МОДЕЛИ КРИПТОСТОЙКИХ ХЕШ-ТАБЛИЦ

А.Д. Трофимов, А. Б. Гуринович
Кафедра вычислительных методов и программирования,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: a.d.trofimov@gmail.com

В современном мире высоких технологий все больше внимания уделяется данным, способам их чтения и хранения, а также способам защитить данные. Обеспечение безопасности хранения данных на данный момент является одной из важнейщих задач, стоящих перед всеми государственными структурами и частными предприятиями. И решением данной задачи на одном из уровней, является использование хеш-таблиц для хранения захешированных данных в них.

Введение

Хеш-таболицы - это структура данных, которая предназначена специально для хранения пар ключ-значение. В такой структуре данных возможны три действия: добавление пары ключ-значение, поиск и удаление пары по ключу. Процесс хеширования преобразует входную строку в выходную битовую строку определенной длины. Криптографические хеши должны отвечать нескольким параметрам: они должны быть стойки к коллизиям, т.е. для строки s должно быть вычислительно невычислимо подобрать строку t, хеши которых были бы равны: H(s) = H(t); они должны быть необратимы: для каждого H(s) = x должно быть невозможно найти s зная x; они должны быть стойки к коллизиям второго рода: должно быть вычислитьно невозможно подобрать парные значения s и S, имеющие одинаковы хеш.

I. Недостатки существующих решений

К сожалению, в настоящее время не всегда достаточно хранить данные просто захешировав их. Существует несколько способов взлома и получения данных из хеш-таблиц. Начиная от обратного хеширования не стойких хеш-функций и заканчивая подбором хешей с использованием радужных таблиц. Также одной из слабостей простого хеширования является сравнение хешей строк и поиска одинаковых хешей(а значит и одинаковых зашифрованных строк). Второй недостаток существующих решений — это хранение всей необходимой информации для дешифровки хешированных данных в самой Базе Данных.

Стандарты и рекомендации образуют понятийный базис, на котором строятся все работы по обеспечению информационной безопасности. В то же время этот базис ориентирован, в первую очередь, на производителей и "оценщиков"систем и в гораздо меньшей степени - на потребителей. Стандарты и рекомендации статичны, причем статичны, по крайней мере, в двух аспектах. Во-первых, они не учитывают постоянной перестройки защищаемых систем и их окружения. Во-вторых, они не содержат практических рекомендаций по формированию режима безопасности.

Информационную безопасность нельзя купить, ее приходится каждодневно поддерживать, взаимодействуя при этом не только и не столько с компьютерами, сколько с людьми.

Иными словами, стандарты и рекомендации не дают ответов на два главных и весьма актуальных с практической точки зрения вопроса: как приобретать и комплектовать информационную систему масштаба предприятия, чтобы ее можно было сделать безопасной, и как практически сформировать режим безопасности и поддерживать его в условиях постоянно меняющегося окружения и структуры самой системы? Стандарты и рекомендации являются лишь отправной точкой на длинном и сложном пути защиты информационных систем организаций. С практической точки зрения интерес представляют по возможности простые рекомендации, следование которым дает пусть не оптимальное, но достаточно хорошее решение задачи обеспечения информационной безопасности.

II. Решение поставленной задачи

Для защититы от подбора хешей было принято решение использовать современные, стойкие к подбору алгоритмы, такие как MD6, bcrypt/scrypt. Для защититься от поиска одинаковых хешей был использован “добавочный пароль”. Добавочный пароль - это случайная строка, прибавляемая к пароль до его хеширования. Для еще большего усложнения следует добавочный пароль был добавлен после пароля, а не до. В этом случае подбирающий лишился возможности подбирать отдельно части суммы H(добавочный пароль) + H(s).

Таким образом мы получаем хэш, увеличенной длинны с конкатенацией пароля с добавочным паролем. В таком случае генерацией радужных таблиц, конкретно для этого добавочного пароля, никто заниматься не будет.Достаточно ограничиться конкатенацией добавочного пароля с паролем и результат гифровать в md5 — ре-

Библиотека БГУИР

зультат взломать, даже зная соль, практически невозможно.

Усилить метод метод добавочного пароля можно добавив динамику в генерацию соли т.е. при правильном вводе пользователем его пароля, динамически заменять соль на новую, при каждом 10-ом логине или каждую неделю. Также если соль сделать из 8-ми абсолютно случаных символов, то при взломе такого хэша, система взлома может распознать его как SHA-хэш и априорно начнёт двигаться в ложном направлении.

Для решения проблемы когда вся необходимая информация для дешифровки хешированных данных хранилась в самой Базе Данных, был использован другой крайне действенный способ защитить данные — использовать локальный параметр. Суть этого метода заключается в дописывании одинакового второго добавочного пароля ко всем хешам, хешированным по тем же принципам что и первый, за исключением того, что второй добавочный пароль хранится за пределами БД и попадает в нее только из настроек.

Так как хеш-функция, как правило, вычисляется последовательно по строке (требования поточности алгоритма), то злоумышленнику при переборе хешей с дополнительным параметром, будет проще, когда подхешовое выражение начинается с этого параметра, т.к. злоумышленник может предвычислить заранее хеш(дополнительный параметр) и далее считать хеш(дополнительный параметр)+хеш(пароль) быстрее(практически с той же скоростью, что и просто хеш(пароль)) для всех паролей, которые будут перебираться.

Это по сути второй локальный параметр, который дописывается ко всем конструкциям пароль + локальный параметр ,и является одинаковой для всех хешей в базе. Смысл в том, что локального параметра в базе данных нет. Это константа системы, которая хранится в памяти приложения, куда она попадает из конфигурации любым способом, только не из базы данных.

Третий прием, использованный для решения проблемы защиты хеш-таблиц от взлабмывания - изспользование шума, т.е. столько фиктивных значений в таблицу хэшей, сколько сможет поддержать оборудование.

От отсутствия связи между хэшами и пользователями и от заполнения таблицы хэшей шумом мы получаем следующие приемущества: во-первых, фиктивные хэши нельзя отличить от настоящих, то есть злоумышленникам придётся проверять всю таблицу на каждый сгенерированный пароль. Так что кроме мощности процессора им требуется и большая оперативная память: если таблица не влезает в память, то это значительно уменьшит их скорость. Это также создаст проблемы при использовании GPU для перебора. Во-вторых, чтобы иметь шансы взломать переборм пароль хотя бы одного пользователя, нужна большая часть таблицы хэшей. Добполнительной мерой безопасности будет сделать таблицу настолько большой, чтобы попытка скопировать существенную её часть была легко заметна.

Таким образом, полученный алгоритм хеширования (см.рис. 1) решает проблему безопасности хранения данных в хеш-таблице, предоставляя три дополнительные меры ее усиления.

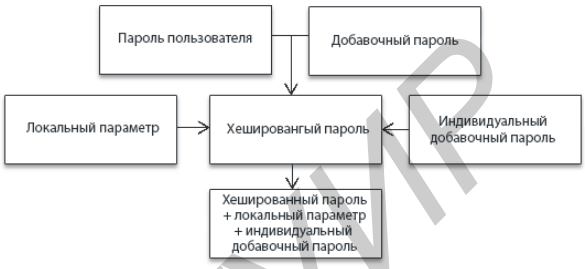

Рис. 1 – Алгоритм сохранения данных в хеш-таблице

При применении выше описанных методик стоит помнить что при переходе на другой тип хеширования, помимо трудозатрат, часто встает вопрос производительности. Действительно, более стойкие алгоритмы потребляют больше ресурсов. Следует не пренебрегать тестированием перед внедрением для нагрузок по скорости аутентификации пользователей в секунду, хотя и чаще всего переход окажется безболезненным. Выбор конкретного идеального типа хеша в конкретной ситуации может сильно разнится.

III. Выводы

Таким образом, используя выбранную мной в ходе работы комбинацию методов хеширования данных, была получена криптостойкая База Данных, стостоящая из хеш-таблиц. Разработанная База Данных отвечает всем параметрам безопасности. При реализации были выявлены и отлажены возможные проблемы скорости чтения и хеширования, а конечные алгоритмы хеширования были протестированы на стойкость. После введения в систему комбинации методов хеширования, было выявлено, что никаких дополнительных физических или программных ресурсов система не потребует. Полученный продукт для хранения данных может быть использован на предприятиях, системы которых должны отвечать параметрам безопасности хранения и использования данных.

1. Cisco blog[Электронный ресурс] – Режим доступа: http://blogs.cisco.com/ – Дата доступа: 21.01.2014
2. Хабрахабр [Электронный ресурс] – Режим доступа: http://habrahabr.ru – Дата доступа: 21.01.2014
3. Small Buiseness Data [Электронный ресурс] – Режим доступа: http://www.smallbizlabs.com/ – Дата доступа: 10.02.2014

Библиотека БГУИР