Исследование алгоритмов хеширования, используемых для аутентификации в Web-приложениях♣

Л.С. Адрова, П.Н. Полежаев

Оренбургский государственный университет
adrova.lyubov@yandex.ru, polezhaevw@mail.ru

Аннотация

Настоящая статья посвящена исследованию алгоритмов хеширования, используемых для аутентификации в Web-приложениях. Анализ времени работы алгоритмов показал, что лучшим вариантом является алгоритм SCRYPT с параметрами (2^7, 2^8) и (2^8, 2^7). Для алгоритмов GOST и SHA512 подобрано оптимальное количество внешних раундов, приводящее к замедлению их работы без значительного сужения выходного множества значений.

1. Проблема хранения пароля пользователя в Web-приложении

В настоящее время широкое распространение получили различные интернет-сервисы, поэтому пользователи стали чаще задумываться о безопасности при работе в глобальной сети. Основной способ аутентификации в Web-приложении — использование логина и пароля, которые обычно хранятся в некоторой базе данных приложения. Зачастую они хранятся в открытом виде (даже на очень популярных сайтах), что облегчает работу злоумышленников по их получению с помощью SQL-инъекций и последующей аутентификации на сайте. Примером систем управления содержимым (Content Management System, CMS), хранящих пароль в открытом виде, служат Aqua CMS, jasP и Liferay.

Один из вариантов хранения пароля в базе данных приложения — использование хешированного пароля. Например, в 1С Битрикс, Drupal 6.15 и PHP-Nuke используются результаты применения алгоритма md5 к паролю. Radiant использует хеширование по схеме sha1(sha1(password)). В данном случае труднее получить исходный пароль, однако для его восстановления могут быть использованы радужные таблицы, содержащие миллиарды пар «пароль – результат хеширования». Радужные таблицы доступны в интернете и составлены для всех словарных паролей. Радужные таблицы могут достигать гигантских размеров, но вычисление исходного пароля будет зависеть только от времени поиска по ним.

Ярким примером служит утечка 6,5 миллионов паролей из баз данных социальной сети LinkedIn. Алгоритм хеширования паролей SHA-1, используемый без соли позволил злоумышленникам за шесть дней получить доступ к 90% паролей социальной сети [5].

Кроме того, в последние годы возросла вычислительная мощность GPU, они способны вычислять миллиарды хешей в секунду. Поэтому в настоящее время даже длинные пароли не могут считаться безопасными.

Для решения данной проблемы необходимо использовать соль, причем уникальную для каждого пользователя. Использование общей соли для всех пользователей сайта позволяет сгенерировать единую радужную таблицу, а затем ее применить к записям всех пользователей для получения паролей.

Неэффективность данного приема можно увидеть на примере хеширования паролей в CMS eFront 3.6.4, которая добавляет к алгоритму хеширования md5 соль, единую для всех пользователей: md5(password.'cDWQR#$Rcxsc').

В случае уникальной соли — генерация паролей бессмысленна, в этом случае единственный способ — брутфорс (перебор паролей по словарю или полный перебор символов из набора) с последующим их хешированием с уникальной солью и сравнением с украденным хеш-значением. Примером такого использование соли может служить алгоритм хеширования Joomla — md5(password.salt).

Чтобы предотвратить взлом паролей нужно использовать медленные хеш-функции или увеличить число раундов для быстрых хеш-функций. Одним из примеров использования увеличения числа раундов хеш-функции может служить алгоритм хеширования CMS IPB 1.3-2.17 и MyBB 1.2.x: md5(md5(salt). md5(password)).

Целью настоящей работы является исследование применительно к аутентификации в Web-приложениях различных алгоритмов хеширования, разработка рекомендаций по их использованию.

Интернет и современное общество: сборник научных статей XVI Всероссийской объединенной конференции IMS-2013, Санкт-Петербург, 9 - 11 октября 2013 г.

Таблица 1. Время работы алгоритмов хеширования BCRYPT и SCRYPT

	SCRYPT										BCRYPT
Стоимость / Размер блока	**2^1**	**2^2**	**2^3**	**2^4**	**2^5**	**2^6**	**2^7**	**2^8**	**2^9**	**2^10**	-
2^1	0,158	0,308	0,607	1,377	2,478	4,485	8,882	17,432	33,758	68,569	-
2^2	0,194	0,353	0,671	1,355	2,601	5,132	10,265	20,543	40,961	83,178	-
2^3	0,258	0,472	0,902	1,820	3,506	6,991	13,943	27,765	55,808	113,115	-
2^4	0,376	0,710	1,971	2,690	5,443	10,682	21,724	44,322	88,712	178,512	11,286
2^5	0,620	1,302	2,311	5,135	10,730	18,227	36,927	73,903	146,181	291,971	21,862
2^6	1,088	2,296	4,432	8,265	16,257	33,563	66,335	131,29	264,293	527,578	42,746
2^7	2,043	3,952	7,864	15,671	31,098	62,299	122,89	247,46	494,052	1007,48	83,959
2^8	3,975	7,743	15,283	30,413	60,269	121,36	243,50	486,65	1018,70	1936,63	167,95
2^9	7,833	15,192	30,014	59,808	119,02	239,60	479,79	967,14	1945,71	3881,92	334,61
2^10	15,290	30,092	59,581	118,02	235,98	476,14	952,70	1954,14	3824,71	7635,36	668,20
2^11	30,809	60,814	120,18	261,11	489,69	953,20	1912,1	3873,87	7691,69	15225,9	1249,4
2^12	61,815	119,527	239,77	476,62	951,58	1944,0	3772,6	7621,75	15122,9	30831,4	2677,6
2^13	124,170	269,815	561,75	983,97	2007,74	4039,9	8053,4	15865,8	31981,7	62938,3	5326,7

2. Исследование различных алгоритмов хеширования

2.1. Анализ времени работы стандартных алгоритмов хеширования

Для исследования были взяты наиболее популярные алгоритмы хеширования: MD5, SHA512, RIPEMD320, WHIRPOOL, TIGER и GOST. Часть из них входит в стандартную библиотеку .NET, другие в открытую библиотеку CryptSharp [3]. В результате исследования была определена зависимость времени их выполнения (в мс) от размера пароля (см. рисунок 1). Длина пароля варьируется от 2 до 20 символов. Время выполнения алгоритмов практически идентично, единственно из общего ряда выбивается алгоритм GOST. Время хеширования этого алгоритма достигает 0,046-0,047 мс. Но в целом время хеширования слишком мало, чтобы противостоять атаке брутфорса**.**

2.2. Анализ времени работы алгоритмов хеширования BCRYPT и SCRYPT

Большее время хеширования обеспечивают алгоритмы BCRYPT [1] и SCRYPT [2]. В результате исследования также была определена зависимость времени выполнения данных алгоритмов (в мс) от размера пароля по сравнению с выше перечисленными алгоритмами хеширования (MD5, SHA512, RIPEMD320, WHIRPOOL, TIGER, GOST) (см. рисунок 2). Это две достаточно медленные хеш-функции. Алгоритм хеширования BCRYPT использует соль для защиты от радужных таблиц. Однако BCRYPT был разработан в 1999 году, поэтому он защищен от перебора на компьютерах того времени. Сейчас появились ПЛИС, которые ускоряют выполнение BCRYPT для брутфорса.

В 2009 году был создан алгоритм SCRYPT, который работает медленнее и требует значительный объем памяти, что является дополнительным достоинством. Алгоритм SCRYPT использует память со случайным доступом, ее объём может настраиваться.

У обоих алгоритмов настраивается стоимость хеширования (количество внутренних раундов), у SCRYPT можно дополнительно изменять размер блока, что позволяет увеличивать время хеширования паролей.

Данные алгоритмы были реализованы на языке C#, исследовалась зависимость времени выполнения SCRYPT от стоимости и размера блока, BCRYPT – от размера блока. Усредненные значения времени в результате повторения замера 100 раз приведены в таблице 1.

Если принять в качестве предельного времени аутентификации 250 мс, то наиболее близкие к нему значения в 247,46 и 243,50 мс обеспечиваются алгоритмом SCRYPT при соответствующих значения (2^7, 2^8) и (2^8, 2^7) (первое число — стоимость, второе — размер блока). У BCRYPT это значение намного ниже, порядка 167,95 мс, что значительно быстрее и, следовательно, уровень безопасности хеширования понижается.

2.3. Исследование алгоритмов хеширования в зависимости от числа используемых раундов

Второй подход к борьбе с полным перебором паролей – увеличение числа раундов. В данном случае алгоритмы BCRYPT и SCRYPT не подходят, так как не рекомендуется добавление дополнительных внешних раундов. В ходе сравнительного анализа алгоритмов хеширования по критерию устойчивости к известным атакам обращения, нахождения коллизии, определения второго прообраза, выяснилось, что среди рассматриваемых алгоритмов хеширования наибольшей стойкостью обладают GOST и SHA512. Поэтому они и были выбрано для дальнейшего рассмотрения.

Рисунок 3 иллюстрирует зависимость времени работы алгоритма GOST от числа раундов, которая имеет линейный характер, рисунок 4 — аналогичную зависимость для алгоритма SHA512.

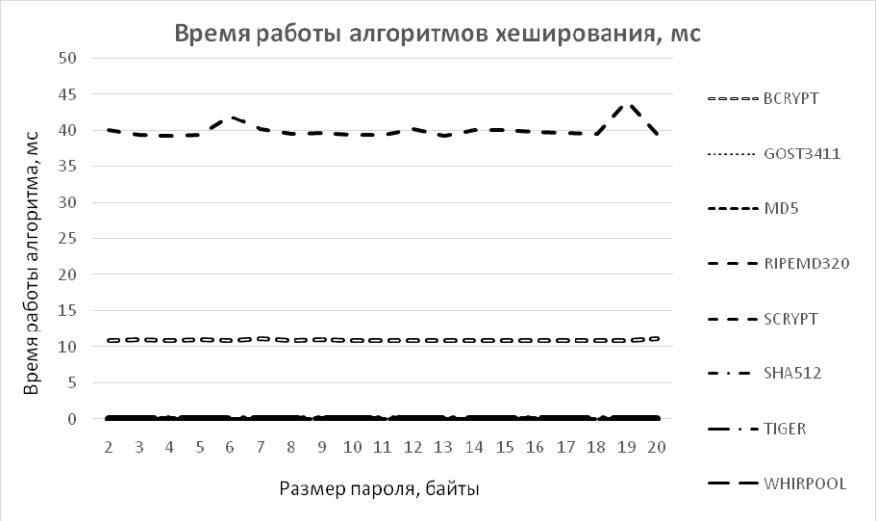

Рис. 1. График зависимости времени выполнения алгоритмов от размера пароля

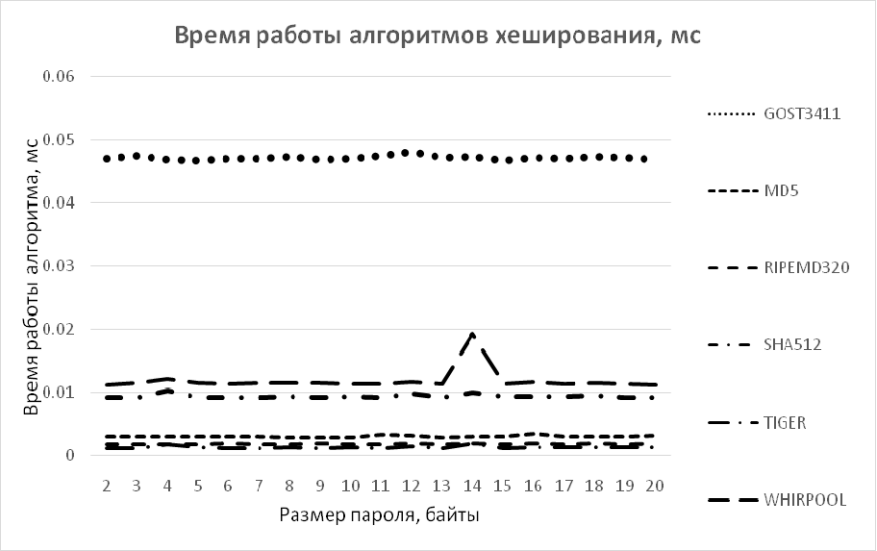

Рис. 2. График зависимости времени выполнения алгоритмов, включая BCRYPT и SCRYPT от размера пароля

Максимальное допустимое время аутентификации достигается при ~4695 раундах. Аналогично для более быстрого алгоритма SHA512 было получено значение в ~15650 раундов.

Расчеты показывают, что при максимальном числе раундов для данных алгоритмов и максимальном времени аутентификации в 250 мс перебор 100000 наиболее распространённых словарных паролей займет приблизительно 7 часов. Это лишний раз подтверждает необходимость использования сложных, близких к случайным паролей.

Полный перебор алфавитно-цифровых паролей средней длины от 6 до 8 символов займет около 880000 дней, что достаточно велико. Однако мощный суперкомпьютер с 1000 ядрами справится с этой задачей за 2.5 года.

Возникает вопрос – как влияет увеличение количества раундов на криптостойкость функции хеширования? Проведя математические расчеты [4], можно показать, что сужение выходного множества после выполнения *i* раундов может быть определено из следующей рекуррентной формулы:

$$k_i = 1 - \frac{1}{e^{k_{i-1}}}, k_0 = 0 .$$

Значение сужения может быть переведено в биты с помощью функции:

$$r(i) = log_2 \frac{1}{k_i},$$

где $r(i)$ — число, показывающее во сколько раз сократилось выходное множество (в битах). На рисунке 5 приведен график зависимости *r* до 50000 итераций.

Данный график показывает медленный рост снижения криптостойкости с ростом числа раундов.

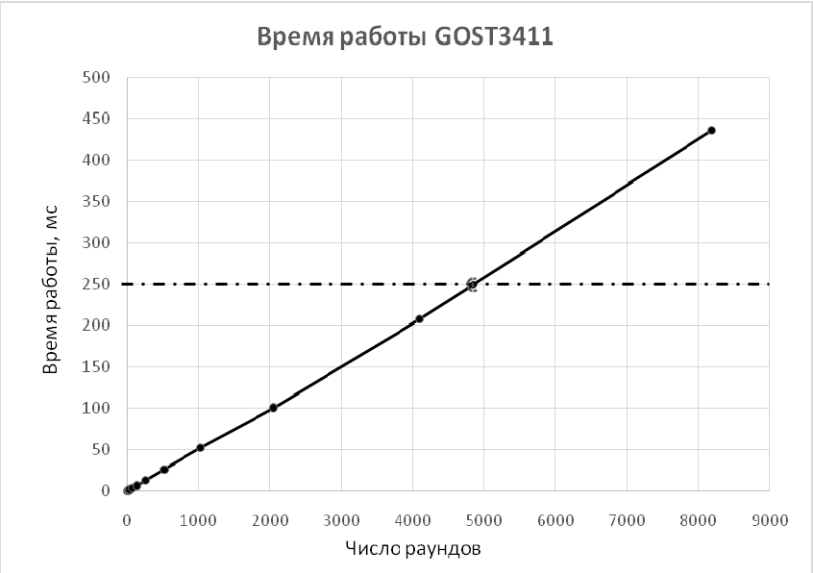

Рис. 3. Зависимость времени работы алгоритма GOST от числа раундов

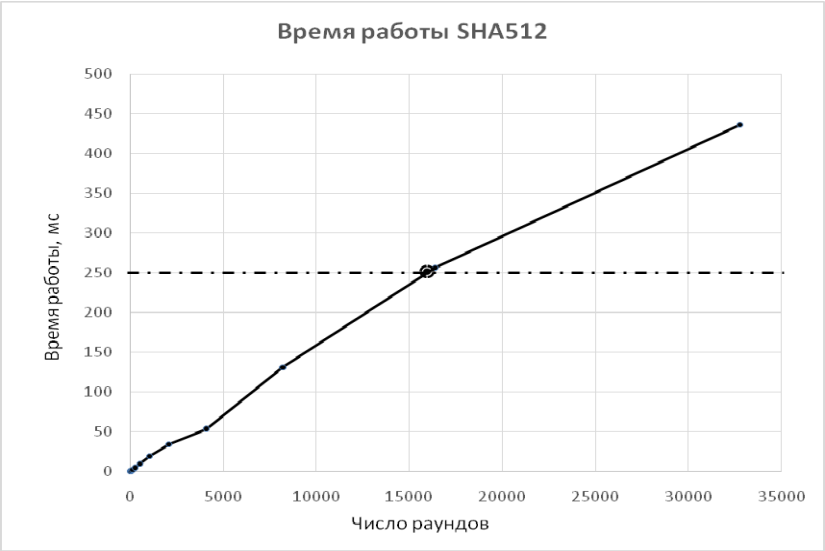

Рис. 4. Зависимость времени работы алгоритма SHA512 от числа раундов

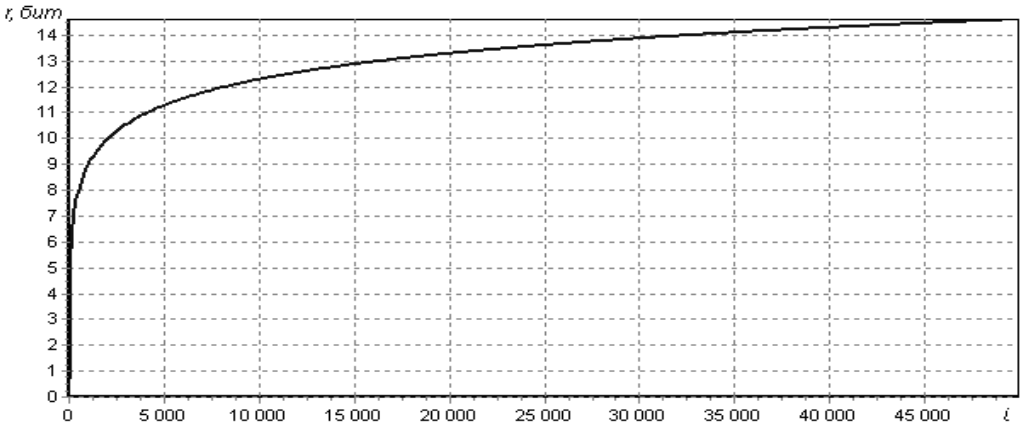

Рис. 5. График зависимости числа *r* от количества итераций

3. Результаты исследования

В результате проведенного исследования получены следующие выводы:

- важно использовать при хешировании паролей уникальную соль для защиты от радужных таблиц;
- необходимо увеличивать время хеширования, для защиты от перебора, в данном случае мы отдаем преимущество функции хеширования — SCRYPT. Данный алгоритм достаточно новый, имеет настраиваемый объем памяти для хеширования, а также это очень медленный алгоритм, что является важным параметром в хешировании. Оптимальное значение пар (раунды, размер блока) для SCRYPT: (2^7, 2^8) и (2^8, 2^7);
- оптимальное количество раундов для GOST-4695, для SHA512 — 15650, что приводит к сужению выходного множества соответственно на 12 и 13 бит, что незначительно при его размере в 256 и 512 бит соответственно;
- современные методы аутентификации по паролю защищены от полного перебора, но не от атаки по словарю. Очень важно, чтобы пользователи создавали случайные пароли длиной от 8 символов и более, так как время перебора по словарю паролей такой длины достаточно велико;
- на основе проведенного исследования, мы рекомендуем использовать SCRYPT с заданными параметрами или GOST/SHA512 с вычисленным числом раундов для большей защищенности Web-приложений от взлома паролей.

Литература

[1] Bcrypt // Википедия — свободная энциклопедия. [Электронный ресурс]. URL: http://ru.wikipedia.org/wiki/Bcrypt.

[2] Tarsnap — online backups for the truly paranoid. [Электронный ресурс]. URL: http://www.tarsnap.com/scrypt.html.

[3] Zer's Programming Page. [Электронный ресурс]. URL: http://www.zer7.com/software.php?page=cryptsharp.

[4] Криптостойкость 1000-кратного хеширования пароля. [Электронный ресурс]. URL: http://habrahabr.ru/post/100301/.

[5] Хакер Polimo опубликовал 6 458 020 парольных хешей LinkedIn. [Электронный ресурс]. URL: http://www.xakep.ru/post/58811/.

Research of hashing algorithms which are used for authentication of Web-applications

L.S. Adrova, P.N. Polezhaev

This article is devoted to the research of hashing algorithms which are used for authentication of Web-applications. Execution time analysis of algorithms has shown that the best option is SCRYPT algorithm with parameters (27, 28) and (28, 27). The optimal number of external rounds was chosen for GOST and SHA512 algorithms.

♣ Исследования выполнены при поддержке РФФИ (проекты №13-07-97046 и №12-07-31089)