МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.

«___» _____2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТР"

Тема: Програмний засіб машинного навчання для визначення стійкості паролів до зламу

Виконавець: Цирень М.В..

Керівник: Ткаченко В.Г.

Нормоконтролер: Тупота Є.В.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Освітнього ступеня магістр
Спеціальність 123 "Комп'ютерна інженерія"
(шифр, найменування)
Спеціалізація 123.02 "Системне програмування"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Литвиненко О. Є.

«____»_____2020 р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)

Циреня Максима Валентиновича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема роботи: Програмний засіб машинного навчання для визначення стійкості паролів до зламу
затверджена наказом ректора від "27" серпня 2020 року № 1203 /ст.
2. Термін виконання роботи: з 05.10.2020 до 31.12.2020
3. Вихідні дані до роботи: ознаки стійкості паролів до зламу

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) аналіз методів оцінки стійкості паролів до зламу;
2) криптографічні методи перевірки стійкості паролів до зламу;
3) програмна реалізація криптографічних алгоритмів для перевірки стійкості паролів до зламу.

5. Перелік обов'язкового графічного матеріалу:
1) зв'язки між рівнями програмного модуля;
2) послідовність виконання функцій програмного модуля;
3) *web*-інтерфейс налаштування модуля перевірки стійкості пароля до зламу;
4) обробки вибору системи шифрування (схема алгоритму);
5) Робота процедури обчислення хеш-функції *MD*4 (схема алгоритму)

6. Календарний план

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1		05.10.2020 – 06.10.2020	
2		07.10.2020 – 10.11.2020	
3		11.11.2020 – 14.11.2020	
4		15.11.2020 – 19.11.2020	
5		29.11.2020 – 13.12.2020	
6		21.12.2020 – 25.12.2020	

7. Дата видачі завдання 05.10.2020

Керівник Ткаченко В.Г.
(підпис)

Завдання прийняв до виконання Цирень М.В..
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Програмний засіб машинного навчання для визначення стійкості паролів до зламу”: 83 с., 25 рис., 25 літературних джерел, 1 додаток.

ПАРОЛЬ, КРИПТОГРАФІЯ, СТІЙКІСТЬ ПАРОЛЯ, ХЕШ-КОД, ПРОГРАМНИЙ КОД

Мета дослідження – розробити програмний засіб машинного навчання для визначення стійкості паролів до зламу.

Об’єкт дослідження – оцінювання стійкості паролів.

Предмет дослідження – програмний засіб машинного навчання для визначення стійкості паролів до зламу.

Встановлено, що запропонований метод перевірки стійкості паролів до зламу відповідає ринковим вимогам.

Результати дипломної роботи рекомендується використовувати при стовренні паролів та у навчальному процесі, як приклад реалізації криптографічних систем.

Публікація: Вовк В.А., Цирень М.В. Можливості використання нейронних мереж для представлення зображень та визначення стійкості паролів до зламу// Тези доповідей наук.-практ. конф. “Сучасні тенденції розвитку системного програмування” (25-26 листопада 2020 р.). – К.: НАУ, 2020. – С. 17.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ВСТУП

Паролі є найбільш поширений інструмент, який використовується для отримання доступу до інформаційних ресурсів.

Кожен день комп'ютери, *Wi-Fi* роутери, мобільні пристрої, розумна техніка в пошуках вразливостей або слабкого захисту скануються хакерами з усього світу. І в разі виявлення незахищених чи уразливого пристрою вони зламують його і використовують для розсилки спаму або видобутку криптовалюта. У гіршому випадку зашифрують ваші дані і вимагатимуть гроші або здійснять зі зламаного пристрої протиправні дії, внаслідок яких правоохоронні органи нагрянуть до вас з обшуком.

Ламають все, що можуть зламати, аж до розумних будинків. Вимога одна - пристрій повинен мати доступ в інтернет. Особливий інтерес для хакерів представляють комп'ютери, сервери, мобільні пристрої і роутери, так як їх легше монетизувати, іншими словами, заробити на них гроші. В першу чергу зловмисникові цікаво автоматично монетизувати зламані пристрої, наприклад, можна зламати і зашифрувати ваш комп'ютер.

З високою часткою ймовірності на комп'ютері у вас виявиться інформація, за розшифровку якої ви будете готові заплатити. Всі дії від сканування мережі до шифрування відбуваються автоматично, зловмисникові залишається тільки підтримувати працездатність системи і виводити отримані від жертв кошти. Зламавши вашу розумну пральну машину, не так просто монетизувати її. З іншого боку, нам відомі ботнети, що складаються з розумних речей: пральних машин, холодильників, телевізорів, чайників, кавоварок та іншої побутової техніки, навіть розумних іграшок і відеокамер. Подібні ботнети вміють і слати спам, і проводити *DDoS*-атаки (до речі, вражаючою потужності), і Майн біткоіни, хоча і сильно поступаючись зламаним персональних комп'ютерів.

Найвідоміший подібний ботнет - *Mirai*. Цей ботнет брав участь в найпотужнішою атаці на експерта в області безпеки Брайана Кребса, в результаті якої його сайт був виведений з ладу. Кращим прикладом, що демонструє міць

ботнету *Mirai*, є відома атака на найбільшого *DNS*-провайдера США 21 жовтня 2016 року, коли недоступні виявилися деякі сервіси типу *Twitter* і *Amazon*, а по всьому східному узбережжю Штатів мережа працювала з серйозними обмеженнями.

Настав той самий час, коли чайники і кавомашини можуть атакувати цілу державу, поки, щоправда, в інтернеті. Пізніше *Mirai* був доопрацьований для Майнінг біткоіни на розумних пристроях. Хоча ефективність розумних пристроїв для Майнінг біткоіни мінімальна, вона не вимагає від власників особливих витрат. *Mirai* хоча і найяскравіший, але далеко не єдиний приклад, так само функціонують і інші ботнети, наприклад *Hajime*. Навіщо ламають Вимагання Напевно, найприбутковіший спосіб монетизації - шифрування вмісту пристроїв і вимога викупу. Комп'ютер жертви зламуються, диски шифруються, ключ шифрування відправляється на сервер зловмисників, а жертві видається інформація, як оплатити і потім розшифрувати свої дані. Ось приклад подібного повідомлення: Як правило, для отримання ключа потрібно заплатити від 300 до 1000 доларів.

Наприклад, для ряду шифрувальників створено програмне забезпечення для дешифрування даних без оплати зловмисникам. Подібне стосується шифрувальників *GandCrab, XData, Bitcryptor* і багатьох інших. Деякі програми тільки імітують шифрування, приховуючи файли і вимагаючи викуп. Їх ще називають фейковий шифрувальником, наприклад *Fake Cerber*. У більшості випадків при зіткненні з фейковий вимагачами допомагає перезавантаження пристрою, рідше доводиться трохи покопатися в налаштуваннях системи, щоб отримати доступ до прихованих файлів. Іноді творці програм-вимагачів просто обманюють жертв: їх програми вміють тільки шифрувати без можливості дешифрування файлів. Це Вайпер, що маскуються під вимагачів.

Прикладом такого Вайпера був сумнозвісний *Petya*, який після шифрування даних вимагав викуп, не маючи ні можливості відстежити, від кого конкретно прийшов платіж, ні розшифрувати дані. *Petya* - це не вимагач, це добре продумане кіберзброя масового ураження, мета якого - знищувати і виводити з ладу. Іноді вимагач перетворюється в Вайпера внаслідок помилки,

так було у шифрувальника *AVCrypt*, де через помилку розробників файли віддалялися безслідно. Для вимоги викупу годі й шифрувати файли і не імітувати їх шифрування, іноді жертву намагаються просто залякати, наприклад передачею його особистих даних спецслужбам. *LeakerLocker* - вимагач для *Android*-пристроїв - погрожував відправити персональні фото, *SMS*, історію дзвінків, історію браузера по всьому списку контактів телефону. На роздуми і оплату давалося 72 години. Як не дивно, для багатьох з нас простіше втратити дані, ніж допустити, щоб вони були розіслані всім друзі, колегам, знайомим, родичам. Та й вимагав *LeakerLocker* не так багато - всього $ 50.

Майнінг криптовалюта Ймовірно, це один з найшкідливіших способів монетизації, при якому зламані пристрій просто буде Майні криптовалюта для зловмисників. Для жертви це позначиться зниженням продуктивності, більш швидким виходом з ладу комплектуючих пристрою через постійну високого навантаження і помітним збільшенням рахунків за електрику. У деяких країнах, де Майнінг криптовалюта заборонений, це може спричинити проблеми з законом. *DDoS* Якщо говорити простими словами, *DDoS* - це створення великої кількості запитів для уповільнення роботи або повного виведення з ладу кінцевої мети. Як правило, метою є сервер, але насправді атаку можна провести навіть на *Wi-Fi* роутер, відключивши в потрібний момент інтернет жертві. Зламаний комп'ютер по команді зловмисника починає масово відправляти запити, це ж роблять інші жертви до тих пір, поки у мети атаки не закінчуються ресурси для їх обробки. Паралельно справжній клієнт при спробі підключитися, наприклад для покупки на сайті, або довго чекає обробки свого запиту, або взагалі не може підключитися до сайту. І, природно, клієнт піде, швидше за все до конкурента мети атаки, бізнес буде втрачати клієнтів, а разом з ними і гроші. Є дві основні моделі монетизації *DDoS*-ботнету. Перша - атакувати комерційні проекти і вимагати гроші. Уявіть ситуацію: ви - власник великого інтернет-магазину, ваш сайт приносить вам гроші, і тут починається *DDoS*-атака. Ви починаєте втрачати гроші, і в цей момент на вас виходить організатор атаки і пропонує угоду: ви платите йому гроші, а він зупиняє атаку. Варіантів у вас три: перший - втрачати гроші і чекати, поки зловмисникам набридне, - це безумовно не найкращий

варіант для бізнесу, другий варіант - заплатити гроші сервісів, які займаються захистом від *DDoS,* і третій - заплатити зловмисникові.

Друга модель монетизації - надавати послуги *DDoS*-ботнету за винагороду. Сьогодні бізнес-війни все активніше переміщаються в Інтернет і *DDoS* стає грізною зброєю для нанесення шкоди конкурентам. Напевно, у вас виникло питання про наслідки ненавмисного участі в *DDoS*-атаці для жертви. Мені не відомі випадки, коли ненавмисне участь в *DDoS*-атаці призводило б до проблем з законом, а ось проблеми при роботі в мережі дуже ймовірні. Ваш *IP*-адреса буде додано до бази скомпрометованих *IP*, і сайти будуть обмежувати вас, виходячи зі своєї політики. Наприклад, *Google* і багато великих сайти можуть вимагати ввести капчу, сайти поменше можуть просто заборонити вам доступ. Ваш інтернет-провайдер може обмежити вам доступ і вимагати усунути проблему. В інтернеті ви можете знайти чимало скарг, пов'язаних з блокуванням інтернету за шкідливу активність.

Приховування скоєних злочинів Злочинцям, що чинять кіберзлочини, необхідно приховувати своє справжнє місцезнаходження, і для цих цілей їм потрібні інструменти: віддалені робочі столи, *VPN*, *SSH* і *proxy*. Скомпрометований комп'ютер жертви може бути використаний як віддалений робочий стіл, на якому буде запущена програма для вчинення злочинних дій. Наприклад, кардерам необхідні віддалені столи для гри в покер на крадені гроші з метою подальшого їх переведення в готівку. На зламаному *Wi-Fi* роутер може бути розгорнуто *VPN*, через який шахраї будуть скоювати злочини. Скомпрометований мобільний телефон може бути використаний як *proxy*. І ось в цьому випадку наслідки для жертви куди серйозніше: зламані пристрої можуть бути використані для продажу наркотиків, терористичної діяльності, кардинг, шахрайства. Це не обов'язково призведе до обшуку в будинку жертви або арешту, хоча подібних історій чимало, проте жертва ризикує потрапити в список підозрюваних і в розробку правоохоронних органів. Вчинення комп'ютерних атак Для комп'ютерних атак потрібно сканування мережі та, в залежності від типу атаки, підбір паролів або експлуатування вразливостей. Все це вимагає ресурсів. Зловмисникам треба орендувати сервера, платити гроші, сервера

будуть блокуватися через скарг, що надходять, вимагати періодичного обслуговування ... Але є альтернативний безкоштовний варіант: комп'ютер жертви може стати ланкою в ланцюзі зловмисників і бути використаний для подальшого злому інших пристроїв. Більшість шкідливих програм поширювалися саме так: скомпрометований комп'ютер заражав інші комп'ютери, ті в свою чергу треті.

Це може привести до реальних проблем, так як в процесі атаки скомпрометований пристрій буде сканувати інтернет і шукати пристрої, на які можна провести атаку: це можуть бути урядові сервера або, наприклад, сайт СБУ. Правоохоронні органи зобов'язані реагувати на дані атаки. Показ реклами Даний спосіб монетизації, як правило, використовується для зламаних роутерів. На скомпрометувати *Wi-Fi* роутер відбувається підміна основного *DNS*-сервера, і всі запити до сайтів проходять через сервер зловмисника, де жертві додається реклама. Найпростіший варіант підміни - періодична переадресація на рекламний Лендінгем, при цьому не важливо, який сайт буде введений користувачем. Більш складний варіант передбачає підміну рекламних блоків *Google* і банерів на діючих сайтах. Напевно, ви здогадуєтеся, що

Для користувачів це один з найбільш зручних варіантів, який, до того ж, не вимагає наявності будь-якого додаткового обладнання або спеціальних навичок. Все, що потрібно - це придумати і запам'ятати стійкий пароль (або кілька паролів). Здається, це зовсім не складно? На жаль, сувора реальність вносить свої корективи в цей процес.

Відповідно до проведеного американським *Ponemon Institute* опитуванням, 88% респондентів протягом останніх двох років забували свій пароль хоча б один раз. Часто співробітники компаній вважають політику безпеки щодо паролів зайвої, і вимога міняти пароль кожні 30 днів викликає у них обурення. У таких випадках в якості нового пароля використовується трохи видозмінений старий, до якого додається «0», «1» або поточний рік.

На жаль, з точки зору комп'ютерної безпеки парольний аутентифікацію далеко не завжди можна назвати вдалим вибором. Як і багато технічних рішень, вона страждає від двох проблем: людського фактора і технічну недосконаліть.

– людський фактор (помилки користувачів). Багато людей не можуть (або просто не хочуть, не бачать сенсу) запам'ятовувати стійкі паролі, оскільки це об'єктивно складно. Тому вони використовують прості, нестійкі паролі, або, якщо їх все ж змушують використовувати стійкі, записують їх на килимках для мишки, на звороті клавіатури або на стікерах, приклеєних до монітора.

– технічна недосконаліть (помилки розробників), найчастіше полягає в помилках, допущених на етапах проектування і / або реалізації програмного забезпечення, здійснює перевірки паролів.

Мета дослідження – розробити програмний засіб машинного навчання для визначення стійкості паролів до зламу.

Об’єкт дослідження – оцінювання стійкості паролів.

Предмет дослідження – програмний засіб машинного навчання для визначення стійкості паролів до зламу.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ОЦІНКИ СТІЙКОСТІ ПАРОЛІВ ДО ЗЛАМУ

1.1. Ознаки стійкості паролів

Мірою стійкості паролів традиційно є ентропія - міра невизначеності, яка вимірюється зазвичай в бітах. ентропіяв 1 біт відповідає невизначеності вибору з двох паролів, в 2 біта - з 4 паролів, у 3 біта - з 8 паролів і т. д. Ентропія в N біт відповідає невизначеності вибору з 2^N паролів. У разі випадкових паролів (наприклад, згенерованих за допомогою генератора випадкових чисел) ентропія обчислюється досить просто: вона дорівнює логарифму по підставі два кількості можливих паролів для заданих параметрів. Так, для випадкового пароля довжиною N символів, складеного з алфавіту, що містить M букв, ентропія буде дорівнює:

$$E = log_2 M^N.$$

Значення ентропії для деяких довжин паролів і наборів символів представлені в таблиці 1.1.

Якщо ж пароль згенерований НЕ неупередженим генератором випадкових чисел, а людиною, то обчислити його ентропію набагато важче. Найпоширенішим підходом до підрахунку ентропії в цьому випадку є підхід, який був запропонований американським інститутом *NIST*:

- ентропія першого символу пароля становить 4 біта;
- ентропія наступних семи символів пароля становить 2 біта на символ;
- ентропія символів з 9-го по 20-й становить 1,5 біта на символ;
- всі наступні символи мають ентропію 1 біт на символ;
- якщо пароль містить символи верхнього регістру і неалфавітні символи, то його ентропія збільшується на 6 біт.

Таблиця 1.1

Значення ентропії для деяких довжин паролів

довжина пароля	**Цифри (10) (1)**	**Латинські букви без урахування регістра (26)**	**Цифри і латинські букви без урахування регістра (36)**	**Латинські букви з урахуванням регістру (52)**	**Цифри і латинські букви з урахуванням регістру (62)**	**Цифри, латинські букви і спеціальні символи (96)**
6	19,9	28,2	31,0	34,2	35,7	39,5
7	23,3	32,9	36,2	39,9	41,7	46,1
8	26,6	37,6	41,4	45,6	47,6	52,7
9	29,9	42,3	46,5	51,3	53,6	59,3
10	33,2	47,0	51,7	57,0	59,5	65,8
11	36,5	51,7	56,9	62,7	65,5	72,4
12	39,9	56,4	62,0	68,4	71,5	79,0

Стійкість того чи іншого пароля повинна розглядатися тільки в контексті конкретної системи парольної аутентифікації: пароль, який є стійким для однієї системи, може виявитися абсолютно не стійким при використанні іншої. Це відбувається через те, що різні системи в різному ступені реалізують (або зовсім не реалізують) механізми протидії атакам, спрямованим на злом паролів, а також тому, що деякі системи містять помилки або використовують ненадійні алгоритми.

Основний спосіб протидіяти злому паролів - штучно уповільнити процедуру їх перевірки. Дійсно, чи займе перевірка 10 наносекунд або 10 мілісекунд - для користувача різниця буде зовсім непомітною, а з точки зору злому швидкість впаде дуже істотно - з 100 мільйонів до 100 паролів в секунду. Уповільнення зазвичай досягається за рахунок багаторазового обчислення криптографічних функцій, причому ці обчислення побудовані таким чином, щоб атакуюча сторона не могла перевірити пароль без повторення обчислень (тобто недостатньо просто додати виклик *Sleep* в процедурі перевірки пароля). Вперше такий варіант був запропонований в 1997 році в роботі «*Secure Applications of Low-Entropy Keys*».

Системи, що реалізують подібні уповільнення, збільшують ефективну ентропію пароля на величину $log_2 C$, Де *C* - кількість ітерацій при обчисленні криптографічного перетворення. Приклади подібних систем наведені в таблиці 1.2.

Таблиця 1.2

Системи, що збільшують ефективну ентропію пароля

де застосовується	Кількість ітерацій, алгоритм	Змін. ефективної ентропії пароля
WPA	4*x*4096 *SHA*-1 (*PBKDF*2)	+14 біт
MS Office 2007	50 000 *SHA*-1	+15,6 біт
WinRAR 3.0 +	~ 100 000 *SHA*-1	+16,6 біт
PGP 9.0+	2*x*1024 *SHA*-1	+11 біт
Adobe Acrobat 5.0 - 8.0	50 *MD*5 + 20 *RC*4	+6,1 біт

Як було сказано вище, системи можуть містити помилки, що знижують її стійкість. Наприклад, для шифрування даних може використовуватися алгоритм з ключем недостатньої довжини або сама процедура перетворення пароля може бути ненадійною. Так, всі версії *Microsoft Word* і *Excel*, починаючи з 97 і до 2003 включно, для шифрування документів за замовчуванням використовують потоковий алгоритм *RC*4 з довжиною ключа 40 біт.

Стійкість системи визначається стійкістю найслабшої ланки, тому якщо відомо, що для шифрування використовується 40-бітний ключ, немає ніякого сенсу використовувати пароль з більшою ентропією. Інший, що став уже класичним, приклад - алгоритм перевірки паролів в операційній системі *Windows NT* / 2000 / *XP* / 2003. Ці системи підтримують два алгоритму - *LM* і *NTLM*. При створенні користувача або зміні його пароля ці системи обчислюють *LM*- і *NTLM*-хеші пароля і зберігають його в базі даних облікових записів.

Алгоритм обчислення *LM*-хеша виглядає наступним чином:

- символи пароля приводяться до верхнього регістру і перетворюються в *OEM*-кодування;
- якщо пароль коротше 14 символів, він доповнюється пробілами до цієї величини;

 отримана послідовність з 14 символів ділиться на дві частини по 7 символів;

 кожна з отриманих частин використовується в якості ключа для шифрування фіксованого значення алгоритмом *DES*;

 отримані в результаті шифрування блоки об'єднуються - це і є *LM*-хеш пароля.

1.2. Дослідження деяких алгоритмів оцінки стійкості пароля

1.2.1 Аналізатор паролів *SeaMonkey*

Цей аналізатор паролів розроблений як частина проекту*SeaMonkey*-вільного набору програм для роботи в *Internet*, створеного і підтримуваного організацією *Seamonkey Council*, яка виділяється з *Mozilla Foundation*. Сам механізм аналізу пароля є частиною*JavaScript* бібліотеки по роботі з паролями. Алгоритм його роботи полягає в обчисленні ваги пароля, що грунтується на даних про символи, з яких цей пароль складено. Вага пароля обчислюється за такою формулою:

$$pwstrengt = \big((pwlength * 10) - 20\big) + (numeric * 10) + \\ +(nymsymbols * 15) + (upper * 10), \quad (2)$$

де:

– *pwlength* дорівнює 5, якщо кількість символів в паролі більше 5, або дорівнює довжині пароля;

– *numeric* дорівнює 3, якщо кількість цифр в паролі більше 3, в іншому випадку - дорівнює кількості цифр;

– *numsymbols* вважається рівним 3, якщо число символів в паролі, відмінних від букв, цифр і знаків підкреслення, більше 3, інакше - кількістю таких символів;

– *upper* дорівнює 3, якщо кількість букв у верхньому регістрі більше 3, або кількістю великих літер в іншому випадку.

Після обчислення вага пароля нормується таким чином, щоб його значення полягало в інтервалі від 0 до 100. Нормування проводиться в тому випадку, коли значення ваги не потрапляє в цей діапазон. У разі, коли *pwstrength* менше 0, значення *pwstrength* прирівнюють до нуля, а коли більше 0, значення ваги встановлюють рівним 100. Ранжування ж пароля за ступенем стійкості залишено на розсуд розробників.

Описаний аналізатор не використовує ніяких перевірок з використанням словників, що робить його оцінки кілька однобокими, і, ймовірно, менш точними.

1.2.2. *Password Strength Meter* (*jQuery plugin*)

Ще одним варіантом оцінювача пароля, що працює на стороні клієнта, є *Password Strength Meter* - плагін, розроблений для *JavaScript* фреймворку *jQuery*.

Процедура оцінки працює наступним чином. Відомо безліч якостей, володіючи якими пароль збільшує або зменшує свою стійкість до підбору. Кожне таке якість має свій строго певну вагу. Алгоритм полягає в поетапної перевірки наявності у пароля цих якостей і, в разі їх присутності відбувається збільшення сумарної ваги пароля, за величиною якого після перегляду всіх характеристик робиться висновок про рівень стійкості пароля.

Розглянемо повний алгоритм процедури оцінки пароля:

1. Вага пароля встановлюється рівним нулю.
2. Якщо довжина пароля менше 4 символів, то робота алгоритму закінчується і повертається результат "занадто короткий пароль". Інакше переходимо до кроку 3.
3. Вага пароля збільшуємо на величину $4 * len$, Де *len* - довжина пароля.
4. Здійснюється спроба стиснення пароля за наступним алгоритмом. Якщо в паролі зустрічається підрядок виду *SS*, де *S* - рядок довжини 1, то перша частина цієї подстроки видаляється і стиснення триває з позиції початку другої

частини цієї підрядка. Наприклад, застосовуючи цей алгоритм до рядка *aaabbcab*, на виході отримаємо рядок *abcab*. Після виконання операції стиснення вага пароля зменшується на величину$len - lenCompress$, Де *len* - довжина пароля, а *lenCompress* - довжина пароля після стиснення.

5. Проводяться спроби стиснення пароля для випадків рядків *S* довжиною 2, 3 і 4 символів. Вага пароля зменшується аналогічно на величину$\text{len} - \text{lenCompress}$.

Відзначимо, що стиснення кожен раз проводиться на підприємстві, що перевіряється паролі, а не рядках, отриманих на попередніх спробах.

6. Якщо пароль містить не менше 3 цифр, то збільшити вагу на 5.

7. Якщо пароль містить не менше 2 знаків, то збільшити вагу на 5.

8. Якщо пароль містить букви, як у верхньому, так і в нижньому регістрах, то збільшити вагу пароля на 10.

9. Якщо пароль містить букви і цифри, то збільшити вагу пароля на 15.

10. Якщо пароль містить знаки і цифри, то збільшити вагу на 15.

11. Якщо пароль містить букви і знаки, то збільшити вагу на 15.

12. Якщо пароль складається тільки з букв або тільки з цифр, то зменшити вагу пароля на 10.

13. Якщо вага пароля менше 0, то встановити його рівним 0. Якщо більше 100, то встановити рівним 100.

14. Пароль, вага якого менше 34, визнається "слабким". Якщо вага від 34 до 67, то пароль відноситься до категорії "хороший", а якщо більше 67, то пароль вважається "відмінним".

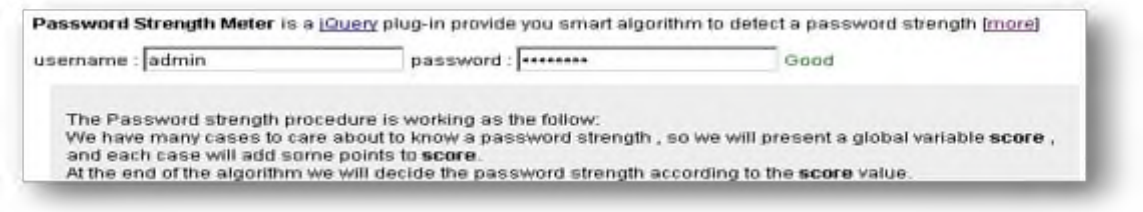

Рис. 1.1 - Приклад оцінки стійкості пароля з використанням *Password Strength Meter*

1.2.3. *Cornell University - Password Strength Checker*

Офіційний*on-line* сервіс, Що надається центром безпеки Корнельського університету (Ітака, США). З його допомогою користувачі можуть перевірити свій пароль, заповнивши *web*-форму і відправивши його на перевірку. Оцінка пароля, як і в випадку з сервісом *Google*, проводиться на стороні сервера.

Реалізація алгоритму не розкрита для загального доступу, проте в описі сервісу вказані вимоги, яким повинен задовольняти пароль, щоб перевірка пройшла успішно:

1. Пароль повинен мати довжину не менше 8 символів.
2. При складанні пароля використовуються символи, по крайней мере, трьох алфавітів з наступного списку.

– великі латинські літери;
– рядкові латинські літери;
– цифри;
– спеціальні знаки (такі як! * (): |);

3. пароль не повинен містити слів з словника;
4. пароль не повинен містити послідовностей повторюваних букв (наприклад, AAA) і послідовностей виду *abc*, *qwerty*, 123, 321.

Ці вимоги повинні строго виконуватися. Якщо хоча б якесь вимога не виконується, то пароль визнається ненадійним.

1.2.4. *Password Strength Tester*

JavaScript аналізатор паролів, який розробляється і підтримується в рамках проекту *Rumkin.com*.

Алгоритм оцінки, реалізований в даному аналізаторі, ґрунтується на загальних положеннях теорії інформації. В якості основної оцінки пароля використовується його ентропія, обчислення якої проводиться з використання таблиць діаграм для англійської мови.

Під ентропією (інформаційної ємністю) пароля розуміється міра випадковості вибору послідовності символів, складових пароль, оцінена методами теорії інформації.

Інформаційна ємність *E* вимірюється в бітах і характеризує стійкість до підбору пароля методом повного перебору за умови відсутності апріорної інформації про характер пароля і застосуванні зловмисником оптимальної стратегії перебору, при якій середнє очікуване кількість спроб до настання вдалою дорівнює2^{E-1}. За твердженням творця цього оцінювача, з метою зменшення завантажується на клієнтську сторону обсягу інформації все небуквених символи були об'єднані в одну групу. Ця група виступає таким собі універсальним символом, який і використовується в частотної таблиці.

Як зазначає розробник, при цьому допущенні значення одержуваної ентропії буде менше, ніж у випадку, коли в частотної таблиці все символи представлені окремо.

Залежно від отриманого значення ентропії паролю присвоюється відповідна характеристика його стійкості.

Таблиця 1.3

Залежність рівня стійкості пароля від значення ентропії

ентропія	**рівень стійкості**	**коментар**
<28 біт	Дуже слабкий	Припустимо захищати тільки не цінну інформацію.
28-35 біт	слабкий	Здатний зупинити велике число початківців зломщиків, ідеально підходить для використання в якості *desktop*-пароля.
36-59 біт	середній	Цілком придатний для комп'ютерів в корпоративній мережі.
60-127 біт	високий	Може бути хорошим, щоб охороняти фін. інформацію
> 128 біт	наднадійною	Пароль має дуже великою стійкістю до підбору.

В плагіні *Password Strength Meter* для *jQuery* при оцінці здійснюється виявлення паролів побудованих шляхом повторення груп символів. Особливо слід відзначити проект *Password Strength Tester*, в якому використовується математична модель, побудована на основі положень теорії інформації.

Необхідно сказати, що в розглянутих програмних рішеннях існують ряд проблем, що не дозволяють вважати оцінку рівня надійності пароля повній. Так *on-line* аналізатори, реалізовані посредствам сценаріїв *JavaScript*, виконуваних на стороні клієнта, обмежені в своїх ресурсах, що виключає можливість здійснювати перевірку за словниками великих обсягів. У той же час варіант перевірки пароля на стороні сервера хоча і вирішує проблему використання словників, однак залишає відкритим питання про надійність передачі і, головне, захищеності оцінюваного пароля від несанкціонованого використання. В останньому випадку користувач може сподіватися лише на те, що обробку пароля проводить сервер, що виконує лише «чисті» операції, а не службовець постачальником інформації для зловмисників.

Крім цього, слід зазначити, що всі розглянуті програмні продукти не підтримують оцінку паролів, що містять букви кирилиці.

Рис. 1.2 - Приклад оцінки стійкості пароля з використанням *Password Strength Tester*

1.3. Алгоритм оцінки стійкості пароля від *Microsoft*

стійкість пароль аналізатор

Аналіз коду показав, що алгоритм використовує два аналізатора стійкості пароля - основний, досить спрощений, і допоміжний - глибший.

Почнемо з поточного алгоритму, за яким аналізатор працює в черговому режимі.

1. Рівень стійкості пароля дається в діапазоні [0; 4] в залежності від обчислюється «бітової стійкості» *bits*.

– *bits* ≥ 128 - 4;

– 128 <*bits* ≥ 64 - 3;

– 64 <*bits* ≥ 56 - 2;

– *bits* <56 - 1;

– порожній пароль - 0.

2. Для оцінки «бітової стійкості» використовується формула (3).

$$bits = \log(charset) * \left(\frac{length}{\log(2)}\right), \quad (3)$$

де

– *bits* -бітова стійкість;

– *log* -натуральний логарифм;

– *length* -довжина пароля;

– *charset* -сумарний розмір множин для кожного з типів нижче, якщо вони присутні в рядку.

1. Малі англійські букви [*abcdefghijklmnopqrstuvwxyz*].
2. Заголовні англійські букви [*AZ*].
3. Спеціальні символи [~ ` ! @ # $% ^ & * () -_ + ="].
4. Цифри [1234567890].
5. Решта символів.

Розглянемо алгоритм роботи допоміжного аналізатора.

Для аналізу стійкості використовуються наступні показники:

– довжина пароля;

– кількість використовуваних типів символів: заголовні латинські, малі латинські, цифри, спец-символи;

– відсутність слова в словнику (слова довжиною від 3 до 16 символів) з урахуванням подібності символів;

– близькість слова до словникового слова з певною ймовірністю;

Стійкість пароля підрозділяється на п'ять класів:

1. Відмінний (*Best*).

– довжина пароля - не менше 14;

– кількість використовуваних різних типів символів - не менше 3;

– в словнику відсутня близьке з ймовірністю 0.6 слово з урахуванням подібності символів.

2. Стійкий (*Strong*).

– довжина пароля - не менше 8;

– кількість використовуваних різних типів символів - не менше 3;

– в словнику відсутня близьке з ймовірністю 0.6 слово з урахуванням подібності символів.

3. Середній (*Medium*).

– довжина пароля - не менше 8;

– кількість використовуваних різних типів символів - не менше 2;

– слово відсутнє в словнику з урахуванням подібності символів;

4. Слабкий (*Weak*).

– пароль складається як мінімум з одного довільного символу.

5. Ніякий.

– всі інші паролі.

Таблиця 1.3

Подоба символів:

'3'	↔	'*E*'		'9'	↔	'*G*'
'*X*'	↔	'*K*'		'+'	↔	'*T*'
'5'	↔	'*S*'		'@'	↔	'*A*'
'$'	↔	'*S*'		'0'	↔	'*O*'
'6'	↔	'*G*'		'1'	↔	'1'
'7'	↔	'*T*'		'2'	↔	'*Z*'
'8'	↔	'*B*'		'!'	↔	'*I*'
'|'	↔	'1'				

1.4. Висновки до розділу

У розглянутих вищепрограмахвикористовуються різні способи оцінки паролів, причому в кожній з них акцент зроблений на оцінку лише якогось певного властивості пароля. Так аналізатор паролів для проекту *SeaMonkey* в якості основи для оцінки використовує знання про довжину пароля і кількості символах з того чи іншого алфавіту, використовуваних при побудові пароля.

Розглянуті алгоритми оцінки стійкості пароля дозволяють з якого-небудь певною ознакою оцінити стійкість пароля до злому. Для цього кожним алгоритмом оцінки стійкості використовуються свій комплекс процедур, званий аналізатором в алгоритмі, який спрямований на вивчення самого пароля, відповідність пароля будь-яким поширеним «заготівлях». Незалежно від того, чи використовується для порівняння база даних найпоширеніших паролів, звана словником, до паролю пред'являються досить жорсткі вимоги, такі, як наявність цифр, великих і малих літер, спеціальних символів, знака підкреслення, а так же фіксована довжина - починаючи з 8 символів (гарантує досить низьку оцінку з боку аналізатора).

Незважаючи на достатню кількість алгоритмів і аналізаторів, лише одиниці з них надають клієнту можливість допустимості генерації паролів, що містять кирилицю. Зміст символів різних шрифтів сприяє багаторазовому збільшенню стійкості пароля, так само як його ентропії.

РОЗДІЛ 2
КРИПТОГРАФІЧНІ МЕТОДИ ПЕРЕВІРКИ СТІЙКОСТІ ПАРОЛІВ ДО ЗЛАМУ

Алгоритм *MD*5 є 128-бітний алгоритм хешування. Це означає, що він обчислює 128-бітний хеш для довільного набору даних, що надходять на його вхід. Цей алгоритм розробив професор Рональд Ривест з Массачусетського технологічного інституту в 1991 році для заміни менш надійного попередника - *MD*4. Алгоритм був вперше опублікований в квітні 1992 року в *RFC* 1321. Після цього *MD*5 став використовуватися для вирішення самих різних завдань, від хеширования паролів в *CMS* до створення електронно-цифрових підписів та *SSL*-сертифікатів.

Про те, що алгоритм *MD*5 можна зламати, вперше заговорили в 1993 році. Дослідники Берт ден Боєр і Антон Боссіларіс показали, що в алгоритмі можливі псевдоколлізії. Через три роки, в 1996-му, Ганс Доббертін опублікував статтю, в якій довів наявність колізій і описав теоретичну можливість злому *MD*5. Це був ще не злом, але в світі почалися розмови про необхідність переходу на більш надійні алгоритми хешування, наприклад *SHA*1 (на момент написання цієї статті вже було доведено, що колізії є і в цьому алгоритмі, тому рекомендую використовувати *SHA*2) або *RIPEMD*-160.

2.1. Аналіз перших атак на злам паролів

Безпосередній злом *MD*5 почався 1 березня 2004 року. Компанія *CertainKey Cryptosystems* запустила проект *MD*5*CRK* - розподілену систему пошуку колізій. Метою проекту був пошук двох повідомлень з ідентичними хеш-кодами. Проект завершився 24 серпня 2004 року, коли чотири незалежних дослідника - Ван Сяоюн, Фен Денгуо, Лай Сюецзя і Юй Хунбо - виявили уразливість алгоритму, що дозволяє знайти колізії аналітичним методом за більш-менш прийнятний час. За допомогою цього методу можна всього лише за годину виявити колізії на

кластері *IBM p*690 (шкода, що у мене немає такого будинку). Першого березня 2005 року була продемонстровано перше використання зазначеної уразливості на практиці. Група дослідників представила два сертифікати *X*.509 з різними наборами ключів, але з ідентичними контрольними сумами. У тому ж році Властіміл Клима опублікував алгоритм, що дозволяє виявляти колізії на звичайному ноутбуці за кілька годин. У 2006 він пішов далі. Вісімнадцятого березня 2006 року дослідник оприлюднив алгоритм, що знаходить колізії за одну хвилину! Цей метод отримав назву «туннелирование». У 2008 році на конференції *Chaos Communication Congress* була представлена стаття про метод генерації підроблених сертифікатів *X*.509. Фактично це був перший випадок реального використання колізій в алгоритмі *MD*5.

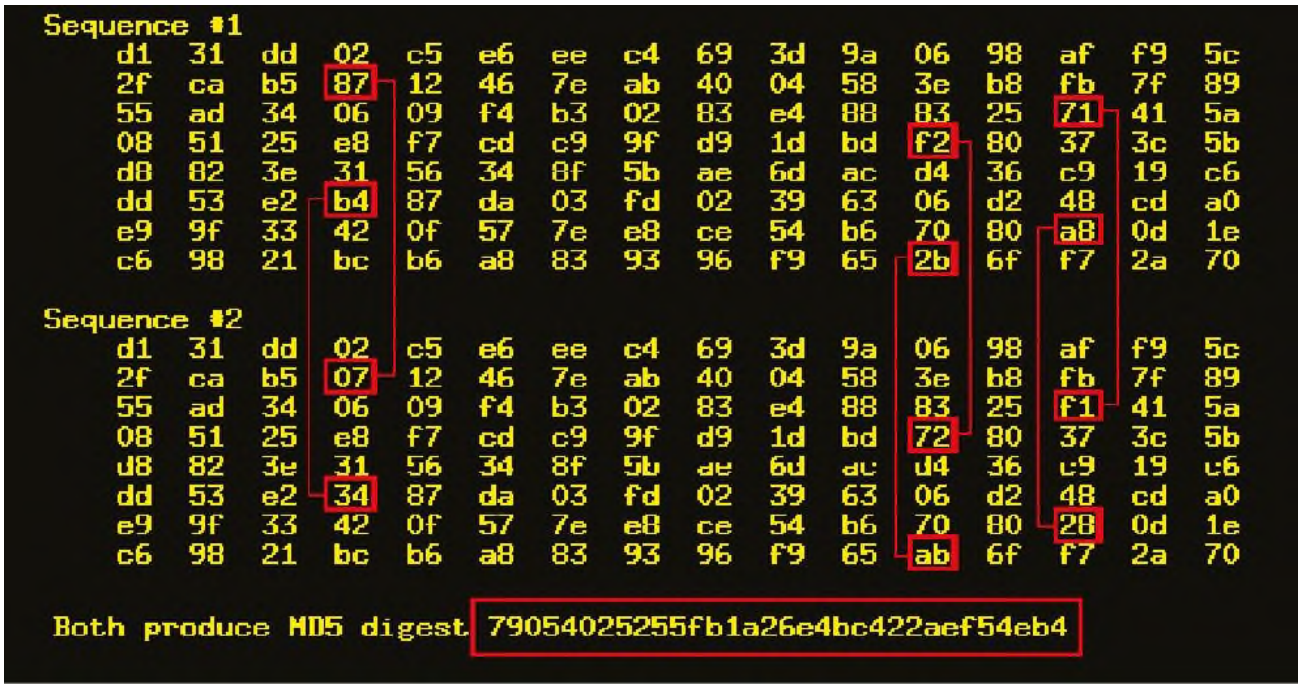

Рис. 2.1. Приклад колізії *MD*5-хеш-кодувань

Велика робота була також проведена і для прискорення злому хешей. У 2007 році Кевін Бриз представив програму, яка використовує *Sony PlayStation*3 для злому *MD*5. Він зумів домогтися дуже непоганих результатів: 1,4 мільярда *MD*5-хеш-кодувань генерувалися всього лише за одну секунду! Уже через два роки, в 2009-му, на *BlackHat USA* вийшла стаття про використання *GPU* для

пошуку колізій, що дозволяло підвищити його швидкість в кілька разів, особливо якщо він виконувався за допомогою декількох відеокарт одночасно.

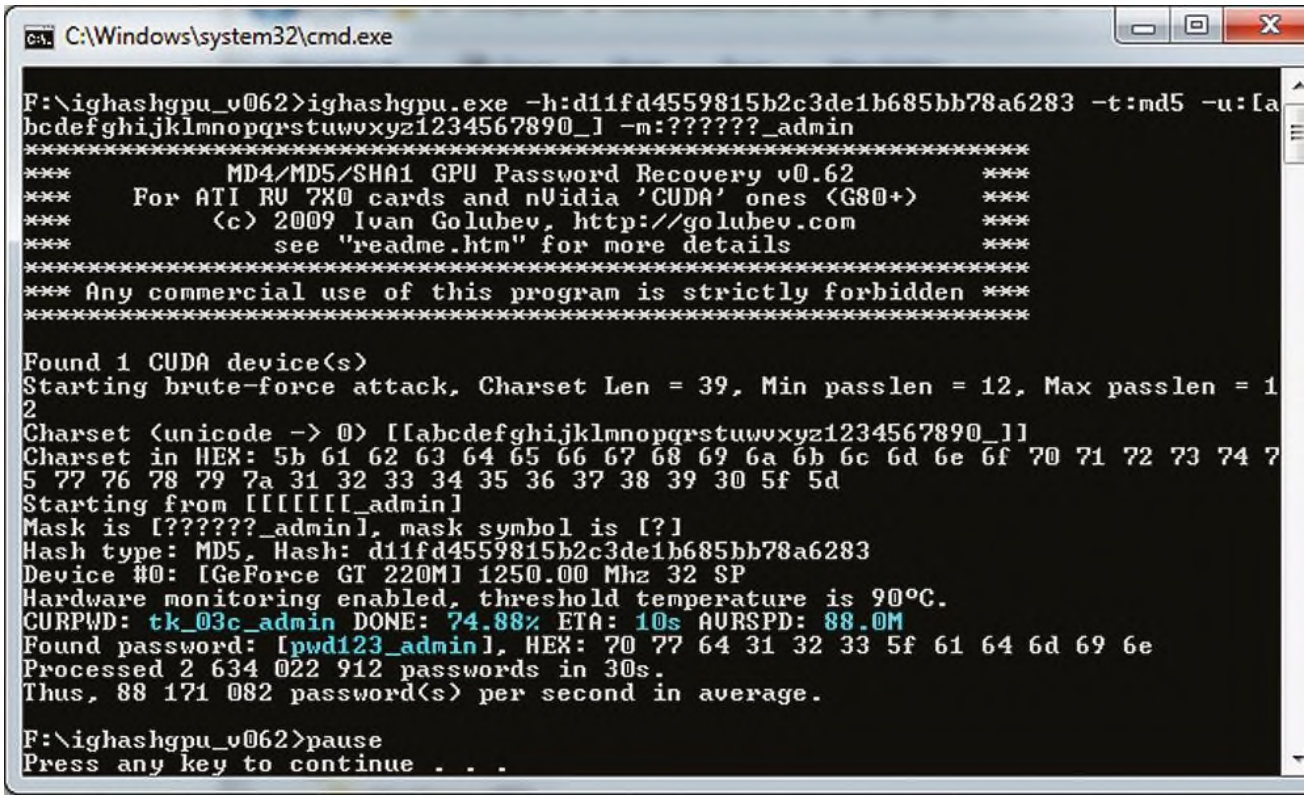

Рис. 2.2. Метод Брут *MD*5 по масці

У 2011 році *IETF* погодився внести зміни в *RFC* 1321 (*MD*5) і *RFC* 2104 (*HMAC-MD*5). Так з'явився документ *RFC* 6151. Він визнає алгоритм шифрування *MD*5 небезпечним і рекомендує відмовитися від його використання. На мій погляд, цей документ офіційно поклав край *MD*5. Однак, незважаючи на те що алгоритм *MD*5 був офіційно визнаний небезпечним, існують тисячі, якщо не десятки і сотні тисяч додатків, які використовують його для зберігання паролів, в електронно-цифрові підписи і для обчислення контрольних сум файлів. До речі, 31 жовтня 2008 року *NIST* оголосила конкурс серед криптографов. Мета конкурсу - розробити алгоритм хешування на заміну застарілим *SHA*1 і *SHA*2. На даний момент фіналісти вже визначені - це *BLAKE, Gostl, JH, Keccak* і *Skein*.

2.2. *Ighashgpu*: злом за допомогою *GPU*

Але вистачить теорії. Давай перейдемо до справи і поговоримо безпосередньо про злом нашого улюбленого алгоритму. Припустимо, що нам в

руки потрапив хеш якогось пароля: *d*8578*edf*8458*ce*06*fbc*5*bb*76*a*58*c*5*ca*4. Для злому цього хешу я пропоную скористатися програмою *Ighashgpu*, яку можна завантажити на сайті*www.golubev.com*або знайти на нашому диску. Утиліта поширюється абсолютно безкоштовно і спокійно працює під виндой. Щоб прискорити процес злому хешу, *Ighashgpu* використовує *GPU*, тому тобі необхідна як мінімум одна відеокарта *nVidia* або *ATI c* підтримкою *CUDA* / *ATI Stream*. Сучасні графічні процесори побудовані на дещо інший архітектурі, ніж звичайні *CPU*, тому вони набагато ефективніше обробляють графічну інформацію. Хоча *GPU* призначені для обробки тривимірної графіки, в останні кілька років з'явилася тенденція до їх застосування та для звичайних обчислень. Почати працювати з програмою не просто, а дуже просто: розпакуйте архів в будь-яке місце на диску і приступай до злому за допомогою командного рядка *Windows*:

ighashgpu.exe -t:*md*5 \

-h:*d*8578*edf*8458*ce*06*fbc*5*bb*76*a*58*c*5*ca*4 *-max*:7

Ми використовуємо вищенаведений спосіб для злому одного певного хешу, згенерованого за допомогою алгоритму *MD*5. Максимальна довжина можливого пароля становить сім символів. Через якийсь час пароль буде знайдений (*qwerty*). Тепер давай спробуємо зламати ще один хеш, але з трохи іншими умовами. Нехай наш хеш має вигляд *d*11*fd*4559815*b*2*c*3*de*1*b*685*bb*78*a*6283, а включає в себе літери, цифри, знак підкреслення і має суфікс «*_admin*». В даному випадку ми можемо використовувати перебір пароля по масці, щоб спростити програмі завдання:

ighashgpu.exe -h:*d*11*fd*4559815*b*2*c*3*de*1*b*685*bb*78*a*6283 *-t*:*md*5

-u: [*abcdefghijklmnopqrstuwvxyz*1234567890_] *-m*: ??????*_admin*

Тут параметр '*-u*' дозволяє вказати набір символів, які використовуються при переборі, а параметр '*-m*' задає маску пароля. У нашому випадку маска складається з шести довільних символів, після яких йде поєднання «*_admin*». Підбір пароля також не складе ніяких труднощів.

Колізією в криптографії називають два різних вхідних блоку даних, які для однієї і тієї ж хеш-функції дають один і той же хеш. Кожна функція на виході дає

послідовність бітів певної довжини, яка не залежить від розміру початкових даних. Звідси випливає, що колізії існують для будь-якого алгоритму хешування. Однак імовірність того, що ти зможеш знайти колізію в «хорошому» алгоритмі, практично наближається до нуля. На жаль чи на щастя, алгоритми хешування можуть містити помилки, як і будь-які програми. Багато хеш-функції або вже були зламані, або скоро будуть. В даному випадку «зламати» - значить знайти колізію за час, який багато менші, ніж заявлена нескінченності.

2.2. *Ighashgpu*: списки

Тепер спробуємо зламати відразу кілька паролів одночасно. Припустимо, що до нас в руки потрапила база даних хешів паролів. При цьому відомо, що кожен пароль закінчується символами *c*00*l*:

*f*0*b*46*ac*8494*b*7761*adb*7203*aa*7776*c*2*a*

*f*2*da*202*a*5*a*215*b*66995*de*1*f*9327*dbaa*6

*c*7*f*7*a*34*bbe*8*f*385*faa*89*a*04*a*9*d*94*dacf*

*cb*1*cb*9*a*40708*a*151*e*6*c*92702342*f*0*ac*5

00*a*931*d*3*facaad*384169*ebc*31*d*38775*c*

4966*d*8547*cce*099*ae*6*f*666*f*09*f*68458*e*

Збережи хеши в файлі *encrypted.dat* і запусти *Ighashgpu* як зазначено нижче:

ighashgpu.exe -t:*md*5 *-u*: [*abcdefghijklmnopqrstuwvxyz*1234567890_]

-m: ??????*c*00*l encrypted.dat*

Після завершення роботи програми в папці *Ighashgpu* з'явиться файл *ighashgpu_results.txt* зі зламаними паролями:

*f*0*b*46*ac*8494*b*7761*adb*7203*aa*7776*c*2*a*:1*rootxc*00*l*

*f*2*da*202*a*5*a*215*b*66995*de*1*f*9327*dbaa*6:*pwd*12*xc*00*l*

*c*7*f*7*a*34*bbe*8*f*385*faa*89*a*04*a*9*d*94*dacf*:*pwd*34*yc*00*l*

*cb*1*cb*9*a*40708*a*151*e*6*c*92702342*f*0*ac*5:*pwd*56*yc*00*l*

4966*d*8547*cce*099*ae*6*f*666*f*09*f*68458*e*:*pwd*98*zc*00*l*

00*a*931*d*3*facaad*384169*ebc*31*d*38775*c*:*pwd*78*zc*00*l*

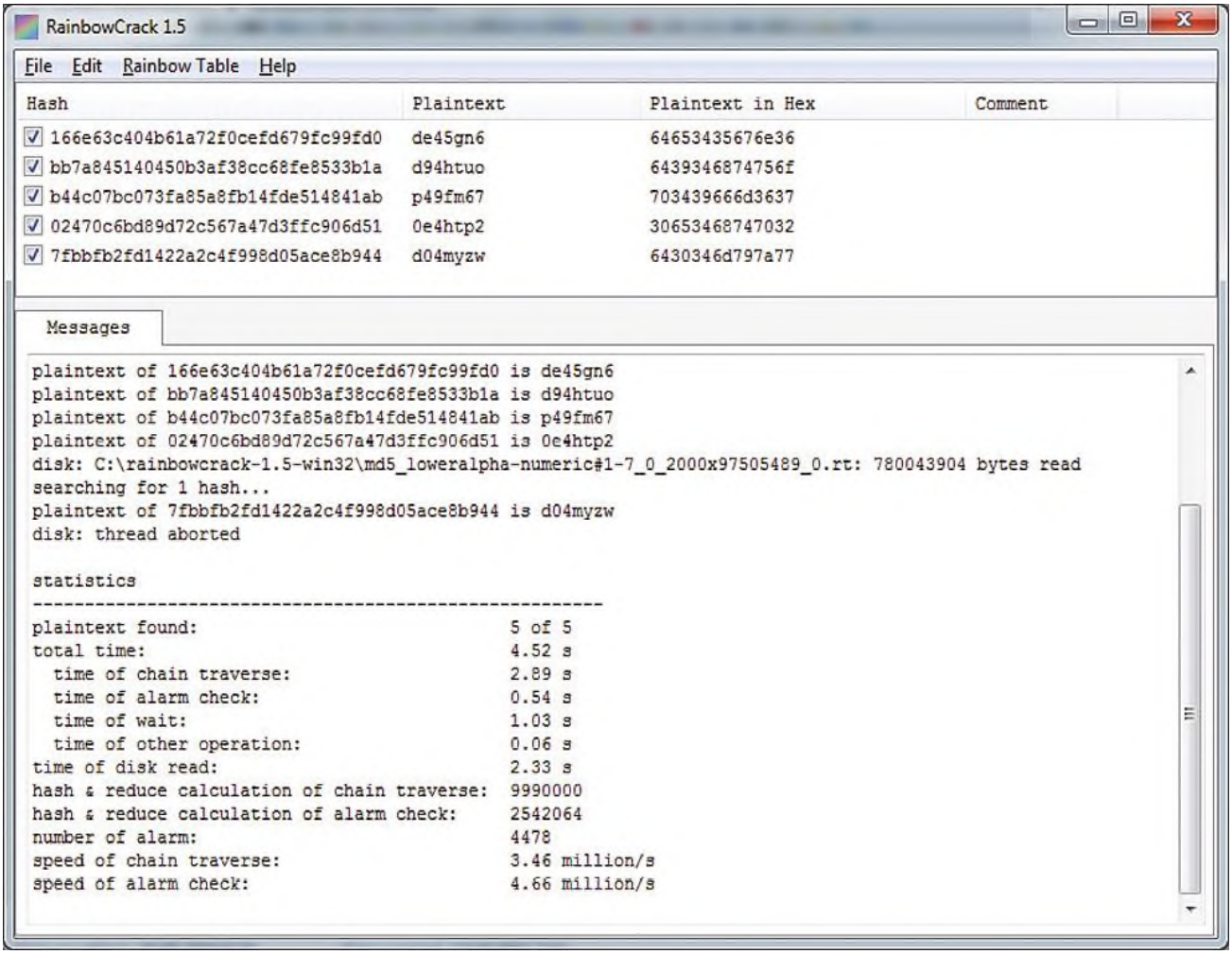

Рис 2..3. Взломание хеші з файлу *encrypted.dat*

Наостанок давай зробимо злом «підсоленого» хешу. Припустимо, що хеш генерується за наступним алгоритмом:

var plain = password + "*S*41*t*";

*var hash = md*5(*plain*);

В результаті ми отримали такий хеш: 42151*cf*2*ff*27*c*5181*bb*36*a*8*bcfafea*7*b*. *Ighashgpu* дозволяє вказувати «сіль» в параметрі «*-asalt*»:

ighashgpu.exe -h:42151*cf*2*ff*27*c*5181*bb*36*a*8*bcfafea*7*b* \

-t:*md*5 *-u*: [*abcdefghijklmnopqrstuwvxyz*1234567890_] \

-asalt:*s*41*t*

І ми знову отримали шуканий пароль легко і швидко.

Для 8-символьного пароля, складеного з перших 126 символів *ASCII*, є 63 527 879 748 485 376 можливих комбінацій. Для 254 символів кількість можливих комбінацій зростає до 17 324 859 956 700 833 536, що аж в 2,7 мільярда разів більше, ніж людей на нашій планеті. Якщо створити текстовий файл, який містить всі ці паролі, то він займе мільйони терабайт. Звичайно, в сучасному світі це можливо, але вартість зберігання такого файлу буде просто захмарною.

2.3. Способи зламу *MD*5 в режимі турбо

Злом хешів шляхом повного перебору навіть на самому кращому залозі займає досить багато часу, особливо якщо пароль більше восьми символів. Найпростіший спосіб збільшити швидкість підбору пароля - це створити базу даних всіх хешів для певного набору символів. У 80-х роках минулого століття хакери вважали, що коли у них з'явиться більше потужне залізо, 640 Кб пам'яті і жорсткий диск розміром в 10 Мб, то така база стане реальністю і підбір будь-якого пароля перетвориться в хвилинна справа. Однак залізо розвивалося, а мрія так і залишалася мрією. Ситуація змінилася лише в серпні 2003 року, після того, як Філіп Оешлін, доктор філософії в галузі комп'ютерних мереж з Швейцарського технологічного інституту в Лозанні, опублікував свою роботу про проблему вибору оптимального співвідношення місце-час. У ній описувався метод злому хеш-функцій за допомогою «райдужних» таблиць. Суть нового методу полягає в наступному. Спочатку необхідно вибрати довільний пароль, який потім хешіруется і піддається впливу функції редукції, перетворюючої хеш в будь-якої можливий пароль (наприклад, це можуть бути перші 64 біта вихідного хешу). Далі будується ланцюжок можливих паролів, з якої вибираються перший і останній елементи. Вони записуються в таблицю. Щоб відновити пароль, застосовуємо функцію редукції до вихідного хешу і шукаємо отриманий можливий пароль в таблиці. Якщо такого пароля в таблиці немає, хешіруем його і обчислюємо наступний можливий пароль. Операція повторюється, поки в «райдужної» таблиці не буде знайдений пароль. Цей

пароль є кінець однієї з ланцюжків. Щоб знайти вихідний пароль, необхідно прогнати весь ланцюжок заново. Така операція не займає багато часу, в залежності від алгоритму побудови ланцюжка це зазвичай кілька секунд або хвилин. «Веселкові» таблиці дозволяють істотно скоротити обсяг використовуваної пам'яті в порівнянні зі звичайним пошуком. Єдиний недолік описаного методу полягає в тому, що на побудову таблиць потрібно досить багато часу.

Тепер перейдемо від слів до справи і спробуємо зламати пару-трійку хешів паролів за допомогою цього методу.

2.4. *Rainbow tables*

2.4.1. Використання «веселкових» таблиць

«Веселкові» таблиці - це особливий тип словника, який містить ланцюжка паролів і дозволяє підібрати пароль протягом декількох секунд або хвилин з ймовірністю 85-99%.

Спочатку необхідно визначитися з програмою. Особисто мені подобається*RainbowCrack,* Яка розповсюджується безкоштовно і працює як на *Windows*, так і на *Linux*. Вона підтримує чотири алгоритму хешування: *LN* / *NTLM, MD*5 і *SHA*1. Програма не вимагає установки, досить розпакувати її куди-небудь на диск. Після розпакування необхідно знайти «райдужні» таблиці для алгоритму *MD*5. Тут все не так просто: їх можна або завантажити безкоштовно, або купити, або згенерувати самостійно. Один з найбільших архівів безкоштовних таблиць доступний на сайті проекту*Free Rainbow Tables*. До речі, ти теж можеш допомогти проекту, якщо скачаєш клієнт з сайту і приєднаєшся до розподіленої міжнародної мережі, яка генерує «райдужні» таблиці. На момент написання статті на цьому сайті вже було доступно 3 Тб таблиць для алгоритмів *MD*5, *SHA*1, *LM* і *NTLM*. Якщо у тебе немає можливості злити такий обсяг інформації, то на тому ж сайті можна замовити диски з «райдужними»

таблицями. На даний момент пропонується три пакети: *LN* / *NTLM, MD*5 і *SHA*1 - по 200 доларів кожен. Ми ж сгенерируем таблиці самостійно. Для цього необхідно використовувати програму *rtgen,* що входить до складу *RainbowCrack*. Вона приймає такі вхідні параметри:

- *hash_algorithm* - алгоритм хешування (*LM, NTLM, MD*5 або *SHA*1);
- *charset* - один з наборів символів, що міститься у файлі *charset.txt*;
- *plaintextlenmin* і *plaintextlenmax* - мінімальна і максимальна довжина пароля;
- *tableindex, chainlen, chainnum* і *partindex* - «магічні числа», описані в статті Філіпа Оешлінa*bit.ly*/*nndT*8*M*.

Розглянемо останні параметри докладніше:

1. *table_index* - індекс «райдужної» таблиці, який можна використовувати при розбивці таблиці на кілька файлів. Я використовував 0, так як моя таблиця складалася всього з одного файлу.
2. *chain_len* - кількість унікальних паролів в ланцюжку.
3. *chain_num* - кількість ланцюжків в таблиці.
4. *part_index* - це параметр, що визначає початок ланцюжка. Творці програми просять використовувати в якості цього параметра тільки число (я використовував 0). Тепер запускаємо генерацію «райдужної» таблиці для *MD*5:

*rtgen.exe md*5 *loweralpha-numeric* 1 7 0 2000 97505489 0

В даному випадку ми створюємо таблицю паролів, що складаються з цифр і великих літер латинського алфавіту і мають довжину від одного до семи символів. На моєму *Eee PC* з процесором *Intel Atom N*450 цей процес зайняв майже два дня :). У підсумку я отримав файл *md*5*loweralpha-numeric* # 1-702000*x*975054890.*rt* розміром в 1,5 Гб.

Далі отриману таблицю необхідно відсортувати, щоб оптимізувати пошук потрібної нам ланцюжка. Для цього запускаємо *rtsort.exe*:

*rtsort.exe md*5*_loweralpha-numeric*# 1-7_0_2000*x*97505489_0.*rt*

Чекаємо пару хвилин і таблиця готова! Тепер можна ламати самі паролі. Для початку спробуємо підібрати пароль для одного хешу: *d*8578*edf*8458*ce*06*fbc*5*bb*76*a*58*c*5*ca*4. Запускаємо *rcrack_gui.exe* і вибираємо *Add*

Hash ... в меню *File*. У вікні вводимо хеш і натискаємо *OK*. Тепер вибираємо файл з «райдужної» таблицею. Для цього використовуємо пункт *Search Rainbow Tables* ... в меню *Rainbow Table*. У вікні для вибору файлу шукаємо файл з таблицею, у мене це *md*5_*loweralpha-numeric* # 1-7_0_2000*x*97505489_0.*rt*, потім тиснемо *Open*. Через кілька секунд пароль у нас в руках! Аналогічну операцію можна зробити і над списком хешів з файлу.

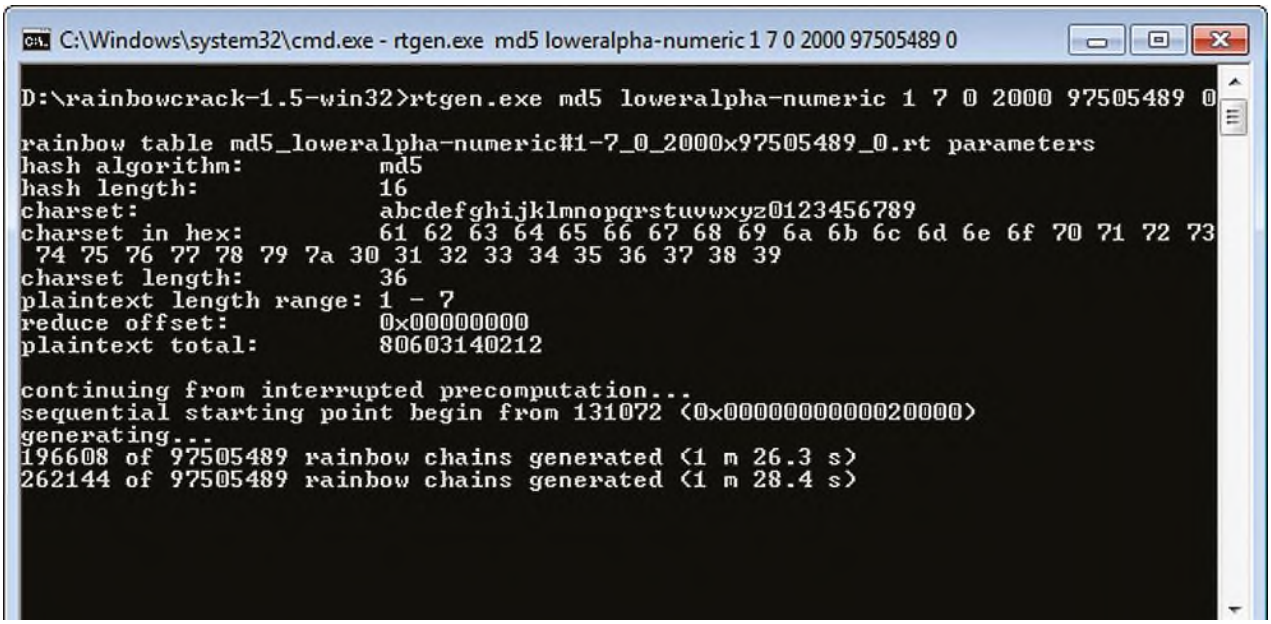

Рис. 2.4. Сгенерована «веселкова» таблиця

2.4.2. «Веселкові» таблиці *vs*. *CPU vs*. *GPU*

Я думаю, ти звернув увагу на те, наскільки швидко *Ighashgpu* здатний зламувати *MD*5-хеш-кодування повним перебором, і на те, що *RainbowCrack* робить це ще швидше при наявності хорошої «райдужної» таблиці. Я вирішив порівняти швидкість роботи цих програм. Для чистоти експерименту я використовував програму *MDCrack*, яка здійснює брут пароля на *CPU* (і є однією з кращих серед програм такого типу). Ось що вийшло в результаті для *GPU* (*nVidia GeForce GT* 220*M*), *CPU* (*Intel Atom N*450, два ядра) і «райдужних» таблиць:

довжина пароля | *GPU* | *CPU* | таблиці

4 символу | 00:00:01 | 00:00:01 | 00:00:16

5 символів | 00:00:02 | 00:00:09 | 00:00:16

6 символів | 00:00:16 | 00:05:21 | 00:00:10

7 символів | 00:07:11 | 09:27:52 | 00:00:04

Як бачиш, швидкість перебору з використанням *CPU* набагато менше, ніж з використанням *GPU* або «райдужних» таблиць. Більш того, більшість спеціалізованих програм дозволяє створити кластер з відеокарт, завдяки чому швидкість перебору пароля збільшується в рази. Я думаю, ти звернув увагу на те, що швидкість підбору 4- і 5-символьного паролів нижче, ніж швидкість підбору пароля з шести або семи символів. Це пов'язано з тим, що пошук пароля починається тільки після завантаження таблиці в пам'ять. Виходить, що з шістнадцяти секунд в середньому тринадцять витрачається на завантаження і три - на злом хешу.

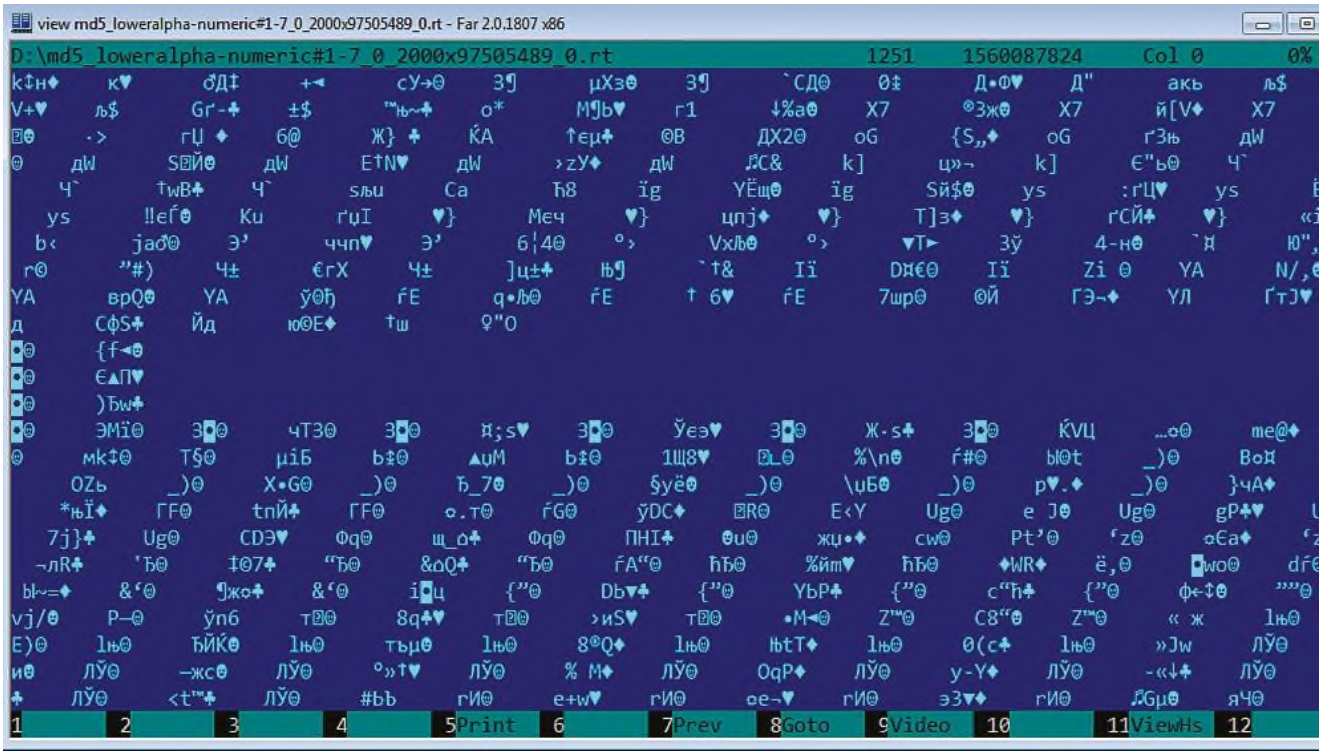

Рис. 2.6. Райдужна таблиця зсередини

2.5. Правила використання систем перевірки стійкості паролів

Щоб зрозуміти, що таке хороший пароль, подивимося, що відбувається в стані ворога!

Коли ви створюєте обліковий запис, сервіс зберігає пароль у одному з численних форматів. Сервіс може покласти пароль прямо в базу даних (у вигляді простого тексту) або згенерувати з нього хеш за допомогою одного з численних алгоритмів. Найпопулярніші:

- MD5
- SHA-1
- Bcrypt
- Scrypt
- Argon2

Перевага зберігання хеш замість самих паролів полягає в тому, що паролів в БД немає. І це правильно, тому що вам потрібно тільки довести, що ви знаєте свій пароль, але сам він не має значення. Коли ви логін, введений пароль хешіруется за допомогою того ж алгоритму, і якщо результат збігається з записаним в базі значенням, то ви довели, що знаєте пароль. А якщо базу зламають, то відновити паролі не вдасться.

Зберігання хеш.

Злом пароля

Злом пароля- це коли зловмисник намагається повернути назад хеш-функцію і відновити пароль з хеша. У випадку з хорошим алгоритмом хешування зробити це неможливо. Але ніщо не заважає зловмисникові пробувати вводити різні значення в надії отримати такий же хеш. Якщо збіг відбудеться, значить, пароль відновлений з хеша.

Злом пароля.

І тут важливий вибір вдалого алгоритму. SHA-1 розроблявся з урахуванням швидкого хешування, це полегшує життя хакерам. Bcrypt, Scrypt і Argon2 розроблялися з урахуванням високих обчислювальних витрат, щоб якомога більше сповільнити злом, особливо на спеціально виділених машинах. І це дуже важливий аспект.

Якщо орієнтуватися тільки на швидкість, то пароль SHA-1, який неможливо зламати, виглядає так:0OVTrv62y2dLJahXjd4FVg81.

А безпечний пароль, створений за допомогою правильно сконфигурированного Argon2, виглядає так:Pa $$ w0Rd1992.

Як бачите, вибір правильного алгоритму хешування перетворює слабкий пароль в що не піддається злому.

І не забувайте, що це залежить тільки від реалізації сервісу, в якому ви входите. І ви не можете дізнатися якість реалізації. Запитати можна, але вам або не дадуть, або одпишуть, що «ми серйозно підходимо до безпеки».

Думаєте, в компаніях серйозно ставляться до безпеки і використовують хороші алгоритми хешування? Подивіться насписокзламаних баз даних, особливо на використані в них хеші. У багатьох випадках застосовувався MD5, найчастіше - SHA-1, і де-не-де використовували bcrypt. Деякі зберігали паролі у вигляді простого тексту. Така реальність, яку потрібно враховувати.

Причому ми знаємо лише, які хеші використовувалися під зламаних базах, і

висока ймовірність, що компанії, які застосовували слабкі алгоритми, не змогли захистити і свою інфраструктуру. Погляньте на список, впевнений, ви знайдете знайомі назви. Те, що компанія виглядає великий і респектабельної, ще не означає, що все робить правильно.

Пароль вибираєте ви

Що ви можете зробити як користувач?Якщо паролі зберігаються у вигляді простого тексту, тоді нічого не поробиш. Коли базу вкрадуть, складність вашого пароля не матиме значення.

При правильно сконфігурованих алгоритмах складність вашого пароля теж не важлива, він може бути12345абоasdf.

Однак в проміжних випадках, особливо при використанні SHA-1, складність пароля має значення. Функції хешування в цілому не призначені для паролів, але якщо ви використовуєте складний пароль, то це компенсує недоліки алгоритмів.

Залежить від конфігурації. У цих алгоритмів є різні компоненти, що впливають на захищеність, і при правильній конфігурації вони здатні запобігти злому.

Висновок: з сильним паролем ви захищені від більшої кількості зломів, ніж зі слабким паролем. А оскільки ви не знаєте, наскільки захищене сховище паролів, ви не можете знати, наскільки буде «досить безпечно» для цього сервісу. Так що припускайте найгірше, коли ваш вибір пароля ще має значення.

Унікальності пароля недостатньо

Гаразд, але з якого дива вам думати про те, щоб використовувати менеджер паролів і генерувати унікальний пароль для кожного сайту? В цьому випадку ви невразливі длясredential stuffing- коли відома пара поштової скриньки і пароля перевіряється на різних сервісах в надії, що людина використовувала ці дані в різних місцях. Це серйозна загроза, тому що повторне використання паролів одна з головних проблем безпеки. Від цього вас захистить генерування унікального пароля для кожного сайту.

Credential stuffing.

А якщо базу вкрадуть і все її вміст стане відомо хакерам, то навіщо вам все ще захищати свій пароль?

Справа в тому, що ви не знаєте, зламана чи база, і продовжуєте користуватися сервісом. Тоді хакери отримають доступ до всієї вашої майбутньої активності на цьому сайті. Пізніше ви можете додати дані банківської картки, і вони про це дізнаються. А сильний пароль означає, що хакери не зможуть увійти під вашим аккаунтом і не зможуть скомпрометувати ваші майбутні дії.

Використання різноманітних служб злому.

Як оцінити силу пароля за допомогою ентропії

Сила пароля характеризується ентропією - числовим представленням кількості випадковості, яка міститься в паролі. Оскільки мова йде про великі числа, то замість1 099 511 627 776 (2 ^ 40)нам простіше сказати «40 біт ентропії». І оскільки злом пароля - це перебір варіантів, то чим їх більше, тим більше часу потрібно затратити на злом.

Для випадкових символів, згенерованих менеджером паролів, ентропія обчислюється за формулою: log2 (<кількість різних символів> ^ <довжина>).

З довжиною зрозуміло, а що таке кількість різних символів? Воно залежить від класів символів, що входять в пароль.

Наприклад, пароль з 10 випадкових малих і великих літер маєlog2 (52 ^ 10) = 57біт ентропії.

Щоб обчислити питому ентропію (її кількість в одному символі заданого класу), можна використовувати рівняннялog2 (n ^ m) = m * log2 (n). отримуємо:<Довжина> * log2 (<кількість різних символів>), Де друга частина є питомою ентропією. Перерахуємо по цій формулі попередню таблицю:

Для обчислення сили пароля потрібно взяти класи символів, які входять в пароль, взяти значення ентропії для цих класів і помножити на довжину. Для наведеного вище прикладу пароля з 10 малих і великих літер ми отримали5.7 * 10 = 57 біт. Але якщо збільшити довжину до 14, то ентропія скакнёт до 79,8 біт. А якщо залишити 10 символів, але додати клас спеціальних символів, то загальна ентропія буде дорівнює 64 біт.

Наведене рівняння дозволяє швидко обчислити ентропію пароля, але тут є підступ. Формула вірна лише в тому випадку, якщо символи не залежать одне від одного. А це відноситься тільки до згенерованих паролів. поєднанняH8QavhV2gвідповідає цьому критерію і має 57 біт ентропії.

Але якщо використовувати більш легкі для запам'ятовування паролі на кшталтPa $$ word11, То ентропія у них буде набагато нижче при тій же кількості символів. Взломщику не доведеться перебирати всі можливі комбінації, досить лише перебрати слова зі словника з деякими змінами.

Таким чином, всі обчислення з множенням довжини на питому ентропію вірні тільки для згенерованих паролів.

Керівництво по ентропії

Чим більше в паролі ентропії, тим складніше його зламати. Але скільки ентропії буде досить? В цілому, близько 16 символів буде за очі, у такого пароля 95-102 біта ентропії, в залежності від класів символів. А який мінімальний поріг? 80 біт? 60? Або навіть 102 біта занадто мало?

Є алгоритм, який за швидкістю змагається з поганим алгоритмом хешування, але зате вивчений набагато краще: це AES.

Він використовується в усіх урядових і військових організаціях, а значить його стійкості цілком достатньо. І працює швидко. Так що якщо AES-ключ з певною кількістю ентропії не можна зламати, то це піде на користь паролю з поганим (але не зламаним) хешем.

Національний інститут стандартів і технологій визначив розміри ключів, які будуть достатні в доступному для огляду майбутньому.

тамрекомендуютьвикористовувати AES-128 в період «2019-2030 і пізніше». Як зрозуміло з назви, мова йде про 128 бітах ентропії.

В іншийрекомендаціїрадять робити ключі розміром не менше 112 біт:

Для забезпечення криптографічного стійкості для потреб Федерального уряду сьогодні потрібно не менше 112 біт (наприклад, для шифрування або підпису даних).

Щоб отримати 128 біт ентропії з використанням великих і малих літер, а також чисел, потрібен пароль довжиною22 символу ((5.95 * 22 = 131 біт).

інші міркування

Чому б не використати спеціальні символи?Я намагаюся їх не застосовувати, тому що вони ламають кордону слова. Тобто для вибору спецсимволи потрібно три кліка замість двох, і це може привести до помилки, якщо я, бува, не вставлю в поле решту пароля.

А якщо використовувати тільки букви і цифри, то при подвійному натисканні виділиться весь пароль.

Що робити, якщо є обмеження по довжині?На деяких сайтах довжина пароля не може досягати 22 символів. Іноді паролі можуть бути тільки дуже короткими, наприклад, не більше 5 цифр. Тоді залишається тільки використовувати пароль максимально можливої довжини.

Також є рекомендації для сайтів по роботі з паролями, і обмеження довжини явно суперечить цим рекомендаціям. Ось щоговоритьНаціональний інститут стандартів і технологій:

Потрібно підтримувати паролі довжиною хоча б до 64 символів. Заохочуйте користувачів робити зручні для запам'ятовування секрети будь-якої довжини, з використанням будь-яких символів (в тому числі пробілів), що сприятиме запам'ятовуванню.

І пам'ятайте, що ступінь захисту паролів на сайтах варіюється від жахливої до чудовою, і вони не скажуть вам, яке реальне положення речей. Якщо максимально допустима довжина пароля невелика, тостворюється враження, Що такий сайт знаходиться на поганий частини шкали.

висновок

Паролі повинні бути сильними, навіть якщо ви не використовуєте одні і ті ж поєднання в різних місцях. Сила пароля вимірюється ентропією, і потрібно прагнути до значення в 128 біт. Для цього достатньо паролів довжиною 22 символи, що складаються з великих і малих літер, а також цифр.

Дослідження на тему паролів

- Інформаційна безпека

Тут я постараюся зібрати воєдино і проаналізувати всю інформацію про користувача паролі на різних ресурсах.

Пароль [parole] - це секретне слово або набір символів, призначений для підтвердження особи або повноважень. Паролі часто використовуються для захисту інформації від несанкціонованого доступу. У більшості обчислювальних систем комбінація «ім'я користувача - пароль» використовується для посвідчення

користувача.

2.6. Аналіз програмного забезпечення для зберігання та перевірки стійкості паролів

Щоб створювати і запам’ятовувати паролі для сервісів, ігрових акаунтів і соціальних мереж треба мати суперпамять або просто скористатися програмою для зберігання паролів. Такі менеджери спочатку запропонують створити складний пароль для ваших сайтів, а потім збережуть їх в локальному або хмарному сховищі. Ми зібрали 10 таких додатків, які допоможуть захистити акаунти від злому і зробити життя комп’ютерного користувача трохи краще.

Перед початком варто звернути увагу на способи зберігання даних в менеджерах паролів. Як писалося вище, інформація може зберігатися або віддалено в хмарі сервісу, або локально, на комп’ютері користувача. У кожного із способів є плюси і мінуси. Тут треба вирішити що важливіше: зручність чи максимальний захист даних.

Хмарне зберігання і синхронізація зручні тим, що сервіси паролів доступні на всіх пристроях, де вони потрібні. Але, якщо хмару зламають, паролі можливо потраплять до зловмисників.

Локальне місце зберігання більш надійне (хіба що у вас не вкрадуть комп’ютер або ноутбук), але менш зручний. Припустимо, ви створили пароль від Фейсбуку за допомогою менеджера паролів і зберегли інформацію на ПК. Але якщо зайти в *Facebook* зі смартфона, то новий пароль автоматично не підтягнеться і його доведеться вводити вручну. Якщо мова про один сервіс. А якщо говоримо про десятки акаунтів на різних сайтах, то подібна схема стає занадто незручною:

– Додаток *KeePass* має застарілий зовнішній вигляд, відкритий вихідний код і портативну версію, яка легко завантажується з флешки на будь-який комп’ютер. Оскільки у *KeePass* немає вбудованої синхронізації, там же, на

особистії хмарі або іншому носії зберігається база даних паролів. Для максимального захисту, рекомендується зберігати на особистії хмарі портативну версію *KeePass* і його базу даних. В цьому випадку скористатися можливостями програми вийде без проблем на будь-якому комп'ютері. *KeePass* доступна безкоштовно на *Android, iOS, mac, Windows* і *Linux*.

– додаток *Sticky Password* створено розробниками антивірусу *AVG*. Крім стандартного набору функцій, сервіс вміє працювати з базами даних інших менеджерів паролів і проводить *Wi-Fi*-синхронізацію. Програма доступна безкоштовно на *Android, iOS, mac* і *Windows*.

– програма *OneSafe* отримала розширений функціонал. Крім умінь класичного менеджера паролів, додаток блокує доступ зловмисникам до файлів або папок на ПК, а також робить резервні копії бази даних паролів на флешки, диски, переносні *HDD* та інші носії. Крім цього, *OneSafe* оснащений функцією *Decoy Safe*. Завдяки їй сервіс створює хибні акаунти користувача, щоб хакери зламували їх, а не оригінальні. Програма доступна безкоштовно на *Android, mac* і *Windows*. Додаток для *iOS* платне.

– 1*Password* – популярний менеджер паролів. Сервіс працює без Інтернету, а також вміє синхронізувати дані через *iCloud* і *Dropbox, Wi-Fi* або мережеві папки на комп'ютері. 1*Password* дозволяє ділитися доступом або паролями з друзями або близькими через додавання останніх в довірені контакти. В наявності розширення для браузерів *Safari, Firefox, Opera* і *Google Chrome*. Програма доступна на *Android, iOS, mac* і *Windows*. За мобільні додатки доведеться заплатити, але є тріал на 30 днів.

– При першому запуску *Dashlane* програма перевіряє вже створені паролі і якщо знаходить слабкі комбінації пропонує змінити, згенерувати і зберегти новий і надійний пароль. *Dashlane* оповіщає власника, якщо зловмисники зламають один з акаунтів користувача і відразу ж запропонують заблокувати його і змінити дані. Крім шифрування, *Dashlane* здатна зберігати чеки і платіжки з покупок в інтернеті, банківські дані, номери карт, рахунки та іншу особисту інформацію. Програма доступна безкоштовно на *Android, iOS, mac* і *Windows*.

– додаток *LastPass* має розширення для зберігання паролів в браузері.

Користувачам *Android* і *iOS* доступні рідні додатки. У *LastPass* є можливість поставити один СуперПароль, а вся інформація зберігається локально. Програма автоматично заповнює логін і пароль на будь-якому доданому сайті, дозволяє ділиться паролями з членами сім'ї або друзями. При цьому користувач сам вирішує дати їм сам пароль, або просто відкрити тимчасовий доступ до потрібного сервісу на короткий час. Програма працює безкоштовно на *Android, iOS, mac* і *Windows*.

– сервіс *Splikity* схожий на інші менеджери паролів і не виділяється особливими функціями. Тут є автоматична синхронізація даних, розширення для популярних браузерів і додатки на мобільники. Але *Splikity* знаходиться в списку 10 кращих програм для зберігання паролів через стильного і максимально зручного інтерфейсу. Цей сервіс — відмінний вибір для тих, хто тільки починає знайомство з менеджерами паролів. Програма доступна безкоштовно на *Android, iOS* і *Web*.

– менеджер паролів *Enpass* отримав настроюваний генератор паролів з вибором форми кодування і довжиною символів. В наявності вікно додавання паролів для браузера *Google Chrome*. Це зручно, коли треба створити багато складних паролів для десятка-другого сайтів, соціальних мереж або сервісів. База даних *Enpass* зберігається відразу в двох місцях: в особистій призначенії для користувача хмарі в зашифрованому вигляді (*Dropbox, Google Drive, OneDrive* і так далі) і локально на комп'ютері або смартфоні користувача. Програма доступна безкоштовно на *Android, iOS, mac* і *Windows*.

– програма *RoboForm* виступає менеджером паролів і захисником системи від фішингових атак. Для цього сервіс запам'ятовує або розпізнає правильні або мічені рекламою посилання, а в потрібний момент попереджає користувача про загрозу. Програма доступна на *Android, iOS, mac, Windows* і *Linux*. Є підписка, яка поширюється на синхронізацію з комп'ютером. Робота з іншими платформами безкоштовна.

– *SafeInCloud* обладнана генератором паролів і функцією автозаповнення полів при вході на будь-який сайт. Програма працює з браузерами *Safari, Firefox, Opera* і *Google Chrome,* синхронізує дані між своїми ж додатками *Windows, Mac,*

Android і *iOS*. Є можливість використовувати особисті хмари в особі *Dropbox, Google Drive, Яндекс*. Диск і *OneDrive*. Програма доступна безкоштовно на *Android, iOS, mac* і *Windows*.

2.7. Висновки до розділу

Кодування даних хеш-функцією *MD4* виконується також в п’ять етапів. Для виконання обчислення хеш-алоритму отримується деяка стрічка. Цю стрічку потрібно перетворити на послідовність нулів та одиниць. Про обрахунок даного хеш-алгоритму детальніше йтиме мова в описі алгоритму даної хеш-функції.

Шифрування інформації хеш-алгоритмом *MD5* виконується в п’ять етапів. Але про обчислення цієї хеш-функції більш детальніше йтиметься в описі її алгоритму.

Шифрування даних хеш-функцією *SHA-1* виконується за два етапи. З ними більш детально можна познайомитися в описі алгоритму даного хешу.

В програмі передбачається використання різних алгоритмів. Деякі алгоритми більш популярні і використовуються для кількох різних криптовалюта (блокчейнів). Найбільш популярні криптографічні алгоритми шифрування на сьогоднішній день це: *DaggerHashimoto, Scrypt, SHA256, ScryptNf, X11, X13, Keccak, X15, Nist5, NeoScrypt, Lyra2RE, WhirlpoolX, Qubit, Quark, Axiom, Lyra2REv2, ScryptJaneNf16, Blake256r8, Blake256r14, Blake256r8vnl, Hodl, Decred, CryptoNight, Skunk, Lbry, Equihash, Pascal, X11Gost, Sia, Blake2s*.

Список алгоритмів досить великий, проведемо аналіз найбільш затребуваних і надійних з них:

1) *SHA256* – саме на цьому алгоритмі побудований класичний біткоіни, для роботи з ним відеокарт вже давно мало, так як для нього наші брати-китайці винайшли спеціальне обладнання – Асіка (від англійської абревіатури *ASIC* (*application specific integrated circuit* – інтегральна схема спеціального

призначення). На цьому ж алгоритмі засноване і безліч копій біткоін криптовалют, наприклад, недавно з'явився *BitcoinCash*.

2) *Scrypt* – на цьому алгоритмі працює «цифрове срібло» – *Litecoin*. Свого часу цей алгоритм був чудовою альтернативою, коли Асіка були розроблені тільки для *SHA256*. Але часи йдуть, з'явилися *Scrypt-ASIC* і цей алгоритм більшості Майнер теж недоступний.

3) *Ethash* (*DaggerHashimoto*) – алгоритм шифрування, що знайшов застосування в криптовалюта *Ethereum*. Для ефективної роботи потрібно мати відеокарти з великим об'ємом ОЗУ і бажано на основі мікропроцесорів *AMD*, хоча і *Nvidia* 10-й серії непогано справляється з *DaggerHashimoto*.

4) *X11* – застосовується в монеті *Dash*, доступний для сучасних відеокарт. Однак в кінці 2017 – початку 2018, з'явилися перші партії *ASIC*-Майнер для *Dash*. *Decred* – модифікація алгоритму *Blake256*, застосовується в кріптомонете *Decred*. Можна Майні на відкритих. Перевірка стійкості до зламу *Decred* найчастіше запускається паралельно з *DaggerHashimoto* на програмі *Claymore's Dual Miner*.

5) *CryptoNight* – на основі даного алгоритму працює *Monero*. Алгоритм примітний тим, що відносно непогано обчислюється на процесорах.

6) *Equihash* – лежить в основі валюти *Zcash*, досить популярної в наш час серед Майнер на відкритих, в силу того що складність мережі *Ethereum* (лідера у *GPU* Майнер) сильно зросла.

7) *X11Gost* – алгоритм лежить в основі кріптомонети *Sibcoin*, яка ще називається «Сибірський Червонец». По суті це форк, російський аналог валюти *Dash*. В основі алгоритму лежить хеш-функція відповідно до ГОСТ Р 34.11-2012.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ ДЛЯ ПЕРЕВІРКИ СТІЙКОСТІ ПАРОЛІВ ДО ЗЛАМУ

3.1. Реалізація криптографічних алгоритмів в популярних системах

Sony Pictures database

Дослідження Троя Ханта, який взяв за предмет своїх досліджень, базу користувачів Sony Pictures, варто відзначити, що всі паролі зберігалася у відкритому вигляді. А далі він проаналізував паролі. Ось такі результати у нього вийшли.

довжина пароля

Як ми бачимо, основна кількість паролів з довжиною від 6 до 10 символів. При цьому у половини він менше 8 символів.

використовувані символи

1% - тільки літери верхнього регістру

4% - тільки цифри

45% - тільки букви нижнього регістра

50% - інші варіанти

Криптографічний стійкість пароля визначається варіацією букв різних регістрів + цифри + спец. символи ^ довжина пароля. На даному прикладі ми можемо спостерігати, що використовуються паси одного типу.

словникові паролі

36% - словниковий пароль

64% - пароль не з словника

При даній перевірці використовувався словник на 1.7 млн слів. Взяти можна тутdazzlepod.com/site_media/txt/passwords.txtЯк бачимо результат невтішний, більше третини пасів, словникові.

Тест на унікальність

8% - унікальний пароль

92% - повторно використовуваний пароль

У базу Sony Pictures так само включені паролі для інших сервісів. Власне на таблиці відображено, скільки користувачів всюди використовують один і той же пас.

брутфорс хешу

18% - складно зламувати

82% - легко зламував через райдужні таблиці

Оскільки всі паролі зберігалися у відкритому вигляді, але навіть в разі якщо це були-б хеші, нам вдалося б дешифрувати приблизно 82% від загального числа, із застосуванням райдужних таблиць.

E-mail password's

Незалежного досліднику в області ІБ, потрапив в руки список з + 20K акаунтів від різних поштовиків, виду email; pass. Списки розділені на 2 частини, можна взяти тут:

users - box.net/shared/m9fv11hcrc

passwords - box.net/shared/jek7g37fjk

Паролі можете навіть не пробувати зіставляти, оскільки все це справа мало намір перемішано. Оскільки метою було не компрометація користувачів, а проведення аналізу використовуваних паролів. Спочатку список складався з 24,546 записи. Всі вони мали такий вигляд username @ domain / password. Після проведення невеликої очищення, залишилося 23,573 аккаунта. Потім були видалені дублікати і на виході вийшов список з 21,686 аккаунта.

За форматом паролів, можна витягти наступну статистику.

43.3% - букви, в нижньому регістрі. Приклад: monkey

2.1% - букви, верхній і нижній регістр. Приклад: Thomas

15.8% - тільки цифри. Приклад: 123456

35.1% - букви і цифри. Приклад: j0s3ph

3.6% - букви, цифри і спец. символи. Приклад: sandra19_1961

30% - закінчується цифрою. Приклад: hello1

Якщо ми подивимося на довжину пароля в наступній діаграмі, то ми побачимо, що більшість з них 6-символьні.

Rootkit.com

6 лютого 2011, як частина атаки на HBGary, угруповання Anonymous, із застосуванням методів соціальної інженерії, вдалося скомпрометувати Jussi Jaakonaho, одного їх технічних адмінів rootkit.com. В результаті був дістати повний дамп ресурсу з усією базою даної, в тому числі і користувачів.

Для дешифрування використовувався John the Ripper. Більшість паролів були підібрані із застосуванням словника на 17.5 MB, а решта домагалися за допомогою інших комбінованих атак. Нижче представлені 10 найбільш часто використовуваних паролів.

Як ми бачимо знову засвітився вже побитий 123456. Так само варто відзначити той факт, 3 місце за популярністю, зайняв пароль аналогічний назвою ресурсу, аж 341 результат. Я так само хочу додати грунтуючись на своєму численному досвіді, коли працюєш з базами, з дешифрування пасів, досить часто трапляються ресурси, які в якості пароля використовують адресу цього самого ресурсу. При цьому дане спостереження не є правило. Але на моїй практиці вже були портали, де% таких пасів був досить великий, а були де взагалі не використовувався ні разу. Я поки не знайшов залежності даного спостереження.

А за наступним посиланням вам буде доступний ТОП-500 використовуваних паролів на ресурсі rootkit.com:

dazzlepod.com/rootkit/password

Для перевірки надійності пароля

Швидкість підбору за допомогою пароля можна описати нехитрої математичною формулою: кількість можливих символів, зведена в ступінь довжини пароля, поділене на кількість які перебираються паролів в секунду. В результаті виходить приблизний час у секундах. Втім, щоб довести людині, що на ділі означає простий пароль, не потрібно вантажити його математикою.

Досить зайти на сайт How Secure Is My Password ?, ввести <> (слово << пароль >> в латинській розкладці) і показати, що такий пас сбрутіть на звичайному PC за 30 секунд. Вобщем потестить паролі досить цікаво: відразу стає видно, що довжина пароля набагато важливіше його складності. Для злому "# R00t $ H3ll" піде 195 років, а на, здавалося б, простий «abcdefg1234567» - 5722 року.

І як колись написав адмін insidepro:

Цікаве спостереження - останні пару тижнів наша база hash.insidepro.com по параметру «Webcrack today» щодня лідирує, що не може не радувати. Причому наступаючий «на п'яти» сервісwww.md5decrypter.co.uk(Дуже гідний сервіс, имхо) має базу в 5 млрд. Хешів, але виходить, що наша 33-мільйонна база ефективніше, незважаючи на те, що вона менше більш ніж в 150 разів. Це ще раз підтверджує тезу - «справа не в кількості, а в якості».

Тому щоб вдало розкривати і дешіфровивать паролі, не треба відразу лізти качати rainbow table які важать сотні гігабайт. А досить розібратися в питанні, тоді відносно не велика база, яку ви будете час від часу поповнювати, буде показувати дуже хороший результати по пробиваемости.

Перечитавши масу супутньої літератури і переглянувши тонну хабратопіков (посилання на цікаві наведені в кінці статті), я вирішив узагальнити інформацію про основні методи генерації надійного і запам'ятовується пароля.

Почну з того, що для генерації і зберігання своїх паролів сам я користуюся чудовою програмоюKeePass. Її функціоналу цілком вистачає для всіх моїх скромних вебмастерскіх потреб. Основним її недоліком є той факт, що вона теж вимагає запам'ятовувати один головний пароль. Тому вся ця метушня навколо придумування пароля також стосується мене і всіх щасливих володарів програми KeePass або її аналогів, тому що один пароль придумати все-таки доведеться.

Поговоримо про методи злому

Щоб розуміти всю глибину проблеми, пару рядків присвячу методикою злому. Отже, як же зловмисник може дізнатися / вгадати / підібрати ваш пароль?

1. Метод логічного вгадування. Працює в системах з великою кількістю користувачів. Зловмисник намагається зрозуміти вашу логіку при складанні пароля (логін + 2 символи, логін навпаки, найпоширеніші паролі і т.п.) і застосовує цю логіку до всіх користувачів. Якщо користувачів багато, дуже скоро станеться колізія і пароль буде вгаданий;

2. Перебір по словнику. Цей вид атаки застосовується, коли база даних з хешировать паролями злита з сервера. Може поєднуватися з заміною букв (помилки) або з підстановкою чисел / слів в початок або кінець слова як приставки або суфікса. Також використовуються словники, набрані в неправильній розкладці клавіатури (російські слова в англійській розкладці);

3. Перебір по таблиці хешировать паролів. Передовий метод злому паролів, коли хеш-кодування вже згенеровані і залишається тільки знайти в базі відповідність хешу паролю. Працює дуже швидко навіть на слабких машинах і не залишає ніяких шансів власникам коротких паролів.

4. Інші методи: соціотехніка і соціальний інжиніринг, використання keylogger'ов, сніфферов, троянів і т.п.

надійність пароля

Узагальнюючи інформацію, отриману з різних достовірних джерел, я виділю основні ознаки стійкого до злому пароля (під зломом я маю на увазіперебір по базах хешів, Коли алгоритм хешування заздалегідь відомий):

1. Довжина пароля (чим більше, тим краще), для запущених випадків рекомендують використовувати 15-тісімвольний пароль;

2. Відсутність словникових слів і частин поширених паролів у складі пароля;

3. Відсутність шаблонів при складанні пароля (під шаблоном я розумію логічний алгоритм генерації пароля, наприклад: «Med777ведев», « 12 @ йцу @ 21» або навіть «q1w2e3r4t5»);

4. Стохастичні послідовності символів з різних груп (рядкові, прописні, цифри, розділові знаки і спец-символи);

Однак, всі ми люди з досить обмеженими здібностями до запам'ятовування незв'язної інформації, тому паролі, які підходять під вищеописані параметри, хоч і будуть досить стійкі до злому з одного боку, але, з іншого боку, їх дуже непросто запам'ятати. Тому розглянемо менш параноїдальні варіанти складання і запам'ятовування паролів.

Як народ запам'ятовує свої паролі?

Проаналізувавши способи генерації паролів хабралюдей, я прийшов до висновку, що основна методологія запам'ятовування пароля грунтується на складанні логічного або асоціативного ряду. Також використовуються всілякі спотворення слів. Це можуть бути:

1. Назви домену упереміж з логіном («gooUSERglcom», «UmailruSer»);

2. Певна стандартна фраза, яка прикріплюється до домену («passgoogleru», «passhabrahabrru»);

3. Поширена слово упереміж з значущими цифрами та іншими знаками («321DR67ag0On», Де 32167 - чит, який викликав 5 чорних драконів в Heroes of Might & Magic);

4. Російські слова в англійській розкладці («, k.lj [htyf» - «страва хрону»);

5. Перестановка букв («Мойна і Вир», «twirret»);

З особливо запам'ятовуються можна виділити техніку шокуючого абсурду: «молюски відгризли мої танцюючі геніталії»,записування паролів в мобілку в якості абонентаіробота з енциклопедією, Як зі словником можливих паролів.

Рекомендації по запам'ятовуванню паролів від професіоналів

Людський мозок не сильний в складанні абсолютно випадкових послідовностей. Тому ми можемо використовувати сильні його сторони, а саме -

складання послідовностей слів, які пов'язані між собою досить, щоб наш пароль було легко запам'ятати. Отже, ось деякі рекомендації з прикладами від Марка Бернета, автора книги Perfect Password. Selection, Protection, Authentication (прим .: все приклади мої):

1. Використовуйте в пароліантоніми,синонімиіомонімита ін. в різних комбінаціях з розділовими знаками і цифрами («молодойстарік18лет», «svetlo! темний», «собака = @ »);

2. Використовуйте формули і вирази («12! = 12.1», «@die ('hard')», «echo $ string»);

3. Використовуйте несправжні адреси електронної пошти (« Ya.Krevedko@ya.ya »);

4. Використовуйте рими в паролі («google'shmugl», «HABRa_kadabra»);

5. Повторення («http: // http: // double_pass», «zloe_zlo»);

6. Візуалізація («Зомбі виїли мені мо3г», «КуКла.Даша.плачет»);

7. Перебільшення («25-й годині ранку», «Путін'а в мери!», «Почухай МЕНІ шлунок»);

8. Використовуйте мати в паролі (тут вправляйтеся самі);

9. Один з найбільш надійних способів запам'ятати пароль - багаторазово набрати його на клавіатурі.

замість висновку

Багато користувачів в мережі надходять як я колись: у них є один простенький пароль для одноразового входу на якісь неважливі сайти і два-три довгих і складних СУПЕР-пароля для всього іншого. Звичайно, це краще, ніж один пароль для всього. Однак я рекомендую для кожного сайту мати свій пароль, дуже вже не хочеться спрощувати життя хакерам.

І, наостанок, факти для параноїків

1. Найпоширеніша цифра в паролі - 1, зустрічається в 21% паролів, в той час як інші цифри присутні в 7% -10% випадків;

2. 24% всіх паролів складаються з 6 символів;

3. Більше 60% всіх паролів містять тільки рядкові символи;

4. Найпоширеніший пароль в мережі: «123456»;

5. існуютьпрограми, Здатні перевіряти понад мільйон паролів в секунду на процесорі Pentium 4 з частотою 3ГГц;

6. Кількість можливих варіантів пароля довжиною 15 символів, в якому можуть бути використані всі типи символів стандартної клавіатури (в англійській розкладці), становить 463 291 230 159 753 000 000 000 000 000 варіантів.

Генератор / валідатор паролів за результатами злому LinkedIn

- Інформаційна безпека

Після аналізу підібранихпаролів до LinkedInз'явилася ідея створити генератор паролів, поєднаний з валідатором, що не допускає легко підбираються паролі. Найпростішого аналізу на довжину, наявність спеціальних символів тут мало - деякі паролі можна легко зібрати з максимально можливих «шматочків» і на їх перебір йде істотно менший час, ніж теоретично заявлене. І гарантій, що програма-генератор не видасть вам подібний пароль немає - випадковість, вона на те й випадковість. Моє творіння не претендує на повне вирішення питання, швидше за це привід для роздумів, але воно цілком працездатний (вихідні коди і невеликої розбір теж присутні).

Надійність vs довіру

Спочатку хотів оформити ідею як якийсь Веб-сервіс, проте одумався; грамотний користувач повинен спершу замислитися, куди може потрапити його можливий пароль, а для мережевих технологій - це питання без очевидної відповіді, і є всі приводи не довіряти подібним сервісів взагалі. Ви тільки закінчили введення пароля, а AJAX'ом він вже тягнути і збережений на сервері. Ні, вже, спасибі.

Невідоме десктопних програм в цьому плані нічим не краще, у нього є багато можливостей непомітно відправити запит в інтернет. Саме через це не користуюся програмами-генераторами сам. І незважаючи на величезні списки джерел ентропії, немає гарантій, що твій пароль не опиниться в чиїйсь базі.

Тому сам пропоную програму в початкових кодах (на C #, виходячи з того, що на розробку не хотілося витрачати більше пари годин) і настійно рекомендую збирати її з початкових кодів.

генерація паролів

Найпростіша і універсальна схема генерації - отримати деяку ентропію і перетворити її в рядок цільової довжини, із заданим набором символів.

Для перетворення є досить-такистандартне рішення, Яке зводиться до виклику хеш-функції, як ефективної згортки пулу ентропії (тобто деяких випадкових значень):

```
string generatePassword(string charset, int length)
{
 byte[] Result;
 SHA1 sha = new SHA1CryptoServiceProvider ();
 result = sha.ComputeHash (seed); // <--- поточний пул ентропії
 // create password
 StringBuilder output = new StringBuilder ();
 for (int i = 0; i <result.GetLength (0) && output.Length <length; i ++)
 output.Append (charset [result [i]% charset.Length]);
 // update seed
 for (int i = 0; i <result.GetLength (0); i ++)
 addToSeed (result [i]);
 return output.ToString ();
}
```

Самі випадкові значення можна набирати різними способами:

- апаратні рішення,

- оточення системи (значення в пам'яті, властивості процесів),
- отримання даних від інтернет-сервісів,
- взаємодія з користувачем.

Але перші три пункти я вважаю абсолютно надлишковими в ідеалі, а в реальності ще й погано прогнозованими. Немає ніякої впевненості, що апаратний датчик не зламається, і не видаватиме «одне і те ж», оточення системи може не бути досить динамічним, а інтернет-сервіс може спеціально повернути (і зберегти) згенеровані в корисливих цілях дані. Це випадкові числа, і навіть маючи можливість на них поглянути, ми нічого не зрозуміємо.

Так що вирішено було використовувати ті дані, процес отримання яких залежить від користувача, і одним з найпростіших і зручних варіантів - збереження переміщень миші. Кожне переміщення характеризується координатами і часом, що хоч і має передбачуваний в загальному вигляд, але в конкретних числах - завжди помітний і практично випадковий, особливо якщо заміряти час лічильником з високою роздільною здатністю (в тиках процесора).

```
void handleMouseData(int x, int y)
{
 / * Store information in seed array * /
 addToSeed (x);
 addToSeed (y);
 addToSeed (ticks ());
}
```

Та невелика частина коду, яка відноситься до формування ентропії з оточення, міститься в цій простій функції:

```
int ticks()
{
 long timer;
 if (QueryPerformanceCounter (out timer) == false)
 {
```

// high-performance counter not supported

throw new Win32Exception ();

}

/ * Ignore significant bits due its stability - we prefer using fast-changing low bits only * /

return (int) Timer;

}

Лічильник QueryPerformanceCounter має різний дозвіл в залежності від системи, а його значення змінюється в залежності від поточної завантаження процесора, кількості процесів, факту context switch. Одна величина, яка, хоч і має прогнозоване зростання, але в конкретних числах його висловити дуже складно.

Як наслідок, ми отримуємо хороші паролі, які не змушуючи користувача натискати якісь хитрі комбінації, годинами чекати закінчення генерації, мучитися зі складним інтерфейсом програми. Я не претендую на акуратність в дизайні інтерфейсів (навіть смішно стало, після усвідомлення скриншота нижче), але припускаю, що немає більш зручного рішення, ніж ткнути в ComboBox і вибрати з сгенеренной варіантів сподобався.

Оцінка часу на злам

Для прямого перебору підсумкове час можна обчислити, виходячи з таких параметрів:

- швидкість функції перевірки,
- потужність алфавіту,
- кількість символів.

Потужність алфавіту - це мінімальне (з розумним критерієм опису) безліч, що містить символи конкретного пароля. Як правило це набори виду:

- маленькі букви,
- заголовні букви,
- цифри,
- спецзнаки.

Тому для обчислення часу просто перевіряємо, символи з яких наборів містить наш пароль, і включаємо в потужність безлічі все символи набору:

double bruteforceCombinations = Math.Pow (26 * Upper + 26 * Lower + 10 * Numeric + specials.Length * special, pwd.Length);

Швидкість хешування розрізняється для різних алгоритмів, я надаю список з грубо оціненими значеннями для популярних методів (але в реальності ми не знаємо, який використовується на сервері, тому коректніше буде залишити тільки найшвидший з популярних алгоритм, зараз це MD5).

І не варто забувати прозакон Мура, Чому все обчислення стверджують про «мільйони років на злом» здаються дуже наївними. Хоча динаміку зростання обчислювальної потужності не можна передбачити напевне, але використовувати поточний темп зростання - краще ніж взагалі нічого.

І тепер, власне, головне - оцінка часуатаки декомпозицией.

Це не так складно, необхідно просто обчислити мінімальне розбиття нашого пароля на слова деякого словника релевантних комбінацій (досить часто зустрічаються підрядка) == довжина пароля. І обчислити «потужність» словника == кількість слів у ньому. Підсумкове кількість комбінацій - все так же «потужність» в ступеня довжини.

Якщо наш пароль виду «Good123Password» - то він розкладається на «Good», «123», «Password», які дуже поширені окремо і час, необхідний для перебору всього лише «зовсім небагато» в кубі.

Питання, звичайно, полягає в якості такого словника, але знову ж таки, будь-який варіант - краще ніж нічого. Я доклав до програми словник, отриманий після аудиту на LinkedIn, складений з найчастіших комбінацій, зустрінутих в 2.5 мільйони знайдених паролів. Сам словник - всього 40 кілобайт.

результати

Посилання на програму і вихідні:dl.dropbox.com/u/243445/md5h/pwdmaster.zip

Одним з приємних результатів є факт, що більша частина згенерованих псевдовипадково Паролі не розкладається на ймовірні комбінації. Якщо в інших програмах генераторах немає явних помилок, або «лазівок», і за якістю генерації вони не гірше моєї - то «можна спати спокійно». Частка паролів для 14-символьних алфавітно-цифрових комбінацій, що розкладаються по словнику (таким чином, що перебір по словнику швидше прямого перебору) - менше однієї тисячної відсотка.

3.2. Реалізація модуля розрахунку хеш-послідовностей

«Складний пароль» в розшифровці не потребує

- Інформаційна безпека

проскочив туттопік про «складні» паролі. На жаль, дивлюся, багато серйозно сприйняли цей «метод» ...

Використання карток шифрування не є надійним методом!Як правильно помітили в коментарях там - «це під час війни наші діди використовували» ... Але використовували більш досконалі методи.

Наведений в тій статті метод використовувати МОЖНА ні в якому разі (!), Паролі за цим методом повністю розшифровуються, Ви просто віддасте їх зловмисникам на блюдечку! І зараз я це доведу ...

Складність і параметри

Всього існувати за таким методом може 36 таблиць, що за складністю одно паролю з однієї букви або цифри. Відмінності таблиць тільки в одному параметрі: зсув вліво на [0-35] - число стовпців, рівне проміжку «a-z0-9».

Можете перевірити:таблиця один- цеz = (i + j-1)% 36,друга- цеz = (i + j+9)% 36, Де і і j приймають значення: «a» = 0, «b» = 1, «c» = 2, ... Перевірте для будь-яких інших значень - відмінність тільки в одній константі, навіть

незважаючи на те, що по рядку друга таблиця починається з «s» (що взагалі з т.з. математики не мало сенсу).

перша:

a + b + 1 = a (0 + 1 + 1 = 0)

a + c + 1 = b (0 + 2 + 1 = 1)

b + a + 1 = a (1 + 0 + 1 = 0)

друга:

a + j + 9 = s (0 + 9 + 9 = 18)

a + k + 9 = t (0 + 10 + 9 = 19)

b + i + 9 = s (1 + 8 + 9 = 18)

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

abcdefghijklmnopqr

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

stuvwxyz 0 1 2 3 4 5 6 7 8 9

Отже, знаючи зашифрований пароль і назву ресурсу - у нас залишається всього 36 вихідних паролів.

Фінальний приклад з оригінального тексту (про нього і нижче):

До речі, ось відразу і мінус видно - користуватися табличками людям складно, помилки - неминучі. Це зашифрований автором тієї статті пароль. У ньому помилка - «aa» на другій позиції - це «9» попершій табличці, А не «b».

Розшифровуємо (правильно) зашифрований пароль ... ось частина з 36 паролів (відповідні 36 різних зсувів таблиці і назвою - «habrahabr»): + 5 j4mmqil7j + 4 k5nnrjm8k + 3 l6ooskn9l + 2 m7pptloam + 1 n8qqumpbn + 0 o9rrvnqco

-1passwordp

-2 qbttxpseq-3 rcuuyqtfr-4 sdvvzrugs-5 teww0svht

-1- це і є шукана константа зміщення таблиці.

Так, це реальна «розшифровка», так, програму написав ... на кінці «p» тому що повтор пароля починається.

Зауважте, що нам потрібна тільки одна буква з пароля і та ж одна - з назви сайту для отримання однієї змінної (зміщення).

Маленькі літери в «зашифрованому» паролі здадуть які букви будуть голосними в назві сайту (зниження криптостійкості).

Так ось з вищесказаного, якщо ми не знаємо назви сайту, але знаємо де там голосна (а вона буде там, де в зашифрованому паролі маленькі букви), то у нас виходить (по одній будь букві) = 36 (табличок) * 6 (голосних = a, e, i, o, u, y) - 216 паролів (не знаючи назви сайту!). Зовсім небагато, але в реальності см. Нижче ("В реальності") - буде швидше за все тільки один.

Великі літери взагалі нічого не змінюють, крім того що їх назад маленькими треба зробити (не додають складності).

добивкидещо ускладнює завдання, але не в наведеному прикладі. Оскільки назва ресурсу довше, ніж пароль, то виходить розшифрувавши короткий пароль (знаючи зашифрований пароль і назву ресурсу), який піде далі по циклу - ми відновимо легко і добивки (ми знатимемо верхній рядок і нижню рядок - середню ми відновимо по тій же таблиці ... тут, крім пари «символів» на позиціях 4,8,16 (решітки, собачки), про неї - нижче).

У гіршому (але не реальному) випадку зловмисникові потрібно буде без поспіху перебрати 36 паролів на інший сайт, щоб увійти.

В реальності: З 36 паролів напевно тільки один буде виглядати як слово (!), Яке ви запам'ятовували, так що в реальності навіть перебору не буде потрібно. Так що пароль буде відомий точно.

Заміна на символи

Відступ про «символи». Автор запропонував замінювати літери отриманого пароля на позиціях 4,8,16 - на символи по табличці.

Знаки пунктуації додавали б складності, якби не ще одна уразливість методу - циклічність повторення пароля. Автор зрушив знаки пунктуації щодо латиниці. Ну що ж - це плюс, так, це ще один параметр складності, але з огляду на що ми точно знаємо що змінюються 4,8 і 16 символи - то ми можемо (по

довжині назви сайту «habrahabr») розшифрувати слово «pas * wor * p »в першій табличці (що під" * "ми не знаємо поки що).

Тепер хитрість в тому, що на (4 + 8) = 12 позиції та ж буква, що і на «4» (пароль-то по колу йде - це обидві літери «s» в слові «pass (4) wordpass (12) word »), але на 12 місці пунктуація не використовувати!

Отже ми можемо розшифрувати 12 букву і тут же дізнаємося 4ую букву.

Знаючи 4 зашифровану букву і соответвуют їй пунктуацію - ми лінійно отримали зміщення пунктуації. З цього зміщення тут же дізнаємося восьмий букву і як бонус - дванадцяту (яка сама). FAIL!

Правда якби автор зашифрував дванадцятий букву, а пароль був би 7-значним - ми б все одно її розшифрували (повторення літери було б на 4 + 7 = тринадцятий місці).

Якщо буде «добивки» - це буде псувати картину, але не сильно, знову ж таки - більша частина добивки буде розшифрована (ми вже будемо знати повністю або майже повністю верхній рядок і повністю - нижню - з неї відновиться середня, крім можливо (!) 1 -2 букв закритих пунктуацією, але тут вже перебір врятує - ще * 36 варіантів - трохи (і це рідкісний буде випадок, щоб вдало пунктуація закрила літери, що їх не відновити з повторюваного пароля).

Варіант ще гірше

Знаючи зашифровані паролі на двох сайтах (+ назви цих двох сайтів) - вихідний пароль повністю відновимо і табличка теж.

Грубо кажучи як система рівнянь

x + Y1 + t = Z1, x + Y2 + t = Z2,

де x = вихідний пароль, Y1, Y2 = назва ресурсів (відомо), t - зсув таблиці, Z1, Z2 = зашифрований пароль (відомо).

Два рівняння, дві невідомих - система лінійна і має одне рішення щодо x і t.

Насправді достатньо однієї літери з обох паролів і тієї ж букви назв сайтів. (Невідома-то величина тільки одна - зміщення таблиці вліво)

Тут навіть перебирати нічого не треба - пароль буде відомий точно.

висновок

Наведений в оригінальній статті алгоритм шифрування аналогічний буквено-цифрового паролю з 1 (одного!) Символу (log3636 = 1).

У разі якщо відомі шифровані паролі від двох сайтів - не є шифруванням взагалі.

Чи можна ускладнити

Навіть якщо Ви підсилите алгоритм тим, що будете використовувати російські букви і символи (які зміщуються незалежно), навіть якщо Ви додасте зрушення таблиці не тільки по горизонталі, але і по вертикалі, і навіть відображення по горизонталі і вертикалі, то отримаєте: 36 * 36 * 36 * 36 * 36 * 2 * 2 = 240 млн таблиць ... Виглядає страшно?

Насправді це аналогічно простому буквено-цифрового паролю приблизно з 5 букв. Що навряд чи можна вважати «надійним» або «складним» (27 біт шифрування).

сильне шифрування- це 128 біт, що приблизно в 140000000000000000000000000000000 раз повинно мати більше варіантів, ніж «ускладнені» 240 млн. (точно?) Це еквівалентно буквено-цифрового паролю з 25 букв. (Log36 (2128) = 24.75)

Єдиний спосіб, яким можна це дійсно «посилити» - це якщо:

1) Табличка буде по-перше «вивернута» (зараз потрібно по вертикалі, а потім по цьому рядку пальцем йти, а потрібно зробити щоб на перетині вертикальної і горизонтальної смуги був «зашифрований» символ) - це зменшить повторюваність і передбачуваність, яка зараз на максимумі, 2) Рядки і стовпці треба розтягнути, щоб не було відповідності латинського-російського-символу, все відокремити один від одного - це ще зменшить повторюваність і передбачуваність ,, 3) Табличка буде містити тільки абсолютно випадкові числа-

цифри-символи в тілі - і я маю на увазі абсолютно випадкові в кожному осередку - ніяких зрушень по рядах, 4) ніяких повторів (пароля, назви сайту), 5) ніяких «позиційних» замін.

Однак, навіть ця табличка не витримає серйозної атаки при тривалому використанні (як і всі «таблички»). Ще раз: використання карток шифрування не є надійним методом шифрування!

md5 ('site + key')

Багато порадили такий метод. Він непоганий, але його можна поліпшити.

Насправді, при наявності достатніх потужностей і обсягів пам'яті для RainbowTables є шанс, що знайдеться ключик. (Ма-а-а-а-а-а-а-аленький, але є - всього103 мільярди паролів в секундуможе один комп'ютер перебирати - так що, наприклад, «password» + «habr» за 10 днів розшифрувати можна і на домашньому комп'ютері)

Краще - робити раунди шифрування, простіше кажучи - візьміть site + key і зашифруйте його 6000 разів прямо ось md5 (md5 (md5 (... так 6000 разів ..... md5 ('site + key') ....). це не зупинить атакуючого, але серйозно уповільнить, настільки щоб атака стала непрактичною. Почитайте (англійською) що KeePass про раундипише - дуже пояснить чому я так говорю.

```
# Псевдокод:
k = key + site
for 0..6000 {
 k = md5 (k)
}
print k
```

А це вже розшифровувати 150 років, проти 10 днів для звичайного md5 (site + key). Але навіть один раз - це досить хороший метод.

завершення

Ось так навіть дуже вражаюче виглядають алгоритми виявляються зовсім нестійкими.

Будьте пильні, придумування і реалізація алгоритмів шифрування - одна з небагатьох областей, в яких можна довіряти тільки людям, що спеціалізуються саме на шифруванні (роками) і тільки алгоритмам, які були перевірені і схвалені іншими фахівцями в цій галузі.

3.2.1. Складення стійких комбінацій

Перелічені нижче способи допоможуть придумати дуже складний символьний ключ, який легко запам'ятати.

1. Створіть візуально контури геометричної фігури або будь-якого предмета на клавіатурі вашого комп'ютера. А потім наберіть символи, за якими проходять лінії.

Уникайте простих «конструкцій» - лінії, квадрати або діагоналі. Їх легко передбачити.

2. Складіть складне речення, що не піддається логіці. Іншими словами, будь-якої каламбур:

Наприклад: Кот Васька на Юпітері вловив щуку.

Потім візьміть перші 2-3 літери кожного слова з придуманого пропозиції:

Кот + Ва + На + Юп + вул + щук

Наберіть склади латинськими буквами:

Rjn + Df + Yf +\u003e g + ek + oer

Після транслітерації вставте між складами якісь добре знайомі вам числа: дату народження, зріст, вага, вік, останні або перші цифри телефонного номера.

*Rjn*066*Df* 45*Yf* 178*\u*003*e g* 115*ek*1202*oer*

Вийшла досить «міцна» комбінація. Щоб згадати її швидко, вам потрібен тільки ключ (пропозиція-каламбур) і використовувані цифри.

3. Візьміть за основу 2 пам'ятні дати. Наприклад, два дня народження (ваше і вашого улюбленого людини).

12.08.1983 05.01.1977

Розділіть число, місяць і рік якимись спецсимволами:

12|08/1983|05\01|1977

Тепер нулі в датах замініть маленькою літерою «*o*».

12 | *o*8 / 1983 | *o*5 \\ *o*1 | 1977

Виходить досить складний ключ.

4. Зробіть спеціальну таблицю: по вертикалі і горизонталі матриці розташуйте латинські букви і цифри, а в рядках і стовпцях - символи в хаотичному порядку.

Для генерації ключа візьміть кілька простих слів, записаних англійськими літерами, наприклад, *my password very strong*

Візьміть першу пару букв. У нашому випадку це «*my*». У вертикальному списку знайдіть «*m*», в горизонтальному «*y*». На перетині ліній ви отримаєте перший символ пароля.

Таким же чином, за допомогою наступних пар, знайдіть інші символи ключа.

Якщо ви забудете пароль, для його відновлення скористайтеся простим ключовим словом і таблицею.

3.2.2.

Кілька методів ускладнення пароля

Яким же повинен бути пароль? Цим питанням задаються сотні користувачів інтернету. Розрізняють такі види паролів:

буквений;

символьний;

цифровий;

комбінований (комбінація попередніх варіантів);

використання регістра.

Перші три види не вселяють довіри. Це занадто прості способи створення пароля. Через недосвідченість ми робимо помилки і ставимо їх. Гаразд, це буде «пасворд» для аккаунта на форумі або іншому подібному місці. А, якщо це вхід в кабінет банку, все пропадуть ваші грошики. Єдине, що рятує - служба безпеки подібних сайтів розробила систему відкидання легких паролів.

Букви, цифри і символи

Поєднання букв, символів і цифр - найбезпечніший вид пароля. Потрібно серйозно поламати голову, щоб його відгадати.

Досвідчені «юзери» радять новачкам використовувати саме цю комбінацію. Також не варто робити його занадто коротким. Довга комбінація дозволить зберегти свої дані і листування в безпеці від третіх осіб.

Головне не використовуйте банальні фрази нижче:

«123»;

«123456»;

«321»;

«*Qwerty*»;

«*Asdfg*».

Ці та інші подібні набори символів з клавіатури гарантують злом. Не тільки вам вони приходять першими в голову, а сотням людей. Їх визначить навіть не спеціальна програма, а зазвичай недоброзичливець.

Як же підібрати пароль для пошти або іншого виду авторизації? Цим питанням варто зайнятися самостійно. На допомогу прийдуть ще кілька варіантів ускладнення пароля.

регістр

Перед тим, як ввести логін і пароль, варто звернути увагу на чутливий регістр деяких форм. Комбінування великих і малих літер зроблять пароль надійнішим.

Складаючи секретне слово, подумайте над його різноманітністю. Чергуйте заголовні і маленькі букви по одній або декілька штук. Такий метод серйозно засмутить мережевих лиходіїв.

Найприкріше, якщо ви самі забудете порядок. За рекомендацією досвідчених користувачів варто перший символ зробити прописних, другий - рядковим, і далі чергувати по одному. Даний рада краще взяти на замітку, щоб потім не ламати собі голову.

Без впровадження особливостей регістра в «пассворд» можна обійтися, але це все-таки це ще один метод підвищити складність пароля.

перевертні

Дата народження, яку згадає будь-який користувач - це самий банальний і простий спосіб. Якщо її правильно обіграти, то може вийти непоганий варіант. Використовуючи «перевертень», багато хто зумів створити виграшний пароль, який навряд чи піддасться розгадки.

Спосіб заснований на написанні символів в зворотному порядку. Вибираєте будь-яку дату, наприклад, коли ви народилися і набираєте текст навпаки. Якщо у

вас задумана фраза «081978», то перевертаючи, отримуємо «879180». Досить просто запам'ятати, як пишеться такий пароль.

Розглянемо і інші більш складні задумки. Припустимо, основа пароля - ваші ім'я та прізвище. Набираємо, вже знаючи техніку із застосуванням регістра - «*PeTrPeTrOv*». Тепер застосуємо тактику «перевертнів». Застосовуємо дату, наприклад, коли народився користувач - 21 листопад 1982 року. Плюс до всього додамо символів. В кінці отримуємо наступний приклад пароля - «*PeTrPeTrOv*! 28912012». Підсумок вийшов приголомшливим, адже для «юзера» він простий і легкий, але ніяк не для зловмисників.

Перевірте надійність і безпеку пароля за допомогою онлайн сервісів:

https://password.kaspersky.com/ru/

https://howsecureismypassword.net/

шифрування

Який же все-таки повинен бути пароль? Дізнаємося ще один відмінний спосіб. Будемо розглядати принцип шифрування. По суті всі розглянуті раніше методи перегукуються з цим. Тут ми покажемо, які бувають паролі шляхом шифрування фраз.

Беремо найбезглуздішу і унікальну фразу, яка легко відкладеться в пам'яті. Нехай буде «космічні таргани». Ви можете використовувати будь-які рядки з пісень і віршів, бажано не сильно відомих.

Потім застосуємо до нашої фразі шифр. Розглянемо кілька вірних способів:

– переписування російськомовного слова на англійській розкладці;

– «Перевертень»;

– заміна букв на символи зовні схожі (наприклад, «о» - «()», «*і*» - «!», «а» - «@»);

– видалення парних або непарних символів;

– викидання приголосних або голосних букв;

– додаток спецсимволами і цифрами.

Отже, задумайтеся кілька слів зі змістом - «космічні таргани». Беремо по 4 букви від кожного, отримуємо «космтара». Перемикається на англійську мову і передруковуємо - «*rjcvnfhf*». Ускладнюємо, починаючи шифр з великої літери і додаючи символи.

Ось яким повинен бути пароль на прикладі спочатку задуманої фрази - « *Rjcvnfhf@*955».

Придумана надійна комбінація з великою кількістю символів. Перевіряється надійність пароля за допомогою спеціальних сервісів, наприклад, *passwodmetr.com*. Комбінацію, як у нас вийшла не просто вгадати шахраям, так особисті дані користувача не залучені. А ось для «юзера» такий «пассворд» знахідка, так як запам'ятати такий надійний пароль не складе труднощів.

Для тих, хто не бажає витрачати зайвий час на роздумування, розробники давно винайшли генератори складних паролів. Цей спосіб забезпечує певний рівень надійності. Кращими все одно вважаються «пассворд», придумані своїм розумом.

Що таке генератор і як ним користуватися? Це розумна програма, яка виводить рандом паролів - скоєно випадково випали комбінації. Він використовує багато розглянуті методи, але «перевертоші" не бере до уваги.

Генератор складних паролів скачується з мережі. Наприклад, візьмемо «*keepass*». Як і будь-який інший генератор працює не складно. Запускається додаток і сама генерація шляхом натискання спеціальної кнопки. Після виконаної операції ПК видає варіант пароля. Залишається справа за малим - вписати отриману комбінацію в незмінній формі або з доповненнями.

Важкі паролі, придумані залізним другом, дуже складно запам'ятати. Рідко хто зберігає їх в розумі, частіше доводиться записувати. Паролів зазвичай буває багато, ми ж не сидимо на одному сайті і постійно реєструємося знову-знову на інших ресурсах. Тому зберігати купу подібної інформації не всім зручно. Можна і зовсім втратити всі папери із записами.

Є один вихід зі зберіганням - надрукувати їх у файлі комп'ютера. Це один з надійніших випадків. Варто лише пам'ятати, що система ПК не вічна і теж приходить в непридатність.

Всі способи створення ускладнених паролів вже розглянуті вище і ви зможете створити пароль для електронної пошти, який надійно захистить ваші дані від третіх осіб.

Кілька порад по створенню паролів:

– не згадувати особисту інформацію про користувача (ПІБ родичів, клички домашніх тварин, номери телефонів, адреси, дати народження та інше);

– в паролі не можна використовувати кирилицю;

– не вживати фрази, які легко обчислюються за допомогою словника популярних паролів (ястерва, *love, alfa, samsung, cat, mercedes* та інші подібні, а також інші їх похідні і поєднання);

– враховувати довжину символів - бажано не менше 10;

– ускладнювати пароль комбінацією всіляких методів - великі та малі літери, цифри, символи;

– не вживати найчастіші паролі - шаблони, думати оригінально (робот, який обчислює ваш пароль, не зможе бути настільки розумним, як людина).

Незважаючи на стрімкий розвиток технологій і поява альтернативних способів розпізнавання власника, парольний захист не здає своїх позицій і залишається вельми популярною і до цього дня.

Пароль перетворився в буденність і використовується для доступу до пристроїв і інтернет сервісів. І з часом їх стає тільки більше. Ситуація, що

склалася, в кінцевому підсумку, призводить до того, що користувачі починають використовувати один і той же пароль на використовуваних пристроях і сервісах.

Даний підхід дуже небезпечний і загрожує серйозними наслідками. Безсумнівно, скомпрометований пароль від соціальної мережі не несе таких наслідків, як пароль від платіжної системи. Але якщо вони будуть ідентичні, то ймовірність того, що доступ буде отримано і до решти використовуваним сервісів дуже високий.

Щоб цього не сталося, паролі повинні бути складними (стійкими до перебору) і різними.

Принципи, що використовуються при створенні пароля

На більшості інтернет ресурсів існують мінімальні правила для встановлюваного пароля, яких, найчастіше, недостатньо для створення дійсно складного пароля. Необхідно ще пам'ятати про те, що:

Логін і пароль не повинен бути ідентичним

Пароль не повинен складатися з особистої інформації (дата народження, телефон і т.д.)

Пароль не повинен складатися виключно з слів

Наприклад, щоб підібрати пароль, що складається з 6 цифр - необхідно перебрати всього 1 мільйон комбінацій. Сучасний комп'ютер справиться з цим завданням за лічені хвилини. З цієї ж причини не варто покладатися на паролі, що складаються виключно з слів і їх поєднань. Такі паролі перебираються з використанням словників популярних слів.

Не варто покладатися і на паролі, які складаються зі слів з додаванням цифр. Вони настільки ж схильні до злому, хоч на це і потрібно набагато більше часу. Однак, в разі успішного злому і понесених втрат в зв'язку з цим, чи це буде мати якесь значення.

Для кращого розуміння, який пароль є надійним, а який схильний до злому, варто звернутися до прикладів. Дані цифри були отримані за допомогою сервісу перевірки стійкості пароля.

Дата народження 12071996 - 0,003 секунди

Ім'я з великої літери *Maksim* і малої *maksim* - максимум напівсекунди

Поєднання, що складається з букв і цифр 7*s*3*a*8*f*1*m*2*a* - близько доби

На перебір наступного поєднання *vSA-DFRLLz* - 1 рік

Поєднання *iu*2374*NDHSA*) *DD* - 204 млн років

Останні два пароля демонструють досить високу стійкість до злому. Робота зловмисника над зломом аналогічного за складністю пароля, швидше за все, закінчиться нічим.

З теоретичною частиною ми розібралися, тепер перейдемо безпосередньо до генерації стійкого і надійного пароля.

При створенні складного і стійкого пароля істотну роль відіграє людський фактор. Труднощі виникають на самому початковому етапі - вигадуванні складного пароля, а після - його запам'ятовування. Адже комбінація розрізнених символів чи привертає до швидкого запам'ятовування.

З проблемою генерації стійкого пароля нам допоможуть он-лайн сервіси. Їх досить багато, з популярних російськомовних сервісів можна відзначити:

Passwordist.com

Online-Generators.ru

PassGen.ru

Працюють представлені сервіси за одним принципом, від вас лише потрібно вказати які символи необхідно використовувати і вибрати довжину генерується пароля.

Окремою особливістю сервісу *Passwordist.com* можна відзначити можливість задати кількість створюваних паролів і генерувати варіанти з кращого читабельністю за рахунок виключення схожих символів, наприклад, *B* і 8.

Надійні паролі згенеровані, але це ще півсправи. Паролі необхідно правильно зберігати, щоб ніхто сторонній не отримав до них доступ.

У зв'язку з цим варіанти із записом в текстовий файл або на стікер з подальшим прикріпленням до монітора відразу відпадають.

Краще і правильніше довірити конфіденційну інформацію менеджеру паролів.

Серед популярних рішень можна відзначити програму *KeePass*. Дана програма є безкоштовною і в той же час дуже функціональна. Крім іншого в ній присутній генератор паролів, завдяки якому відпадає необхідність у використанні он-лайн генератора.

Для доступу до бази збережених паролів необхідно буде встановити майстер-пароль. Для його створення можна, наприклад, скористатися методикою набору слів в іншій розкладці, щоб створити складний пароль, але при цьому самому не забути його.

Локальна база з паролями на вашому комп'ютері апріорі буде менш схильна до злому, ніж загальнодоступні сервіси в інтернет, так що тут зі складністю переборщувати не варто.

Багато сайтів намагаються допомогти користувачам встановити більш складні паролі. Для цього встановлюють базові правила, які вимагають зазвичай вказати хоча б одну велику літеру, одну малу літеру, одну цифру і так далі.

3.3. Висновки до розділу

В третьому розділі розкрито етапи розробки системи первірки стійкості паролів до залму.

Складність ключа - міра стійкості до підбору на символьному рівні за допомогою ручних і автоматизованих методів (логічного обчислення, підбору за словником). Вона визначається кількістю спроб зломщика, тобто, скільки йому знадобиться часу на обчислення складеної користувачем комбінації.

На складність пароля впливають такі чинники:

– кількість символів в ключі. Чим більше знаків в послідовності, тим краще. У комбінації з 5 знаків є велика ймовірність швидкого злому. А ось на підбір послідовності з 20 знаків можуть піти роки, десятиліття і навіть століття.

– чергування великих і малих літер;

– символьні набори. Різноманітність типів символів підсилює стійкість. Якщо зробити ключ з маленьких і великих літер, цифр і спецсимволов довжиною в 15-20 знаків, шансів його підібрати практично немає.

ВИСНОВКИ

Сучасна криптографія включає в себе три напрямки: шифрування із закритим ключем, шифрування з відкритим ключем і хешування. Сьогодні ми поговоримо про те, що таке змішування і з чим його їдять. В цілому під хешированием розуміють перетворення вхідних даних довільної довжини в вихідну бітову рядок фіксованої довжини. Найчастіше хеш-функції застосовують в процесі аутентифікації користувача (в базі даних зазвичай зберігається хеш пароля замість самого пароля) і для обчислення контрольних сум файлів, пакетів даних, тощо. Одним з найбільш відомих і широко використовуваних алгоритмів хешування є *MD*5.

Як було доведено в роботі, безумовно можливо створити ймовірні паролі, особливо ті, які слідують шаблонами людського мислення і містять дні народження, родичів, домашніх тварин, інтереси, місця - в загальному, речі, які якось пов'язані з нами. Це просто, як зазвичай працює наш мозок, ми можемо легко запам'ятати їх, тому що вони важливі для нас.

Тема безпечних і надійних паролів завжди залишається актуальною. Особливо сьогодні, коли зловмисники зламують акаунти за допомогою багатьох автоматичних способів підбору паролів. Довжина і складність пароля повинні бути настільки неприступними, наскільки важлива інформація, яку ви зберігаєте під цими паролями.

Придумати пароль, а особливо максимально надійний, часом буває дуже складно.Особливо ця проблема актуальна, коли потрібно реєструватися на багатьох ресурсах, де система запитує придумати пароль не менше 8 символів. Короткий пароль без цифр і з буквами одного регістра вважається небезпечним. Надійний пароль повинен бути максимально довгим і складним для злому.

Як придумати надійний пароль? Вихід в цій ситуації дуже простий - нічого не потрібно придумувати, треба скористатися безкоштовним онлайн генератором паролів. Він допоможе згенерувати випадковий пароль практично для будь-яких цілей.

Часто придумати пароль потрібно в особистий кабінет, більшість користувачів вводять пароль, який вони вже використали раніше на інших ресурсах і той, який їм легко запам'ятати. Ця ситуація в корені неправильна, такий спосіб захисту акаунтів дуже легкий для злому. Як створити надійний пароль для облікового запису? Для цього скористайтеся спеціальними сервісами, які за запитом генерують максимально складну комбінацію. Її не обов'язково заучувати, для зберігання паролів використовуйте спеціальні Менеджери паролів. Наприклад, програми *LastPass, RoboForm, KeePass, Sticky Password, OneSafe* і багато інших, які допоможуть вам зберігати паролі в безпеці.

Безпечні генератори паролів (*secure password generator*) - спеціальні сервіси для швидкої генерації складних випадкових паролів. Рандомна генерація паролів завжди більш безпечна, ніж ввести пароль - дату свого народження :)

Сьогодні в мережі можна знайти багато сервісів, які створюють випадково згенеровані паролі, і більшість з них дійсно вирішать ваші завдання. Ці генератори паролів використовують алгоритми для створення випадкових наборів букв, цифр і спеціальних символів. Рандомні паролі зазвичай створюються для задоволення певних вимог безпеки таких, як використання комбінації букв, цифр і спеціальних символів. Ці паролі вважаються надійними, тому що вони випадкові. Тобто вони генеруються з використанням алгоритму, а не, наприклад, комбінації імені вашого вихованця і його дня народження.

На жаль, досить легко отримати їх за допомогою таких методів, як робот-павук, який спрямований на збір інформації про конкретній темі, спираючись на системи ШІ, які обробляють і аналізують зібрані дані

Незважаючи на те, що можна генерувати все більше і більше ймовірних паролів, нам зазвичай все ще потрібно кілька спроб знайти правильний. Кількість необхідних спроб зазвичай пропорційно здійсненності списку паролів.

Можна подумати, що кілька паролів можуть бути легко визначені/попереджені, і в цьому є рація, але тільки в деякій мірі. В даний час більшість систем авторизації обмежують кількість спроб введення пароля для запобігання атак методом підбору.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

Додаток А

Програмний код основної частини