ISSN 2409-546X

ЮНЫЙ УЧЁНЫЙ

МЕЖДУНАРОДНЫЙ НАУЧНЫЙ ЖУРНАЛ

6+

8
2023

Юный ученый

Международный научный журнал

№ 8 (71) / 2023

Издается с февраля 2015 г.

СОДЕРЖАНИЕ

ИНФОРМАТИКА

Роль хэширования в работе «белого хакера»

Ким Андрей Станиславович, учащийся 10-го класса

Научный руководитель: *Симаков Егор Евгеньевич, учитель математики, информатики и ИКТ;*
Научный руководитель: *Симакова Марина Николаевна, учитель математики*
МАОУ Лицей № 1 г. Южно-Сахалинска

Алгоритмы защиты информации берут свое начало в Римской империи, когда предпринимались первые попытки зашифровать информацию, чтобы она не попала к посторонним людям. Однако любой человек, знающий путь составления шифра мог получить засекреченную информацию. Со временем технологии развивались и сегодня для защиты информации применяются хэш-функции — способ кодировки, который нельзя расшифровать. Такие функции, например, используются для хранения паролей.
Ключевые слова: *хакер, white hat, хэш-функция, хэширование, шифрование.*

Специальность «белый хакер»

Хакер — это человек, который хорошо разбирается в какой-либо технологии, понимает, как она работает и умеет эксплуатировать её недостатки и уязвимости. Тех, кто занимается кибербезопасностью, часто называют «белыми хакерами» (или «white hat»). Они противостоят киберпреступникам — «black hat» и ищут уязвимости, чтобы помочь разработчикам сделать продукт безопаснее. Они разбираются в системных ошибках, умеют их исправлять, защищают информацию с помощью криптографии и выстраивают преграды на пути вредоносных программ.

«Белые хакеры» тестируют защищенность компаний, по согласованию с заказчиком атакуя значимые сегменты инфраструктуры так, как это бы делали реальные киберпреступники. После их работы компания получает отчет о слабых местах в защите и возможность закрыть уязвимости. Спрос на их услуги растет с каждым годом. Однако все российские компании отмечают, что найти квалифицированных специалистов очень сложно. Ведь «белые хакеры» — это специалисты именно в наступательной безопасности, их работа требует опыта, постоянного обучения и развития в профессии. Они решают нестандартные задачи и должны нетривиально мыслить.

Ежегодный мировой ущерб от киберпреступлений составляет более триллиона долларов. В 2022 году российские компании, в том числе объекты критической информационной инфраструктуры, переживали шквал кибератак. По данным из открытых источников с февраля 2022 г. хакеры похитили персональные данные 65 миллионов россиян и скомпрометировали не менее 13 миллионов банковских карт. Атаки хакеров могут привести не только к краже денег со счетов или личных данных, но и к гораздо более серьезным последствиям. Например, в США в 2021 году из-за киберпреступников была нарушена поставка топлива сразу в 11 штатов. А в 2022 году хакеры нанесли удар по испанским больницам и поликлиникам и вывели из строя их системы: врачи не могли оказывать помощь.

Все это послужило толчком к развитию отрасли кибербезопасности: видя реальный ущерб от кибератак, компании стали увеличивать бюджеты на средства защиты и услуги проверки защищенности инфраструктур. По расчетам фонда «Центр стратегических разработок», в следующие пять лет российский рынок кибербезопасности вырастет в 2,5 раза — с 185,9 млрд. до 469 млрд. рублей.

Хэширование данных и хэш-функции

Хэш-функция — функция, осуществляющая преобразование массива входных данных длины в выходную битовую строку, выполняемое определённым алгоритмом. Преобразование называется хэшированием. Исходные данные — входной массив, «ключ» или «сообщение». Результат преобразования — «хэшем», «хэш-код», «хэш-сумма», «сводка».

Хэш-функции могут использоваться для различных целей:

— Криптографические хэш-функции. Хэширование часто используется в алгоритмах электронно-цифровой подписи, где шифруется не само сообщение, а его хэш-код, что уменьшает время вычисления, а также повышает криптостойкость. Также в большинстве случаев вместо паролей хранятся значения их хэш-кодов.

— Контрольные суммы. Такие алгоритмы являются хэш-функциями, вычисляющими контрольный код, используемый для обнаружения ошибок, которые могут возникнуть при передаче и хранении информации. Алго-

ритмы вычисления контрольных сумм гораздо быстрее, чем криптографические хэш-функции. Однако у них полностью отсутствует криптостойкость — возможно легко «подогнать» сообщение под заранее известную контрольную сумму.

— Геометрическое хэширование. Это метод, широко применяемый в компьютерной графике и вычислительной геометрии для решения задач на плоскости или в трёхмерном пространстве. Хэш-функция получает на вход метрическое пространство и разделяет его, создавая сетку из клеток. Хэш-таблица является массивом с двумя или более индексами и называется «файлом сетки». Геометрическое хэширование применяется в телекоммуникациях при работе с многомерными сигналами.

Существует множество алгоритмов хэширования, различающихся различными свойствами:

— разрядность — количество разрядов электронного устройства или шины, одновременно обрабатываемых или передаваемых этим устройством;
— вычислительная сложность — функция зависимости объёма работы, которая выполняется алгоритмом, от размера входных данных;
— криптостойкость — способность противостоять криптоанализу.

Выделяют два важных вида криптографических хэш-функций — ключевые и бесключевые. Ключевые хэш-функции называют кодами аутентификации сообщений. Они дают возможность без дополнительных средств гарантировать как корректность источника данных, так и целостность данных в системах с доверяющими друг другу пользователями.

Можно выделить следующие наиболее распространенные хэш-функции.

— MD5 — генерирует 128-битное хэш-значение. Разработана для использования в криптографии, однако в ней были обнаружены уязвимости, вследствие чего для этой цели она больше не подходит.
— SHA-1 — создает 160-битное хэш-значение. В шестнадцатеричном формате это целое число длиной в 40 символов. Считается более устойчивым к атакам, чем MD5.
— SHA-2 — вторая версия алгоритма, имеет множество разновидностей. Наиболее часто используемая — SHA-256. Алгоритм возвращает 256-битное хэш-значение (шестнадцатеричное значение из 64 символов). Исследования показывают, что этот алгоритм значительно превосходит в безопасности MD5 и SHA-1. Если рассматривать с точки зрения производительности, то вычисление хэша с его помощью происходит на 20–30 % медленнее, чем с использованием MD5 или SHA-1.
— SHA-3. Этот алгоритм хэширования был разработан в конце 2015 года и до сих пор еще не получил широкого применения.

«Хорошая» хэш-функция должна удовлетворять двум свойствам: быстрое вычисление и минимальное количество «коллизий» — равенств значений хэш-функции на двух различных блоках данных. Рассмотрим несколько примеров реализаций «хэш-функций».

Пример 1. Хэш-функция может вычислять «хэш» как остаток от деления входных данных на M — количество всех возможных «хэшей» (выходных данных): *h(k) = k mod M*. При чётном M значение функции будет чётным при чётном k и нечётным — при нечётном k. Также не следует использовать в качестве M степень основания системы счисления компьютера, так как «хэш-код» будет зависеть только от нескольких цифр числа k, расположенных справа, что приведёт к большому количеству коллизий.

Пример 2. «Хэш-код» как набор коэффициентов получаемого полинома. Хэш-функция может выполнять деление входных данных на полином по модулю два. M должна являться степенью двойки, а бинарные ключи ($K = k_{n-1} k_{n-2} \dots k_0$) представляются в виде полиномов, в качестве «хэш-кода» берутся значения коэффициентов полинома, полученного как остаток от деления входных данных K на заранее выбранный полином P степени m:

$$K(x) \mod P(x) = h_{m-1}x^{m-1} + \cdots + h_1 x + h_0$$
$$h(x) = h_{m-1} \dots h_1 h_0$$

При правильном выборе P(x) гарантируется отсутствие коллизий между почти одинаковыми ключами.

Пример 3. «Хэш-функция», основанная на умножении. Пусть w — количество чисел, представимых машинным словом. Например, для 32-разрядных компьютеров $w = 2^{32}$. Выберем некую константу A так, чтобы A была взаимно простой с w. Тогда $h(k) = \left\lfloor M \left[\frac{A}{w} * K \right] \right\rfloor$. В этом случае на компьютере с двоичной системой счисления M является степенью двойки, и h(K) будет состоять из старших битов правой половины A*K.

Разработка программы-шифровальщика на основе функции SHA-256

Рассмотрим подробнее алгоритм SHA-256. Исходное сообщение после дополнения разбивается на блоки, каждый блок — на 16 слов. Алгоритм пропускает каждый блок сообщения через цикл с 64 или 80 итерациями. На каждой итерации 2 слова преобразуются, функцию преобразования задают остальные слова. Результаты обработки каждого блока складываются, сумма является значением хэш-функции. Инициализация внутреннего состояния производится результатом обработки предыдущего блока. Поэтому независимо обрабатывать блоки и складывать результаты нельзя.

Таким образом, принцип работы алгоритма на основе хэш-функции SHA-256 можно разделить на три этапа:

1. Этап предварительной обработки
2. Начальные хэш-значения
3. Фаза вычисления хэша

Далее приведен программный код — реализация описанного алгоритма. В начале дадим некоторые общие пояснения:

— Все переменные беззнаковые, имеют размер 32 бита и при вычислениях суммируются по модулю 2^{32};
— message — исходное двоичное сообщение;
— m — преобразованное сообщение.

Этап 1. Инициализация переменных.

Первые 32 бита дробных частей квадратных корней первых восьми простых чисел [от 2 до 19]:

```
h0 := 0x6A09E667
h1 := 0xBB67AE85
h2 := 0x3C6EF372
h3 := 0xA54FF53A
h4 := 0x510E527F
h5 := 0x9B05688C
h6 := 0x1F83D9AB
h7 := 0x5BE0CD19
```

Таблица констант. Первые 32 бита дробных частей кубических корней первых 64 простых чисел [от 2 до 311]:

```
k[0..63] := 0x428A2F98, 0x71374491, 0xB5C0FBCF, 0xE9B5DBA5,
0x3956C25B, 0x59F111F1, 0x923F82A4, 0xAB1C5ED5, 0xD807AA98,
0x12835B01, 0x243185BE, 0x550C7DC3, 0x72BE5D74, 0x80DEB1FE,
0x9BDC06A7, 0xC19BF174, 0xE49B69C1, 0xEFBE4786, 0x0FC19DC6,
0x240CA1CC, 0x2DE92C6F, 0x4A7484AA, 0x5CB0A9DC, 0x76F988DA,
0x983E5152, 0xA831C66D, 0xB00327C8, 0xBF597FC7, 0xC6E00BF3,
0xD5A79147, 0x06CA6351, 0x14292967, 0x27B70A85, 0x2E1B2138,
0x4D2C6DFC, 0x53380D13, 0x650A7354, 0x766A0ABB, 0x81C2C92E,
0x92722C85, 0xA2BFE8A1, 0xA81A664B, 0xC24B8B70, 0xC76C51A3,
0xD192E819, 0xD6990624, 0xF40E3585, 0x106AA070, 0x19A4C116,
0x1E376C08, 0x2748774C, 0x34B0BCB5, 0x391C0CB3, 0x4ED8AA4A,
0x5B9CCA4F, 0x682E6FF3, 0x748F82EE, 0x78A5636F, 0x84C87814,
0x8CC70208, 0x90BEFFFA, 0xA4506CEB, 0xBEF9A3F7, 0xC67178F2
```

Этап 2. Предварительная обработка.

```
m := message ‖ [единичный бит]
m := m ‖ [k нулевых бит], где k — наименьшее неотрицательное число,
такое, что (L + 1 + K) mod 512 = 448, где L — число бит в сообщении
m := m ‖ Длина(message) — длина исходного сообщения в битах в виде
64-битного числа с порядком байтов от старшего к младшему
```

Этап 3. Сообщение обрабатывается последовательными порциями по 512 бит.

```
разбить сообщение на куски по 512 бит
для каждого куска:
разбить кусок на 16 слов длиной 32 бита (с порядком байтов от старшего
к младшему внутри слова): w[0..15]
```

Этап 4. Создание дополнительных 48 слов.

```
для i от 16 до 63
  s0 := (w[i-15] rotr 7) xor (w[i-15] rotr 18) xor (w[i-15] shr 3)
  s1 := (w[i-2] rotr 17) xor (w[i-2] rotr 19) xor (w[i-2] shr 10)
  w[i] := w[i-16] + s0 + w[i-7] + s1
```

Этап 5. Инициализация вспомогательных переменных.

```
a := h0    b := h1    c := h2    d := h3
e := h4    f := h5    g := h6    h := h7
```

Этап 6. Основной цикл.

```
для i от 0 до 63
    Σ0 := (a rotr 2) xor (a rotr 13) xor (a rotr 22)
    Ma := (a and b) xor (a and c) xor (b and c)
    t2 := Σ0 + Ma
    Σ1 := (e rotr 6) xor (e rotr 11) xor (e rotr 25)
    Ch := (e and f) xor ((not e) and g)
    t1 := h + Σ1 + Ch + k[i] + w[i]
    h := g      g := f      f := e      e := d + t1
    d := c      c := b      b := a      a := t1 + t2
```

Этап 7. Полученные значения добавляются к ранее вычисленному результату.

```
h0 := h0 + a   h1 := h1 + b   h2 := h2 + c   h3 := h3 + d
h4 := h4 + e   h5 := h5 + f   h6 := h6 + g   h7 := h7 + h
```

Этап 8. Получаем итоговое значение хэша.

```
digest = hash = h0 ‖ h1 ‖ h2 ‖ h3 ‖ h4 ‖ h5 ‖ h6 ‖ h7
```

Результат кодирования фразы «The quick brown fox jumps over the lazy dog»

```
SHA-256(“The quick brown fox jumps over the lazy dog”)
 = D7A8FBB3 07D78094 69CA9ABC B0082E4F 8D5651E4 6D3CDB76
2D02D0BF 37C9E592
```

Тестирование программы-шифровальщика

Часто хакер проделывает огромную работу для проникновения в систему, и от получения полного доступа его отделяет один шаг — подбор пароля к хэшу. Восстановление паролей к хэшам или хэшкрекинг — сложный процесс, для которого требуются знания в различных областях — криптографии, комбинаторике, программировании. При этом «белый хакер» полностью изолирован от применения сломанных паролей для доступа к чужим аккаунтам. На соответствующих форумах публикуются только хэши (или целые списки паролей) для расшифровки. Эти списки не содержат ни имени ресурса, ни имен пользователей, ни почтовых ящиков, ни IP-адресов, никакой другой приватной

информации. Основным инструментом для хэшкрекинга являются словари, которые должны состоять из реальных паролей пользователей.

Предположим, что имеется огромный список различных хэшей, который надо быстро обработать. Однако атака по большому словарю даже на мощной системе будет идти много суток. Чтобы ускорить процесс используются особые, частотные словари, в которых пароли отсортированы в порядке убывания частоты их употребления. Эффективней будет проверить хэши сначала на самые часто употребляемые пароли, затем — на более редкие и так далее. Это позволит быстро сломать все популярные пароли и существенно облегчить список хэшей для последующей работы.

Рассматриваемый в рамках данной статьи процесс трудно назвать взломом в прямом смысле этого слова. Это скорее перебор. На профессиональном языке этот способ называют брутфорсом. Данный метод заключается в следующем. Программа для брутфорса начинает подбирать определённые пароли из списка или генерирует их сама по заданному алгоритму, кодирует их заданным образом (в данном случае алгоритмом sha-256) и сравнивает с хэшом.

Для «взлома» описанного выше алгоритма использовался инструмент для пентестеров hashcat. Данная программа является бесплатной и на сегодняшний день одной из самых эффективных. Она также может похвастаться высокой скоростью перебора пароля. Однако, скорость также зависит от характеристик устройства, а именно процессора и видеокарты. На используемом компьютере скорость процесса брутфорса составляла 1255400 паролей/сек.

```
Speed.#1.........:  1255.4 MH/s (65.81ms) @ Accel:64 Loops:512 Thr:512 Vec:1
Speed.#*.........:  1255.4 MH/s

---------------------------
* Hash-Mode 1400 (SHA2-256)
---------------------------
```

Предположим, что целью «белого хакера» является проверка, какая существует возможность взломать внутренний месенджер. С помощью каких-либо манипуляций был получен хэш пароля администратора:

63b2baa40b3007b5149e08ed0e5ad83b84512135d0ec9ffe75dbbaa589445aa1

Далее возьмём словарь — список паролей для брутфорса. Будем выбирать из него самые популярные варианты, и с помощью hashcat попробуем разгадать пароль.

```
Dictionary cache built:
* Filename..: C:\Users\User\Desktop\`\ignis-1M.txt/ignis-1M.txt
* Passwords.: 1000000
* Bytes.....: 8755172
* Keyspace..: 1000000
* Runtime...: 1 sec

63b2baa40b3007b5149e08ed0e5ad83b84512135d0ec9ffe75dbbaa589445aa1:qwerty_01

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1400 (SHA2-256)
Hash.Target......: 63b2baa40b3007b5149e08ed0e5ad83b84512135d0ec9ffe75d...445aa1
Time.Started.....: Thu Feb 02 20:04:23 2023 (0 secs)
Time.Estimated...: Thu Feb 02 20:04:23 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (C:\Users\User\Desktop\`\ignis-1M.txt/ignis-1M.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  1657.1 kH/s (10.99ms) @ Accel:256 Loops:1 Thr:64 Vec:1
Speed.#*.........:  1657.1 kH/s
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 655360/1000000 (65.54%)
Rejected.........: 0/655360 (0.00%)
Restore.Point....: 573440/1000000 (57.34%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 2360881 -> 959799
Hardware.Mon.#1..: Temp: 55c Fan:  0% Util: 66% Core: 135MHz Mem: 405MHz Bus:16

Started: Thu Feb 02 20:03:16 2023
Stopped: Thu Feb 02 20:04:25 2023
```

В итоге пароль был подобран за одну секунду! Пароль был **qwerty_01.** Цель достигнута — пароль определен!

ЛИТЕРАТУРА:

1. Галуев, Г. А. Математические основы криптологии: Учебно-методическое пособие / Г. А. Галуев — Таганрог: Изд-во ТРТУ, 2003. — 120 с.
2. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. / Б. Шнайер — М.: Триумф, 2012. — 816 с. — ISBN 978–5–9908462–4–1.
3. Портал «Antichat» — URL: https://forum.antichat.com/ (дата обращения: 12.10.2022).
4. Портал «Innovakon». Статья «Умный брутфорс. Обнаружение brute force атак. Что такое хэшкрекинг» — URL: https://innovakon.ru/services/umnyi-brutfors-obnaruzhenie-brute-force-atak-chto-takoe-heshkreking.html (дата обращения: 23.11.2022).
5. Портал «Insidepro» — URL: https://forum.insidepro.com/ (дата обращения: 12.01.2023).
6. Портал «Internet Union». Статья «Лучшие инструменты пен-тестера: брутфорс паролей» — URL: https://iuni.ru/the-best-pentester-tools-password-brute-force-how-to-guard-against-brute-force-attacks.html (дата обращения: 12.01.2023).
7. Портал «Skillbox». Статья «Наступательная кибербезопасность: подробный гайд от «белого» хакера» — URL: https://skillbox.ru/media/code/nastupatelnaya-kiberbezopasnost-podrobnyy-gayd-ot-belogo-khakera/ (дата обращения: 04.12.2022).
8. Портал «Wikipedia». Статья «Хэш-функция» — URL: https://ru.wikipedia.org/wiki/Хэш-функция (дата обращения: 24.11.2022).
9. Портал «ТАСС». Статья «Кто такие «белые хакеры» и почему за ними идет охота» — URL: https://tass.ru/obschestvo/16592637 (дата обращения: 15.11.2022).