Математические структуры и моделирование
2007, вып. 17, с. 121–125

УДК 681.322

ПРОВЕРКА ПАРОЛЕЙ НА НАДЕЖНОСТЬ С ПОМОЩЬЮ МОДИФИЦИРОВАННОГО ФИЛЬТРА БЛУМА

Д.Н. Лавров, Е.А. Первушин

В статье представлены алгоритмы проверки паролей на надежность, а также генерации произносимых и легконабираемых на клавиатуре паролей. При реализации алгоритмов использовался модифицированный алгоритм Блума.

1. Введение

Обычно проблема удостоверения подлинности пользователя решается с помощью паролей. Слабость этого метода заключается в том, что ненадежный пароль может быть легко угадан. Процедура угадывания состоит в переборе по словарю наиболее вероятных паролей.

В основном пароль – это компромисс между надежностью и удобством для пользователя. Если есть возможность применить программу, хранящую пароли в шифрованном виде, то, воспользовавшись генератором случайных паролей, можно устанавливать длинные надежные пароли. Но в противном случае, когда пользователь вручную вводит пароль, на выбор влияет запоминаемость: пользователь ставит пароль, который сможет запомнить и быстро набрать. При установке пароля рекомендуется осуществить проверку на надежность: первая проверка — есть ли такой пароль в словаре.

Пользователь может применить генератор, который по некоторым правилам создает произносимые пароли (произносимые пароли легче запоминаются). Вскрытие пароля может вестись по словарю, каждое слово которого удовлетворяет этим правилам. Можно вести перебор в порядке

Омский государственный университет.
E-mail: lavrov@omsu.ru

убывания вероятности их генерации, если генерируемые пароли неравномерно распределены. Здесь встает вопрос о размере пространства генерируемых паролей. Если размер относительно мал, то пароль, полученный генератором, следует считать ненадежным. В этом случае в проверку на надежность следует включить тест на соответствие пароля правилам, по которым генератор его построил.

Прямой поиск в словаре -– неэффективная по времени и занимаемому месту операция. Спаффорд предложил использовать фильтр Блума [1] для отбраковки паролей из словаря [2]. Фильтр гарантированно отбрасывает пароли из словаря, но может отбросить и пароли, не включенные в словарь. Фильтр состоит из фиксированного количества хэш-функций и битового массива, изначально заполненного нулями. Добавление пароля состоит в следующем: каждая хэш-функция переводит слово в индекс в массиве, далее биты по этим индексам устанавливаются в единицу. При запросе таким же образом вычисляются индексы. Проверяются биты, находящиеся в таблице по этим индексам. Если среди них хотя бы один ноль, то это слово не было добавлено, иначе пароль отбрасывается, но есть вероятность отбросить ранее не добавленное слово. Эта вероятность зависит от размера массива, количества функций и количества добавленных слов и может быть сколь угодно малой. Скорость запроса не зависит от количества добавленных слов.

Модифицируем фильтр Блума следующим образом: алгоритм добавления:

Введем параметры: $n > 0$, $0 < s \leq n$ – целые числа, например $n = 3$, $s = 1$. При добавлении слова разобьем его на n-граммы – слова длины n. Каждая следующая n-грамма отстоит от предыдущей на s символов. Пусть L – длина слова. Тогда количество n-грамм равно

$$\left[\frac{(L-n)}{s}\right] + 1,$$

где округление происходит до большего целого. Возможно, последняя n-грамма будет содержать меньше, чем n, символов. Далее полученные n-граммы заносятся в фильтр.

2. Алгоритм проверки

Для проверки слова на принадлежность словарю требуется проверить все n-граммы, полученные из проверяемого слова. Как и прежде, возможны коллизии, то есть отбрасывание ранее не добавленных n-грамм. Если все n-граммы отброшены, то слово будет отброшено. Если хотя бы одна из n-грамм принята, то такое слово не добавлялось в фильтр.

Рассматривая n-граммы как правила, можно так же генерировать произносимые пароли. Пусть ячейка в таблице представляет целое число. При добавлении n-граммы, как и раньше, вычисляются значения хэш-функций, которые являются индексами в таблице. Значения чисел, находящихся в таблице по этим индексам, увеличиваются на единицу. Чтобы посчитать, сколько раз была добавлена n-грамма, необходимо вычислить хэш-функции от этой n-граммы и найти минимум среди чисел, находящихся в таблице по этим индексам (при условии, что удаления из таблицы не происходят, то есть хранимые числа не уменьшаются). Результат, возможно, не будет верным из-за коллизий или переполнения ячеек. Далее таким же образом добавляется эта n-грамма, укороченная до $n - s$ символов. Таким образом, после добавления словаря для последовательности $n - s$ символов $a_1 \dots a_{n-s}$ и s символов $b_1 \dots b_s$ можно вычислить вероятность того, что после $a_1 \dots a_{n-s}$ следует $b_1..b_s$.

Для хранения частот можно было использовать другую структуру данных, например n-мерную матрицу, количество элементов которой равно длине алфавита в степени n. Но преимущество предложенного метода в том, что он не зависит от алфавита, а скорость запроса по-прежнему не зависит от количества добавленных слов.

Пусть N – длина массива, K – количество хэш-функций, после добавления D элементов вероятность того, что K случайным образом выбранных ячеек не пусты, равна

$$P = \left(1 - \left(1 - \frac{1}{N}\right)^{KD}\right)^K.$$

При фиксированных D и N значение P минимизируется при $K = ln2 \cdot N/D$ [3]. Вероятность, что хотя бы одна ячейка из K случайно выбранных ячеек пуста, равна $1 - P$. Пусть из всего множества n-грамм G добавлено G_1. Обозначим $G_2 = G/G_1$. При добавлении новой n-граммы ее значение будет верным, если хотя бы одна из K ячеек, соответствующих этой n-грамме, хранит ноль – это произойдет в $(1 - P) * G_1$ случаях из G_1. При запросе n-грамм из G_2 неверный результат будет возвращен в $G_2 \cdot P$ случаях, верный - $G_2 - P \cdot G_2$. Таким образом, доля неправильных значений при запросе из G равна $1 - \frac{(1-P)G_1+G_2-P \cdot G_2}{G_1+G_2} = P$. В формулу $P = (1 - (1 - 1/N)^{KD})^K \approx (1 - e^{-KD/N})^K$ подставим $K = \ln 2 \cdot N/D$. Получим:

$$(1/2)\ln 2 \cdot N/D = P \ln(1/2)\ln 2 \cdot N/D = \ln PD/N = -\ln 22/\ln P,$$

при $P = 0,01 = 1\% D/N \approx 0,1043$. То есть хранение 10% от количества

всех n-грамм с учетом того, что частоты 99 процентов n-грамм будут сохранены верно, займет не больше памяти, чем количество всех n-грамм.

3. Генерация паролей

Чтобы выбрать случайным образом один элемент (часть слова длины n или $n-s$) из заданного множества $w_1, \cdot, w_d$ с учетом частоты их добавления, посчитаем сумму, сколько раз был добавлен каждый из элементов. В пределах полученной суммы случайным образом выберем неотрицательное число. Повторный перебор этих слов с вычислением суммы ведется до достижения выбранного числа. Последнее слово будет выбранным. Таким образом, те элементы, которые не были добавлены, не будут выбраны.

Пусть задан алфавит, из букв которого будут генерироваться пароли (возможно, включающий конечный символ). Начало слова выбирается случайным образом из всех последовательностей длины $n-s$. Пока не будет достигнута максимальная длина пароля или не встретится конечный символ, происходит генерация новых s символов по $n-s$ последним сгенерированным символам $a_1 \dots a_{n-s}$. Для этого из множества $\{a_1..a_n - s, b_1, .., b_s | b_i - \}$ выбирается один элемент, последние s символов которого станут продолжением генерируемого слова.

4. Генерация паролей с фильтрацией для удобного набора на клавиатуре

Зададим множество правил - биграмм, которые состоят из букв заданного алфавита, причем буквы каждой отдельной биграммы находятся рядом на клавиатуре либо обе буквы одинаковы. Генерация слова осуществляется по ранее описанному алгоритму, но из меньшего набора n-грамм. Под ограничения не попадают первые два символа. Остальные символы удовлетворяют следующему условию: пусть $a_1 a_2 b_1 \dots b_m$ - сгенерированный пароль, для каждого символа b_i из множества $b_1, \dots, b_m$ в множестве правил есть биграмма, второй символ которой b_i, а первый находится в слове $a_1 a_2 b_1 .. b_{i-1}$ на расстоянии от одного (т.е. эти символы рядом) до заданного числа r, например 4.

Для представления множества правил используем фильтр Блума в классическом варианте. Задавая вероятность коллизии, можно влиять на вероятность допуска правил, не включенных в исходное множество. Таким образом, не все символы сгенерированного слова будут удовлетворять правилам. Это расширяет пространство генерируемых паролей.

Литература

1. Bloom B.H. Space/time trade-offs in hash coding with allowable errors // Communications of the ACM. N.13(7). P.422-426. July 1970.
2. Spafford E.H. Opus: Preventing weak password choices // Computer and Security. N.11 P.273-278. May 1992.
3. Broder A., Mitzenmacher M. Network applications of Bloom filters: A survey // In Proc. of the 40th Annual Allerton Conference on Communication, Control, and Computing. P.636-646. 2002.