были ограничены объемом имеющихся данных. К примеру, многие проблемы, связанные с естественным языком и речью, плохо подходят для математически точных алгоритмических решений. Для лучшего решения этой проблемы часто используются статистические модели машинного обучения, которые требуют подготовки данных для построения и оценки [1]. Теперь точный выбор математической модели теряет свое значение, потому что есть достаточно большие данные, чтобы компенсировать их необходимость [2].

Способность эффективно обрабатывать массивные наборы данных стала неотъемлемой частью широкого круга научных и других академических дисциплин. Однако это не устраняет необходимости глубокого понимания теоретических основ проблемы. Большие данные позволяют ученым преодолевать проблемы, связанные с малыми выборками исходных данных таким образом, чтобы уменьшить влияние предположений на математическую модель. Это позволяет избежать проблемы с перестроением существующей модели для каждого нового набора входных данных, лучше обрабатывать зашумленные исходные данные.

В последние годы появилось множество новых систем для решения задач хранения и обработки больших массивов данных. В настоящее время более 220 таких систем относятся к так называемым NoSQL систем данных, и регулярно появляются новые решения (http://db-engines.com/en/). Принимая это во внимание, мы сталкиваемся с ограниченными или отсутствующими теоретическими основами для моделей данных и языков запросов, и нам не хватает четких стандартов, которые помогли бы избежать проблем при работе. Многие системы NoSQL предназначены для развертывания на компьютерах с распределенными кластерами, и предлагают широкий выбор спецификаций уровня согласованности данных, обеспечивают встроенную поддержку параллельной обработки с использованием структуры MapReduce. Однако не все проблемы с большими данными поддаются решению с применением MapReduce. Например, изменяющиеся во времени графики и динамические сети, требовательные к обработке в режиме реального времени, и масштабируемая обработка потока данных создают дополнительные проблемы. Однако при решении поставленной задачи нужно проявлять осторожность. Далеко не всегда нужно использование обработки больших объемов данных, им нужны правильные данные [4]. Часто данные могут быть противоречивыми, неполными, неточными, субъективными, избыточными, необъективными и зашумленными. Такие данные потенциально могут создать путаницу и дезинформацию, а не давать полезные сведения. Как описано в «Big Data or Right Data?» [4], нам необходимо задать правильный вопрос:

1) Как мы обрабатываем, фильтруем и моделируем исходные данные, чтобы получить нужные?
2) Как мы определяем достоверность таких данных?
3) Как мы различаем достоверные данные и ошибочные? Фильтрация ошибочных данных является нетривиальной проблемой и возможный источник проблем.
4) Как мы определяем и устраняем дубликаты?

Конфиденциальность данных также актуальна, поскольку она касается юридических и этических ограничений. О каких вопросах конфиденциальности следует позаботиться? Нужно ли анонимизировать данные? Это связано с тем, что отслеживание происхождения данных является основным требованием во многих крупных приложениях: информация о происхождении используется для преобразования данных, позволяет проводить аудит, моделировать подлинность, осуществлять контроль доступа для производных данных и оценивать качество и доверие к данным. Около 400 лет назад Галилей сказал, что «книга природы написана на языке математики». Сегодня это становится еще актуальнее, учитывая потенциал больших данных для инновационных открытий в науке и аналитике, основанной на данных. Большие данные могут стать следующим рубежом инноваций, конкуренции и производительности.

Списокиспользованныхисточников

1. V. Gudivada, D. Rao, and V. Raghavan, “Big Data–Driven Natural Language– Processing Research and Applications,” Big Data Analytics, V. Govindaraju, V. Raghavan, and C.R. Rao, eds., Elsevier, 2015 (in press).
2. Halevy, P. Norvig, and F. Pereira, “The Unreasonable Effectiveness of Data,” IEEE Intelligent Systems, vol. 24, no. 2, 2009, pp. 8–12.
3. V. Gudivada, D. Rao, and V. Raghavan, “Renaissance in Data Management Systems: SQL, NoSQL, and NewSQL,” Computer (in press).
4. R. Baeza-Yates. “Big Data or Right Data?” Proc. 7th Alberto Mendelzon Int’l Workshop on Foundations of Data Management (AMW 13), 2013, vol. 1087, paper 14; http://ceur-ws.org/Vol-1087/paper14.pdf.

ИССЛЕДОВАНИЕ И АНАЛИЗ ЗАЩИЩЕННОСТИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА ОСНОВЕ ПОСТРОЕНИЯ ТЕСТА НА ПРОНИКНОВЕНИЕ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Петухов А. В.

Медведев С. А. – канд. техн. наук, доцент

Огромный рост количества мобильных устройств за последнее время, открыл новые возможности

хранения и передачи пользовательской информации, но, с другой стороны, спровоцировал увеличение угроз безопасности и конфиденциальности пользовательских данных. Тестирование на проникновение широко используется для оценки безопасности компьютерных систем или сетей средствами моделирования атаки злоумышленника.

Задача теста на проникновение сводится к активному моделированию и анализу системы на наличие потенциальных уязвимостей, которые могут спровоцировать некорректную работу целевой системы, либо полный отказ в обслуживании. Можно выделить несколько типов теста на проникновение:

1) Pen-Test Виртуальных сетей,
2) Pen-Test Приложений,
3) Pen-Test Интернет сайтов,
4) Физический Pen-Test,
5) Pen-Test Серверной части,
6) Социально-сетевой Pen-Test.

Безопасность мобильной операционной системы iOS состоит из Secure Boot Chain, System Software Authorization, Secure Enclave и Touch ID [1]. Хотя Apple встроила эти меры в операционную систему iOS, есть некоторые дополнительные обстоятельства, которые следует учитывать. Что происходит, когда безопасность системы iOS обходится путем использования Jailbreak tool? Как это влияет на устройство, как это влияет на приложение? Поскольку разработчики приложений полагаются на эти основные функции безопасности, которые доступны после запуска приложения, можно с уверенностью сказать, что могут возникнуть непредвиденные проблемы, если это не так. А как насчет процесса разработки мобильных приложений? Конечные пользователи не знают, что приложение должным образом проверено на предмет безопасности. Эти проблемы иллюстрируют необходимость и создают предпосылки для оценки безопасности мобильных приложений.

Для эффективного анализа необходимо установить соединение между рабочей станцией и анализируемым устройством. Это можно сделать с помощью стандартного интерфейса USB или через сеть. Поскольку iOS закрытая операционная система, ограничения безопасности затрудняют дальнейший анализ, необходимо выполнить процедуру взлома "Jailbreak". Основной операционной системой, от которой наследуется iOS, является Mac OS X, уходящая корнями в BSD Unix, однако реализация в iOS урезана, и многие из распространенных утилит не существуют. Первое что потребуется, это SSH, чтобы упростить удаленный вход в сеть WiFi, используя утилиту OpenSSH из репозитория взломанных приложений Cydia.[2] Далее подключаемся в систему через SSH.

Необходимо выделить более узкую область поиска, например, поиск незащищенной аутентификации, недостатки памяти, недостатки шифрования, связь клиент-сервер, тип передаваемых данных и т. д, чтобы получить большее представление о внутренней работе приложений и потенциальном риске (рис 2).

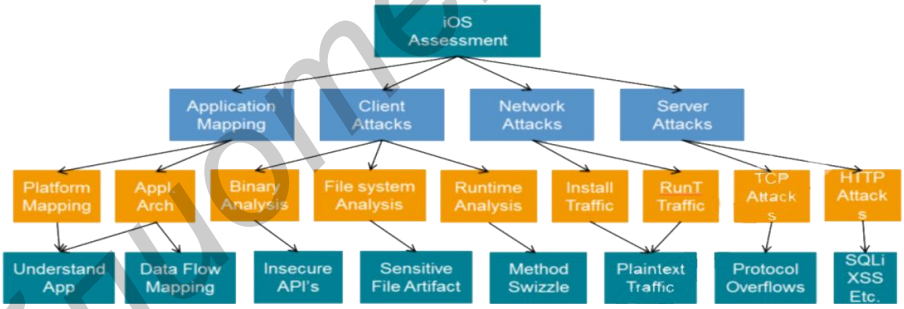

Рис 2- Security testing flow

Поскольку все приложения, доступные в Apple Store, зашифрованы, необходимо воспользоваться дешифратором приложений iOS. В результате выхода класса дампа можно фильтровать для поиска ключевых слов, таких как «login», «authentication», «username» или «password». Все, что может идентифицировать метод, который мы можем использовать для проверки подлинности. Просматривая методы, существующие в бинарном приложении, можно определить, какие из них могут быть нацелены на определенную функциональность (например, проверку подлинности) [3].

Дляанализ использования сети мобильного приложения, важно найти к какому хосту подключается приложение, какие протоколы используются при передачи данных, для последующих прокси атак. С помощью поддельного сертификата, перенаправляется весь исходящий и входящий трафик на наш прокси сервер. Для более полного выделения пакетов из сетевого запроса применялись снифферы.

Поиск базы данных приложения, как правило, осуществляется по двум расширениям: .db или .sqlite, таким образом можно посмотреть хранящуюся информацию в локальной базе данных приложения.

Следует отметить несколько недостатков данного метода:

- Поскольку архитектура каждого приложений разная, необходимо подбирать оптимальную комбинацию поиска уязвимых секторов;

- Из-за закрытости мобильной операционной системы iOS, некоторые версии не поддаются root - взлому, что усложняет или делает невозможным дальнейшее прохождение теста.

Разработан и применен метод использования теста на проникновение для приложений в мобильной операционной системы iOS. Проведена оценка эффективности метода, а также рассмотрены преимущества и

недостатки данного метода.

Список использованных источников:

1. Apple. iOS Security. 2017, February 01, 10c http://images.apple.com/iphone/business/docs/iOS_Security_Feb14.pdf
2. Dave Shackleford “A Penetration Testing Maturity and Scoring Model” RSA Security Conference 2014 24-26c
3. Mark Rasch “Legal Issues in Penetration Testing” November 26, 2013, 67c

АЛГОРИТМ ПОИСКА НАИЛУЧШЕЙ КОНТР-СТРАТЕГИИ В ИГРАХ С НЕПОЛНОЙ ИНФОРМАЦИЕЙ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Пилипчук В.О.

Бахтизин В.В. – профессор, канд. техн. наук, доцент

В докладе приведён алгоритм поиска стратегии, максимизирующей выгоду, получаемую участником в играх с неполной информацией. Предложена модификация данного алгоритма, позволяющая эффективнее использовать память, необходимую для хранения дерева решений.

Одним из оценочных критериев алгоритмов искусственного интеллекта является поведение системы в наихудшем для неё сценарии. Наилучшая контр-стратегия может быть рассчитана через рекурсивный обход дерева решений, проходя по каждому состоянию игры. Иллюстрации к алгоритму [1] приведены на рис. 1.

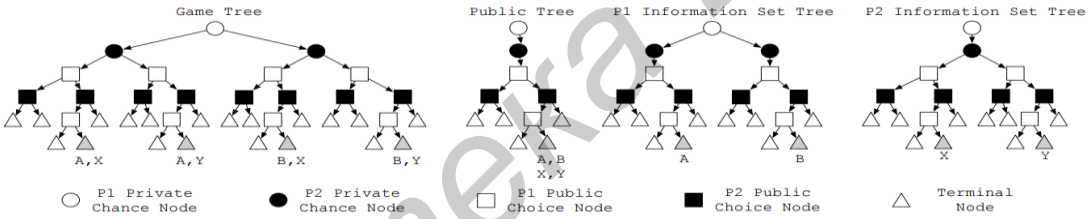

Рис. 1 – Деревья решений псевдо-игры

Игра начинается в главном узле (Game Tree), где белые круглые узлы представляют события, определяющие приватную информацию участника 1, которые и делают данную игру – игрой с неполной информацией. Дочерние узлы определяют приватную информацию участника 2. При обходе дерева сверху вниз белые и чёрные квадратные узлы представляют узлы принятия решений для участника 1 и 2 соответственно. Треугольные узлы являются конечными, они определяют результат действий, которые были приняты на пути к ним.

Поскольку в данной игре существует приватная информация, которая не известна противной стороне, каждый игрок имеет разное представление данной игры – на рис. 1 это представление участника 1 (P1 InformationSetTree) и участника 2 (P2 InformationSetTree). Приватная информация не известна противнику, поэтому белые и чёрные круглые узлы имееют по одному дочернему узлу. Каждый узел в этом дереве представляет собой набор состояний игры, которые игрок не способен различить (Informationset).

Рассмотрим расчёт наилучшей контр-стратегии для игрока 2. Выполним рекурсивный обход его приватного дерева решений (Information set tree). В конечном узле 'X' необходимо рассмотреть все состояния игры, в которых она может находиться: 'A,X' или 'B,X'. Необходимо знать вероятности достижения узлов 'A' и 'B' противником в его приватном дереве решений, согласно его принятым решениям и случайным событиям. Учитывая данные вероятности, можно рассчитать ненормализованное значение математического ожидания выгоды, которое получит игрок 2, достигнув узла 'X', как сумму произведений выгоды каждого неразличимого конечного состояния на вероятность достижения данного состояния противником. Будем возвращать это значение при обходе дерева решений. Рекурсивно возвращаясь в узлы принятия решений игроком 2, будем выбирать решение с наибольшей выгодой и возвращать ожидаемую выгоду этого решения. В узлах случайных событий и узлах принятия решений противником будем возвращать сумму значений дочерних узлов. Значение, полученное в результате обхода дерева решений, будет являться максимальной выгодой против фиксированной стратегии противника.

Предлагается использовать открытое дерево решений (PublicTree), которое описывает решения, принимаемые участниками игры. При обходе данного дерева в каждом конечном узле необходимо рассмотреть все возможные приватные параметры, с которыми противник может оказаться в данном узле, и повторить расчёты, описанные в оригинальном алгоритме. Данная модификация позволяет использовать значительно меньше памяти для игр с большим количеством узлов принятия решений при прочих равных