V — множество вершин: $V = \{exp\_i\}$, exp_i — допустимый эксплойт: $exp\_i = F(osInf, vulnInf)$; E — множество ребер: $E = \{\rho i\}$, ρi — вероятность срабатывания i-го эксплойта.

Построение дерева выполняется с использованием эвристического алгоритма на основе информации об уязвимостях системы. В частности, если эксплойт может сработать без предварительных условий, то он размещается в узле с уровнем, равным 1 (на следующем за корнем уровне дерева). Если же для успешного срабатывания эксплойта необходима реализация некоторых предварительных условий, то он размещается в поддереве, корнем которого является реализующий данные условия узел.

4) Уточнение дерева атак. Производится оценка вероятности срабатывания эксплойта с учетом информации об операционной системе и эксплойте $\{ osInf \mid vulnInf \}$. Для оценки вероятности могут использоваться различные техники, в частности таблицы. Однако поскольку пространство данных имеет большую размерность, то преобразование пространства данных в вероятность с использованием правил или таблиц не позволяет покрыть все варианты входных данных. Для решения этой проблемы используются обобщающие возможности нейронной сети. Вычисление вероятности производится по следующей формуле:

$\rho i = neuronet(osInf, vulnInf)$.

Обучение сети производится на основе экспертной оценки, уточненной при построении конкретного дерева атаки.

5) Обход дерева атак. Целью обхода дерева является нахождение упорядоченного по вероятности множества всех поддеревьев, эксплойты в которых актуальны для данной системы. Мы обходим дерево от корня к листьям по пути с наибольшей условной вероятностью срабатывания эксплойтов. При обработке очередного узла мы запускаем эксплойт с наиболее подходящими для данной операционной системы и набора сервисов параметрами. Если очередной узел не сработал, то мы производим усечение всего поддерева с корнем в данном узле и переходим далее, обходя оставшуюся часть дерева по убыванию условной вероятности путей.

6) Построение последовательностей эксплойтов. Мы строим цепочки сработавших эксплойтов с указанием возможностей, которые может получить атакующий при выполнении эксплойтов из цепочки.

На рис. 1 представлен пример работы предложенной модели. С учетом правил построения и обхода дерева можно утверждать, что в процессе обхода будут достигнуты все допустимые вершины дерева, т. е. эксплойты, которые могут сработать на данной ОС. Кроме того, для такого метода можно рассчитать вероятность того, что все актуальные для данной системы эксплойты будут обнаружены за заданное количество шагов.

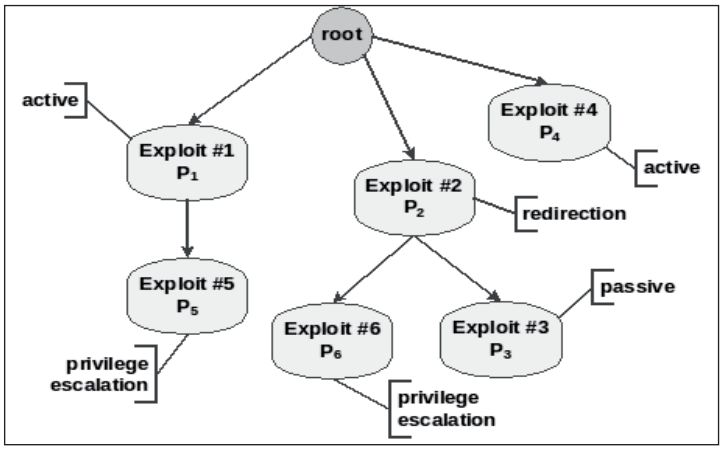

Рис. 1. Пример работы предложенной модели

Для решения задачи генерации вероятностей срабатывания эксплойтов была выбрана искусственная нейронная сеть на основе радиальных базисных функций [7]. На вход сети поступает вектор свойств, ассоциированный с рассматриваемым состоянием дерева.

3. Архитектура системы

Разработанная система включает следующие взаимодействующие компоненты: сканер уязвимостей Nessus Security Scanner, сервер системы Metasploit Framework и анализатор, разработанный в среде Matlab и взаимодействующий с другими компонентами системы. Архитектура системы представлена на рис. 2.

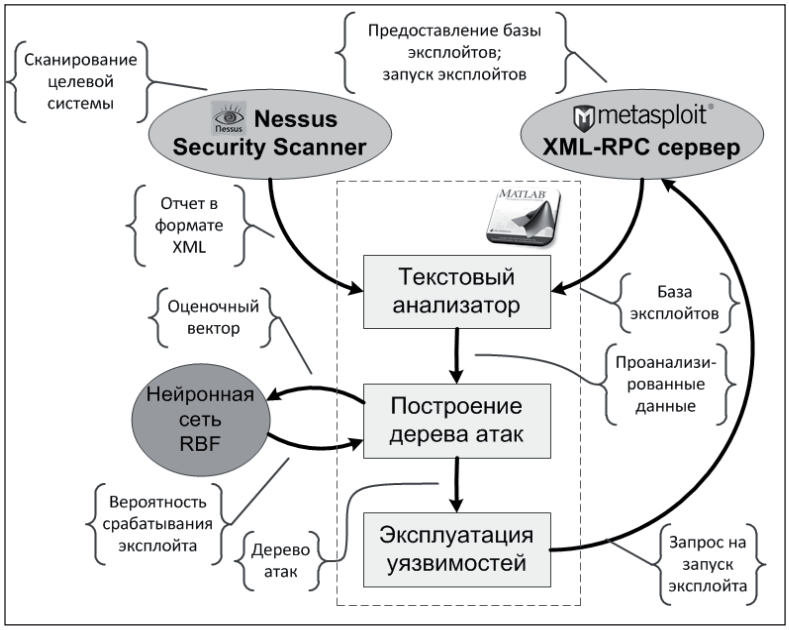

Рис. 2. Архитектура разработанной системы

Nessus Security Scanner

Nessus применяется для получения информации об исследуемой системе, а именно информации об уязвимостях системы, о типе и версии операционной системы, об открытых портах и активных сервисах, предоставляемых системой.

В настоящее время данная информация получается в ручном режиме и сохраняется в виде отчета сканера в формате XML, и затем этот отчет в качестве входного параметра передается программе-анализатору.

Metasploit Framework

Анализатор взаимодействует с сервером Metasploit Framework в автоматическом режиме.

Взаимодействие осуществляется посредством протокола XML-RPC. После подключения к серверу запрашивается база доступных эксплойтов, которая передается скрипту для дальнейшего анализа. Впоследствии системе Metasploit Framework отдается команда на запуск определенного эксплойта и получение результатов работы эксплойта.

Анализатор

Первым этапом работы анализатора в соответствии с предлагаемым методом является сканирование целевой системы. Сканирование производится с использованием сканера уязвимостей Nessus с целью определения типа и версии операционной системы, под управлением которой работает ПЭВМ, открытых портов и сервисов, работающих на ПЭВМ, и уязвимостей исследуемой системы. В настоящее время сканирование производится в ручном режиме, результаты импортируются в систему анализа. Однако нет теоретических ограничений на получение информации от другого сканера.

Вторым этапом является получение базы эксплойтов из Metasploit Framework. Данный этап выполняется в автоматическом режиме.

Далее следует построение дерева атаки. Алгоритм построения дерева атаки базируется на модели деревьев атак с использованием искусственной нейронной сети для решения задачи вычисления вероятностей срабатывания узлов построенного дерева и эвристическом алгоритме построения связей между узлами.

При построении дерева атаки выполняется выбор подходящих для эксплуатации целевой системы модулей Metasploit на основе анализа полученной на предыдущих этапах информации с последующей генерацией вероятностей срабатывания выбранных эксплойтов посредством нейронной сети.

На последнем этапе работы анализатора выполняются обход дерева и визуальное отображение построенного дерева с цветовой индикацией вероятности срабатывания эксплойта. Для каждого узла дерева запрос на запуск эксплойта с необходимыми параметрами отсылается серверу Metasploit Framework. Производится проверка успешности срабатывания эксплойта. В случае, если эксплойт не сработал, все дерево, лежащее ниже, отсекается. В случае срабатывания эксплойта узел помечается как успешный.

4. Результаты тестирования системы

В ходе работы была подобрана следующая размерность сети: 8 нейронов входного слоя, 50 нейронов скрытого слоя и 1 нейрон выходного слоя.

Для обучения сети используется выборка из приблизительно шестидесяти векторов, сформированная синтетически. В рамках тестирования системы для анализа эффективности ее работы были также проведены атаки на ПЭВМ с помощью инструмента автоэксплуатации, предоставляемого Metasploit Framework. Результаты проведенных экспериментов представлены в таблице 1.

Таблица 1. Результаты экспериментов

ОС	Metasploit Framework		Разрабатываемая система	
	Опробовано эксплойтов	Первый сработавший эксплойт	Опробовано эксплойтов	Первый сработавший эксплойт
Windows XP SP3	102	22	10	5
Windows XP SP2	51	22	10	5

ОС	Metasploit Framework		Разрабатываемая система	
	Опробовано эксплойтов	Первый сработавший эксплойт	Опробовано эксплойтов	Первый сработавший эксплойт
Windows 2000	51	46	10	5

Как видно из представленных результатов, предложенный метод доказал свою эффективность в сравнении со стандартным средством автоэксплуатации Metasploit Framework. ПЭВМ, работающие под управлением Windows Vista и Windows 7, проэксплуатировать не удалось. В соответствии с результатами исследований система имеет ряд преимуществ перед существующими средствами:

• анализирует и учитывает в своей работе информацию об операционной системе и сервисах исследуемой ПЭВМ;

• рассчитывает вероятности срабатывания эксплойтов и запускает наиболее «вероятные» эксплойты первыми;

• обеспечивает визуальное отображение дерева атаки.

В системе также реализована, но еще не протестирована возможность эксплуатации систем Linux.

5. Дальнейшие исследования

В рамках дальнейших исследований предполагается производить построение и обход графа атаки, что включает запуск не только активных эксплойтов, но и пассивных, а также реализацию многостадийных атак. Для более детального анализа безопасности в максимально приближенных к реальности условиях предполагается производить учет ответной реакции системы на атакующие воздействия и в соответствии с этим осуществлять коррекцию построенного графа атак.

СПИСОК ЛИТЕРАТУРЫ:

1. *Schneier B.* Attack Trees // Dr. Dobb's Journal, 1999. URL: http://www.schneier.com/paper-attacktrees-ddj-ft.html (15.03.2012).
2. *Camtepe S. A., Yener B.* A Formal Method for Attack Modeling and Detection // TR-06-01, Rensselaer Polytechnic Institute, Computer Science Department. 2006. URL: http://citeseer.ist.psu.edu/751069.html (15.03.2012).
3. *Sheyner O., Haines J., Jha S., Lippmann R., Wing J. M.* Automated Generation and Analysis of Attack Graphs // Proceedings of the IEEE Symposium on Security and Privacy. Oakland, CA, USA, 2002. P. 273–284.
4. *Jha S., Sheyner O., Wing J.* Two Formal Analyses of Attack Graphs // Proceedings of the 15th IEEE Computer Security Foundations Workshop. Nova Scotia, Canada, June 2002. P. 49–63.
5. *Sheyner O.* AttackGraph Tool 0.5. URL: http://www.cs.cmu.edu/~odobzins/scenariograph/as_files/AttackGraph-0.5.tar.gz (15.03.2012).
6. *Von Ohiemb D., Lotz V., Gollmann D., Gьnter K., Waidner M.* Formal security analysis with Interacting state machines // Lectute Neotes in Computer Science. 2002. № 2502. P. 212–228.
7. *Хайкин С.* Нейронные сети: полный курс. 2-е изд.: Пер. с англ. М.: Издательский дом «Вильямс», 2006. – 1104 с.