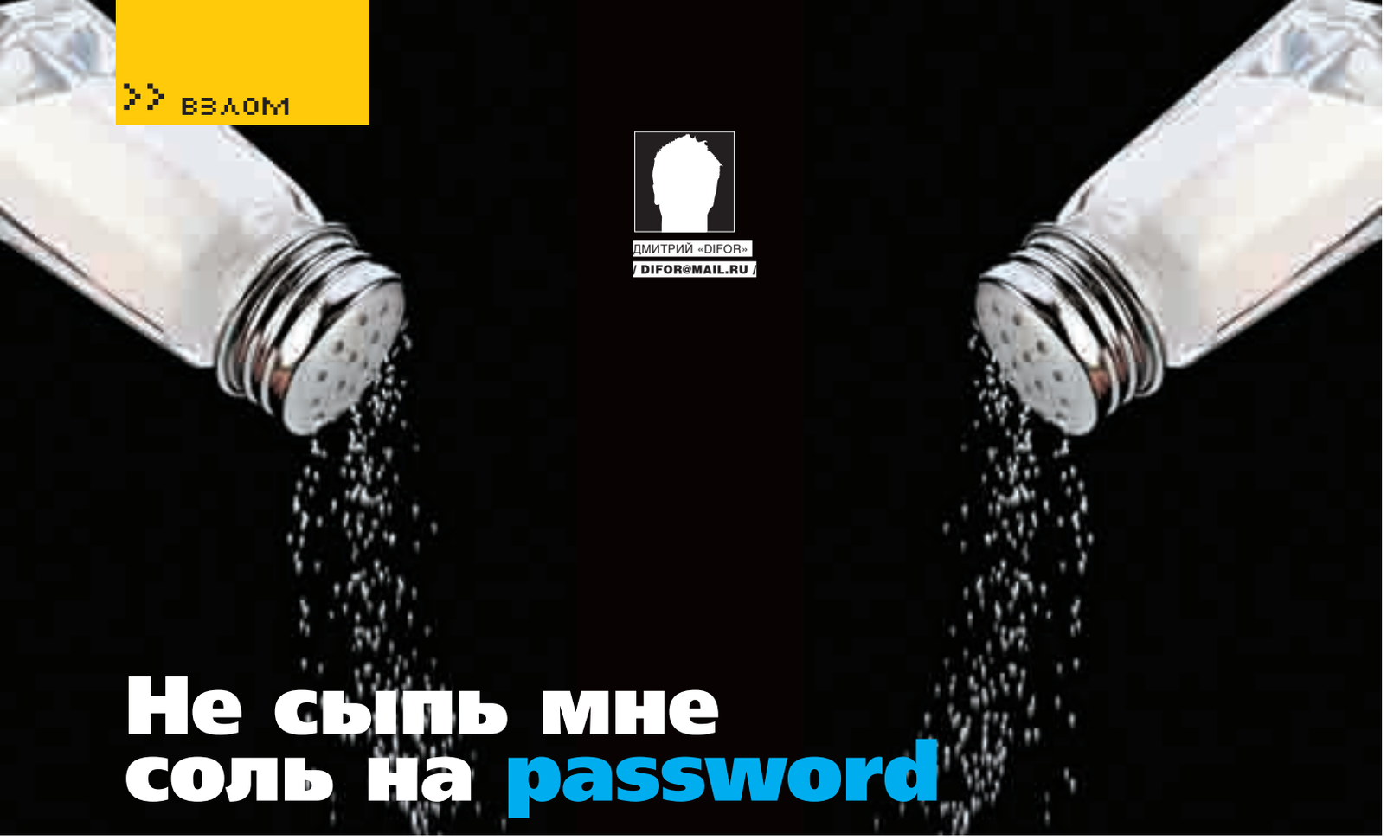


ДМИТРИЙ «DIFOR»
/ DIFOR@MAIL.RU /

Не сыпь мне соль на password

Реанимируем умерший MD5

Жил-был на свете бодрый парнишка Джон, кличка у того парнишки была Потрошитель. В Сети его знали не иначе как John The Ripper. И вот в один прекрасный день появился его хозяин и дал Джони задачу подобрать пароль к хэшу, который хозяин вытащил с помощью супермегаприватного сплоита с портала любителей морских свинок и прочей живности. Работал Джони долго-долго, пока, наконец, не понял, что с хэшем что-то не то. Обратился он к своему закадычному другу PasswordsPro. А тот ему: «Дурак ты, Джони, пасс соленый». Вот, собственно, с этого-то все и началось...

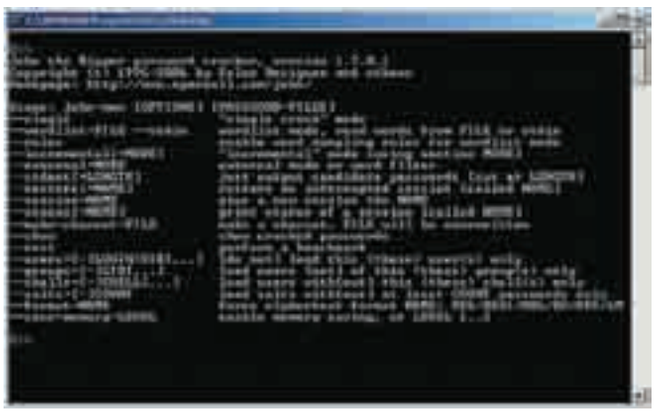
Итак, что же такое соль, хэш и как это все соединить вместе, я и попробую объяснить тебе, о уважаемый читатель. Однако для того чтобы тебе была понятна суть статьи, необходимо небольшое лирическое отступление о математическом алгоритме MD5. MD5 (Message Digest 5) — алгоритм хэширования, который был разработан профессором Л. Ривестом в 1991 году; он насчитывает 4 раунда, по 16 шагов в каждом. Предназначен он для создания отпечатков, или контрольных сумм. MD5 является односторонним алгоритмом хэширования, то есть обратной расшифровки не имеет. Выходная строка всегда имеет постоянную длину в 32 символа. Восстановление данных, зашифрованных этим алгоритмом, возможно лишь методом грубой силы, то есть брутфорсом (берется хэш от предполагаемого текста, хэш-суммы сравниваются, если они не равны, значит, текст другой).

А теперь отойдем от теории и перейдем к практике. На данный момент известна куча программ для осуществления подбора паролей: под любые платформы и с любым интерфейсом, написанные гуру и просто начинающими программистами, желающими внести свой вклад в историю. Однако, как и бывает, в историю попадают только самые-самые. Сейчас особой популярностью пользуются такие программные продукты, как PasswordsPro, MD5Inside, John the Ripper, ну и конечно, небезызвестный проект RainbowCrack, который вообще заслуживает отдельной статьи. Рассмотрим использование первой из вышеупомянутых утилит в контексте наших задач. Средняя скорость перебора по набору символов, состоящему из латинских букв нижнего регистра и цифр, на моем слабеньком компьютере с камнем 1,7 ГГц и 512 метрами мозгов на борту составляет здесь 3,3 миллиона паролей в секунду. Но люблю я эту утилиту не только за ее скорость, но и за функционал. Это перебор по словарям, по таблицам радуги, предварительный перебор по мини-словарям с дальнейшим полным брутом и т.д. Лучше один раз увидеть, чем сто раз прочитать.

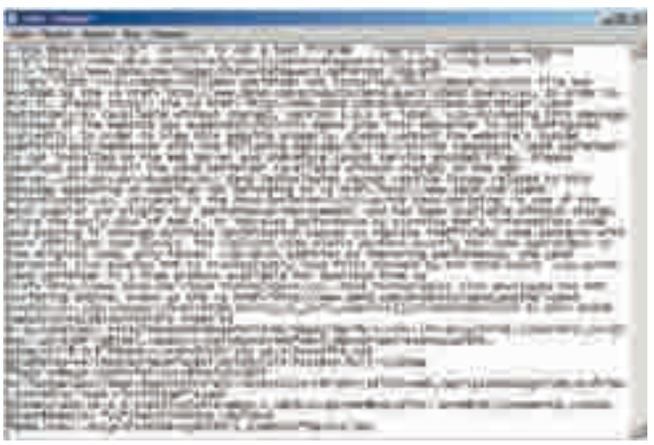
Итак, вернемся к нашим солено-маринованным паролям. Думаю, не стоит тебе рассказывать, что в алгоритме MD5, как и в его предшественнике, возможно появление коллизий (повторов), то есть твой сложный пароль длиной в 32 символа, содержащий спецсимволы, цифры, буквы разных регистров, может дать такую же хэш-сумму, как и, к примеру, пяти-, шестисимвольный простой пароль. Однако вероятность появления коллизий цифрового дайджеста MD5 критически мала. Получается что-то из разряда: если много-много обезьян посадить за печатные машинки, дать им много-много времени, то они рано или поздно напишут текст, доступный для восприятия (с появлением интернета выяснилось, что это ложь :) — примечание Forb'a). Теоретически это вполне реально, количество всех возможных сообщений, дающих цифровые дайджесты, равно 2^{256}. Однако на их поиск потребуется задействовать слишком много компьютерных ресурсов, полный перебор значений займет $1{,}5 \times 10^{62}$, а суммарный объем памяти для хранения всех дайджестов составит 2^{230}. Чтобы избежать этого, а точнее, чтобы свести шанс появления коллизии практически к нулю, разработчики программного обеспечения придумали довольно-таки интересный способ искусственного усложнения пароля — накладывание «соли».

Итак, «соль» представляет собой некий набор символов; обычно это символы обоих регистров, цифры и спецсимволы, которые накладываются или склеиваются с самим паролем или с хэш-суммой пароля.

На данный момент известны следующие способы наложения соли: md5(md5(salt).md5(pass)), md5(md5(pass).salt). Первый способ используется в форумых движках IPB версией ниже 2.0.*. По умолчанию соль и хэш там хранятся в таблице members_converge. Использование «соления» было введено для повышения безопасности системы. На мой взгляд, сильно там ничего не повысилось, просто еще один пункт в change-логе проекта. Второй способ соления применяется в форумном движке vBulletin. Там соль и хэшированные пароли по умолчанию хра-



› Рабочее окно программы John the Ripper



› Код после прохода по нему Zend'ом

нятся в таблице vb_user. Если сравнить криптостойкость обоих методов, то второй более грамотен в плане реализации. Первая попытка сделать что-то подобное была осуществлена в одном из движков для форума. Суть алгоритма состояла в вычислении двойного MD5-хэша от текста. Этот способ не является чем-то даже немного походящим на хороший крипстойкий алгоритм; перебор пароля, состоящего из букв, с помощью PasswordsPro занял пару секунд. Лично я считаю его использование в качестве основного способа просто опасным. В этой статье мы рассмотрим способ усиления первого варианта соления. Пример будет на языке программирования PHP.

```
<?php
      $text='proba';
      $salt='123!#&%asgfHTA';
      $crypted=md5(md5($salt).md5($pass));
    echo $crypted;
?>
```

Итак, что мы тут видим. Функция вычисления хэш-суммы от «соленого» пасса довольно-таки проста. Сначала получаем хэш от текста proba (c0a8e1e5e307cc5b33819b387b5f01fd), затем хэш от самой соли — от 123!#&%asgfHTA (033352797d18a1bb33e77562559b474d). Далее две хэш-суммы склеиваются в одну строку (033352797d18a1bb33e77562559b474dc0a8e1e5e307cc5b33819b387b5f01fd). После этого получаем хэш от нее (e612c1f3055ac3f9c31f52d421a3e721). Такой способ не защищает совсем уж слабые пароли. Для вскрытия их иногда даже не надо знать саму соль, лишь алгоритм накладывания. Получаем следующие действия:

1. По хэш сумме «соленого» пароля находим строку длинной 64 символа; перебор упрощает то, что используются лишь латинские символы нижнего регистра в интервале a-f и цифры.
2. Отрезаем ту часть полученной строки, которая представляет собой хэшированную соль (в нашем случае это символы с 33-64-й).
3. Скармливаем на перебор полученную строку (1-32-й символы).

Все это можно существенно облегчить, если использовать названную выше программу. Вот как с этой «проблемой» справляется PasswordsPro:

1. Скачиваем, распаковываем, запускаем саму программу.
2. В настройках выбираем нужным нам язык (в моем случае это русский)
3. После установки нужного языка и перезапуска оболочки программы топаем в пункт меню «Атака полным перебором». Мудрить мы не будем, поэтому оставляем только галочку «Набор символов - a..z».
4. Теперь пришло время кормить зверька. Добавляем новый хэш, соль и т.д.: хэш: e612c1f3055ac3f9c31f52d421a3e721; Salt (HMAC-ключ): 123!#&%asgfHTA. Тип хэша выбираем md5(md5(salt).md5(pass)) [PHP], жмем «Добавить» и начинаем перебор.

Как видишь, этот способ не является криптостойким. Попробуем усилить его в пару раз. В моем алгоритме я буду использовать метод сдвига и замены. Начнем. Первым делом стоит объявить массив спецсимволов, который будет участвовать при работе метода замены:

```
$spec=array('~','!','@'
,'#','$','%','^','&','*
','?');
```

Далее мы получим уже описанную выше хэш-сумму от стандартного метода соления (md5(md5(pass). md5(salt))):

```
$crypted=md5(md5($text).
md5($salt));
```

Объявим еще одну переменную, в которой будет храниться хэш от несоленого пароля:

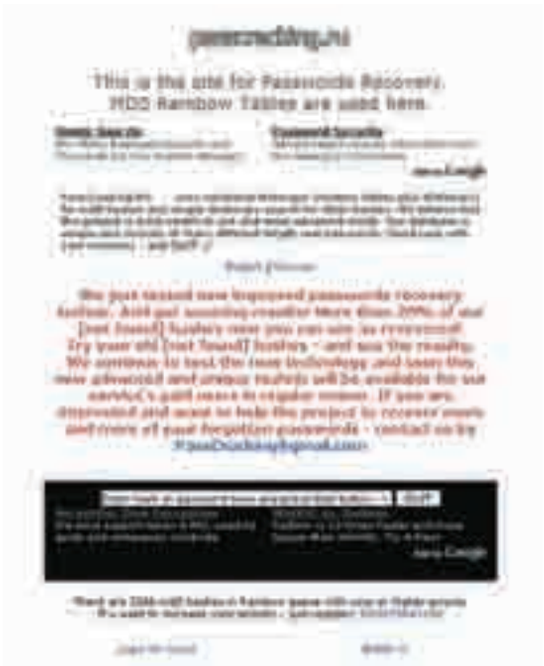
```
$c_text=md5($text);
```

Следующим шагом будет составление таблицы соответствия хэш-суммы от соления и хэш-суммы от не соления.

Таблица была построена для уяснения принципа усиления алгоритма. Алгоритм усиления будет следующим: если n'ый символ в MD5-хэше от plain-текста (нешифрованного текста) является цифрой, то в соленом хэше он поменяется на спецсимвол, номер которого в ранее объявленном массиве соответствует этой цифре. То есть второй символ в plain-строке - ноль. Следовательно, он заменит символ «6» в соленой строке символом «~». Далее, второе условие: если n'ый символ в MD5-хэше от plain-текста является буквой и попадает в диапазон a-d, то в соленой строке он переводится в верхний регистр. Ну и если

› Таблица сравнения хэшей

Plain pass	Plain pass + Salt	Changed
c	e	C
0	6	~
a	1	A
8	2	*
e	c	c
1	1	!
e	f	f
5	3	%
e	0	0
3	5	#
0	5	~
7	a	&
c	c	C
c	3	C
5	f	%
b	9	B
3	c	#
3	3	#
8	1	*
1	f	!
9	5	?
b	2	B
3	d	#
8	4	*
7	2	&
b	1	B
5	a	%
f	3	3
0	e	~
1	7	!
f	2	2
d	1	D



> Passcracking.ru — русский проект, построенный на базе таблиц радуги (Rainbow Tables)

> При знании соли и полного хэша перебор занял от силы секунды три

> Думаю, что способ, описанный в этой статье, поможет тебе вывести защиту веб-сайтов, форумов, самописных монстров и прочего креатива на новый уровень. Не бойся экспериментировать и советуйся с единомышленниками.

> http://insidepro.com — официальный сайт разработчика утилиты PasswordPro.
http://distributed.ru — портал, посвященный распределенным вычислениям, в том числе и взлому криптоалгоритмов (RainbowCrack, RC5 crack).
http://citforum.ru — крупнейший проект по компьютерной тематике. Горы материалов, начиная от описания распиновки конекторов RJ-45 и заканчивая информационной безопасностью.

> На диске к журналу ты найдешь все указанные в статье программы и даже немного больше: исходный код функций и несколько простых примеров, демонстрирующих работу нового алгоритма.

ни одно из условий не выполняется, то в соленой строке он поменяется на символ с соответствующим порядковым номером из строки md5(md5(pass).md5(salt)). Вот, собственно, листинг всего вышеописанного:

```
for ($i=0;$i<strlen($crypted);$i++){
   if (ord($c_text[$i])>=48 and
      ord($c_text[$i])<=57){
         @$temp.=$spec[$c_text[$i]];
   } elseif(ord($c_text[$i])>=97
      and ord($c_text[$i])<=100)
   {
      @$temp.=strtoupper($crypted[$i]);
   } else {
       @$temp.=$crypted[$i];
   }
}
```

После использования этого усиленного метода простенький пароль proba превращается в зверя «E~1*c!f%0#~&C3%9##*!?2#*&1%3~!21». А чтобы такой зверь не отличался от остальных своих MD5-братьев, перед выводом на экран, записью в базу или же сравнением переводим его снова в MD5-хэш. Шанс подбора такого пароля снижается практически до нуля. Вот полный листинг статьи в виде единого кода. Для удобства я сделал его в виде обыкновенной PHP-функции.

```
<?
$salt="123!#&%asgfHTA";
$pass="proba";
function my_crypt($pass,$salt){
   $spec=array('~','!','@','#','$','%','^',
'&','*','?');
   $crypted=md5(md5($salt).md5($pass));
   $c_text=md5($pass);
   for ($i=0;$i<strlen($crypted);$i++){
      if (ord($c_text[$i])>=48
         and ord($c_text[$i])<=57){
          @$temp.=$spec[$c_text[$i]];
      } elseif(ord($c_text[$i])>=97
         and ord($c_text[$i])<=100){
         @$temp.=strtoupper($crypted[$i]);
      } else {
         @$temp.=$crypted[$i];
      }
   }
   return md5($temp);
}
echo my_crypt($pass,$salt);
?>
```

Однако если требуется шифрование данных для их дальнейшего использования в незашифрованном виде, стоит обратить свой взор на симметричные алгоритмы шифрования. В случае необходимости быстрого получения исходного текста, на мой взгляд, стоит задействовать блочный шифр RC4. Это очень быстрый и достаточно легкий в реализации поточный шифр. Если нужно более серьезное шифрование, то возможно использование победителя конкурса AES, объявленного в конце 2000 года, — алгоритма Rijndael. Его разработали два бельгийских криптографа — Димен (Daemen) и Риймен (Rijmen). Применение этих алгоритмов поможет вывести защиту твоего ПО на новый уровень. Однако не стоит исключать и возможность физического доступа к файлам, к примеру, через взлом самого хостинга или веб-сервера. Тут тебе на помощь придут обфускаторы исходного кода систем, самым мощным из которых по-прежнему считается Zend (читай номер за июль 2006 года). После этого даже при завладении злоумышленником исходными кодами проекта дешифровка может занять слишком много времени и быть просто финансово нерентабельной. Как говорилось выше, все сказанное не является призывом к действию, но, судя по статистике потенциально уязвимых веб-проектов, стоит задуматься об этом.

Эта статья не является панацеей от хакеров, дураков, ошибок в коде, но может существенно помочь тебе защитить свою БД. Даже если будет сделан ее дамп, пароли все равно останутся твоей маленькой тайной. Надеюсь, что в дальнейшем будет больше профессиональных проектов, посвященных криптографии и шифрованию, и люди смогут обменивать наработками в этой области.