ФГБОУ ВПО «Чувашский государственный университет им. И. Н. Ульянова»

Факультет ИВТ

Кафедра Вычислительной Техники

Лабораторная работа №3

**Атака дней рождения**

**Выполнил:**

Студент гр. ИВТ-21-11

Исаков Тимофей

**Проверил:**

Лобастова О. А.

Чебоксары 2015г.

**Цель.**

Исследование атаки типа «Атаки дней рождений».

**Результат выполнения.**

Исследование проводилось на словаре англоязычных паролей.

Поиск коллизий.

Start generation

Collisions: 2585710

Finish generation

Generation time

hours:0

minutes:0

seconds:0

milliseconds:436

microseconds:13

nanoseconds:700

Start testing on the dictionary

Finished

Dictionary test time

hours:0

minutes:0

seconds:0

milliseconds:850

microseconds:367

nanoseconds:300

Collisions:62

Not found:3160058

Если осуществлять перебор в одном потоке, то данный процесс становится довольно медленным, так как при генерации у нас постоянно будут возникать коллизии (для данной хеш-функции). Что существенно будет замедлять процесс. Но в принципе, если у нас часто будет появляться коллизии, то мы можем сделать вывод, который как раз и следует из определения атак дней рождений, чем больше мы сгенерируем значений, тем больше вероятность коллизии.

Проверим. Сгенерируем 1млн хешей.

Start generation

Collisions: 228218204

Finish generation

Generation time

hours:0

minutes:0

seconds:43

milliseconds:707

microseconds:696

nanoseconds:600

Start testing on the dictionary

Finished

Dictionary test time

hours:0

minutes:0

seconds:1

milliseconds:4

microseconds:25

nanoseconds:300

Collisions:625

Not found:3159495

Число коллизий заметно выросло (в 10 раз).

Увеличим до 2млн:

Start generation

Collisions: 541544242

Finish generation

Generation time

hours:0

minutes:1

seconds:57

milliseconds:394

microseconds:315

nanoseconds:500

Start testing on the dictionary

Finished

Dictionary test time

hours:0

minutes:0

seconds:2

milliseconds:804

microseconds:74

nanoseconds:500

Collisions:1240

Not found:3158880

Увеличим до 10 млн:

Start generation

Collisions: 3076282602

Finish generation

Generation time

hours:0

minutes:10

seconds:42

milliseconds:834

microseconds:460

nanoseconds:600

Start testing on the dictionary

Finished

Dictionary test time

hours:0

minutes:0

seconds:1

milliseconds:139

microseconds:759

nanoseconds:800

Collisions:7003

Not found:3153117

Как мы видим, чем больше мы генерируем хешей, тем больше мы получаем коллизий. Мы бы получали больше коллизий, если бы генерировали по словарю, а не последовательно сгенерированные строки.

Можно проверить сколько коллизий возникнет при генерации хешей по словарю:

Start testing on the dictionary

Finished

Dictionary test time

hours:0

minutes:0

seconds:4

milliseconds:980

microseconds:128

nanoseconds:300

Collisions:4337

Original:3155783

Проведем прямую атаку дней рождений, то есть сгенерируем заранее 2^31 сообщений. Тогда вероятность совпадения будет больше 0,5.

Start generation

Finish generation

Generation time

hours:0

minutes:4

seconds:28

milliseconds:307

microseconds:869

nanoseconds:400

Start testing on the dictionary

Finished

Dictionary test time

hours:0

minutes:0

seconds:1

milliseconds:88

microseconds:724

nanoseconds:900

Collisions:5523

Not found:3154597

А потом случайно будем подавать пароли из этого списка, чтобы найти искомый. А можно и последовательно☺ тем более что 2^31 это не такой и большой диапазон.

Исходный код программы.

#include <iostream>

#include <map>

#include <string>

#include <fstream>

#include <chrono>

// 32битная хеш-функция

unsigned int HashH37(const char \* str)

{

unsigned int hash = 2139062143;

for(; \*str; str++)

hash = 37 \* hash + \*str;

return hash;

}

// атака на коллизии

void generation(std::map<int, std::string> &hashes) {

std::string buf;

for (int i = 0; i < 3; ++i) {

buf.push\_back(char(-127));

}

int pos = buf.length()-1;

int collision = 0;

while (hashes.size() < 65536) {

if ( buf[pos] == 127) {

int b\_pos = pos;

while (buf[b\_pos] == 127 && b\_pos >= 0) {

buf[b\_pos] = char(-127);

b\_pos -= 1;

}

buf[b\_pos] = char((int(buf[b\_pos]) + 1));;

}

buf[pos] = char((int(buf[pos]) + 1));

int hash = HashH37(buf.c\_str());

if (hashes.find(hash) != hashes.end()) {

++collision;

}

hashes[hash] = buf;

}

std::cout <<"Collisions: " << collision << std::endl;

}

// полная атака

void generation\_full(std::map<int, std::string> &hashes) {

std::string buf;

for (int j = 0; j < 8; ++j) {

buf.push\_back(char(-127));

}

int pos = buf.length()-1;

unsigned long long int collision = 0;

while (hashes.size() < 4294967295) {

if ( buf[pos] == 127) {

int b\_pos = pos;

while (buf[b\_pos] == 127 && b\_pos >= 0) {

buf[b\_pos] = char(-127);

b\_pos -= 1;

}

buf[b\_pos] = char((int(buf[b\_pos]) + 1));;

}

buf[pos] = char((int(buf[pos]) + 1));

int hash = HashH37(buf.c\_str());

if (hashes.find(hash) != hashes.end()) {

++collision;

}

hashes[hash] = buf;

}

std::cout <<"Collisions: " << collision << std::endl;

}

int main() {

std::map<int, std::string> hashes;

std::cout << "Start generation" << std::endl;

auto start = std::chrono::steady\_clock::now();

generation\_full(hashes);

auto stop = std::chrono::steady\_clock::now();

std::cout << "Finish generation" << std::endl;

std::cout <<"Generation time" << std::endl;

std::cout << "hours:" << std::chrono::duration\_cast<std::chrono::hours>(stop - start).count() << std::endl

<< "minutes:" << std::chrono::duration\_cast<std::chrono::minutes>(stop - start).count() % 60 << std::endl

<< "seconds:" << std::chrono::duration\_cast<std::chrono::seconds>(stop - start).count() % 60 << std::endl

<< "milliseconds:" << std::chrono::duration\_cast<std::chrono::milliseconds>(stop - start).count() % 1000 << std::endl

<< "microseconds:" << std::chrono::duration\_cast<std::chrono::microseconds>(stop - start).count() % 1000 << std::endl

<< "nanoseconds:" << std::chrono::duration\_cast<std::chrono::nanoseconds>(stop - start).count() % 1000 << std::endl;

std::ifstream fin("english.txt");

std::string pswd;

unsigned int collision = 0;

unsigned int not\_found = 0;

std::cout << "Start testing on the dictionary" << std::endl;

char buf[100];

start = std::chrono::steady\_clock::now();

while (fin.getline(buf, 100)) {

int hash = HashH37(buf);

if (hashes.find(hash) != hashes.end()) {

++collision;

} else {

++not\_found;

}

}

stop = std::chrono::steady\_clock::now();

std::cout << "Finished" << std::endl;

std::cout <<"Dictionary test time" << std::endl;

std::cout << "hours:" << std::chrono::duration\_cast<std::chrono::hours>(stop - start).count() << std::endl

<< "minutes:" << std::chrono::duration\_cast<std::chrono::minutes>(stop - start).count() % 60 << std::endl

<< "seconds:" << std::chrono::duration\_cast<std::chrono::seconds>(stop - start).count() % 60 << std::endl

<< "milliseconds:" << std::chrono::duration\_cast<std::chrono::milliseconds>(stop - start).count() % 1000 << std::endl

<< "microseconds:" << std::chrono::duration\_cast<std::chrono::microseconds>(stop - start).count() % 1000 << std::endl

<< "nanoseconds:" << std::chrono::duration\_cast<std::chrono::nanoseconds>(stop - start).count() % 1000 << std::endl;

std::cout << "Collisions:" << collision << std::endl;

std::cout << "Not found:" << not\_found << std::endl;

return 0;

}