Teoria

Kryptograficzne funkcje hashujące, zwane też funkcjami skrótu to algorytmy, które potrafią przekształcić dane dowolnej długości w krótki ciąg bajtów (zwykle, dla czytelności zapisywany w formacie szesnastkowym).

OpenSSL to wieloplatformowa, otwarta implementacja protokołów SSL (wersji 2 i 3) i TLS (wersji 1) oraz algorytmów kryptograficznych ogólnego przeznaczenia. Dostępna jest dla systemów uniksopodobnych (m.in. Linux, BSD, Solaris), OpenVMS i Microsoft Windows. OpenSSL zawiera biblioteki implementujące wspomniane standardy oraz mechanizmy kryptograficzne, a także zestaw narzędzi konsolowych (przede wszystkim do tworzenia kluczy oraz certyfikatów, zarządzania urzędem certyfikacji, szyfrowania, dekryptażu i obliczania podpisów cyfrowych). OpenSSL pozwala na używanie wszystkich zastosowań kryptografii.

Crunch jest jednym z najwygodniejszych narzędzi do tworzenia słowników z listami haseł. Pozwala on wygenerować wszystkie kombinacje, które możemy wykorzystać do testowania. Trzeba przygotować się na to, że wygenerowany przez crunch słownik będzie sporych rozmiarów.

John the Ripper został stworzony przez Solar Designer. Program na początku był opracowany dla systemu operacyjnego UNIX, aktualnie uruchamia się na piętnastu różnych platformach i obsługuję różne architektury sprzętowe. Jest to jeden z najpopularniejszych programów do łamania oraz testowania haseł. Jego fundamentalnym celem jest wykrycie słabych haseł będących najsłabszym ogniwem większości dzisiejszych serwerów. Dzięki temu narzędziu możemy zapobiec złamaniu słabego hasła przez intruza, nakazując danemu użytkownikowi, który posługuje się złamanym hasłem, jego zmianę z uwzględnieniem cech charakterystycznych mocnego hasła.

Base64 służy do kodowania ciągu bajtów za pomocą ciągu znaków. Kodowanie to przypisuje 64 wybranym znakom wartości od 0 do 63. Ciąg bajtów poddawany kodowaniu dzielony jest na grupy po 3 bajty. Ponieważ bajt ma 8 bitów, grupa 3 bajtów składa się z 24 bitów. Każdą taką grupę dzieli się następnie na 4 jednostki 6-bitowe, więc istnieją dokładnie 64 możliwe wartości każdej z tych jednostek. Jednostkom przypisywane są odpowiednie znaki na podstawie arbitralnie ustalonego kodowania (patrz tabela poniżej). Jeśli rozmiar wejściowego ciągu bajtów nie jest wielokrotnością liczby 3, to stosowane jest dopełnianie – na końcu wynikowego ciągu dodawana jest taka liczba symboli dopełnienia (ang. pad), aby ten miał długość podzielną przez 4.

Istnieje kilka różnych schematów uwierzytelniania, których można używać w systemach Linux. Najczęściej stosowanym i standardowym schematem jest przeprowadzanie uwierzytelnienia na ***/etc/passwd*** i ***/etc/shadow***. Ponieważ hasła użytkowników umieszczane są w pliku ***/etc/shadow***, konieczne jest połączenie obu plików w jeden. Użytecznym narzędziem łączącym oba te pliki jest narzędzie ***unshadow***.

/etc/passwd to plik, w którym informacje o użytkowniku (takie jak nazwa użytkownika, identyfikator użytkownika, identyfikator grupy, lokalizacja katalogu domowego, powłoka logowania, ...) są przechowywane podczas tworzenia nowego użytkownika.

/etc/shadow to plik, w którym ważne informacje (takie jak zaszyfrowana forma hasła użytkownika, dzień wygaśnięcia

hasła, niezależnie od tego, czy hasło musi zostać zmienione, minimalny i maksymalny czas między zmianami hasła, ...) są przechowywane, gdy zostaje utworzony nowy użytkownik.

Zadania

1.1 Za pomocą narzędzia `rand` dostarczonego wraz z pakietem `OpenSSL`, wygeneruj 4 bitowe hasło i zakoduj go za pomocą Base64. Następnie, za pomocą narzędzia `OpenSSL`, wygeneruj hash MD5 tego hasła. Nie korzystaj z pomocniczych plików.

1.2 Korzystając z poleceń systemowych (np. `tr`, `head`, `cut`, `base64`) oraz pliku `/dev/urandom` wygeneruj bezpieczne hasło dla użytkownika (hasło ma mieć długość 16 znaków, nie może posiadać nie-alfanumerycznych znaków) a następnie wygeneruj, za pomocą narzędzia `OpenSSL`, funkcję skrótu MD5 dla wygenerowanego wcześniej hasła. Nie korzystaj z pomocniczych plików.

1.3 Korzystając z narzędzia `crunch`, wygeneruj do pliku listę haseł składających się z samych cyfr o długości 3 znaków. Zapisz je do pliku. Za pomocą narzędzia `OpenSSL` wygeneruj funkcje skrótu SHA-1 dla wszystkich wygenerowanych haseł.

1.4 Korzystając z narzędzia `crunch`, wygeneruj do pliku listę haseł o długości 5 znaków według wzoru, gdzie:

- pierwszy znak hasła to cyfra
- drugi znak hasła to mała litera *a*
- trzeci znak hasła to znak specjalny
- czwarty znak hasła to mała litera *b*
- piąty znak hasła to cyfra

Zapisz je do pliku. Za pomocą narzędzia `OpenSSL` wygeneruj funkcje skrótu SHA-3 dla wszystkich wygenerowanych haseł.

1.5 Korzystając z narzędzia `crunch`, wygeneruj do pliku listę haseł o długości 3 znaków według wzoru:

- pierwszy znak hasła to litera ze zbioru {`a, b, c`}
- drugi znak hasła to cyfra za zbioru {`4, 6, 8`}
- trzeci znak hasła to znak specjalny ze zbioru: {`?, %, :`}

Przykładowe hasła: `a4?, b6%, c8:, ...`. Zapisz je do pliku. Za pomocą narzędzia `OpenSSL` wygeneruj funkcje skrótu SHA-3 dla wszystkich wygenerowanych haseł.

1.6 Za pomocą słownika `rockyou.txt` oraz programu JohnTheRipper, spróbuj złamać hash MD5: `8afa847f50a716e64932d995c8e7435a`.

1.7 Za pomocą słownika `rockyou.txt` oraz programu JohnTheRipper, spróbuj złamać hash SHA-256: `437d9b521abe3c4102db90f7873cb4699cf9e38476c32b586cb786eb39eb6992`.

1.8 Ze strony `https://gparted.org/download.php` pobierz plik `gparted-live-1.3.1-1-amd64.iso`. Za pomocą `OpenSSL` sprawdź integralność pobranego pliku. (*Integralność polega na zapewnieniu, że przetwarzana informacja nie została w żaden sposób zmieniona. Zmiana taka może być przypadkowa (błąd podczas transmisji) jak i celowa (zmiana przez atakującego)*).

1.9 Napisz skrypt w języku Python, w którym wygenerujesz hash MD5 dowolnego ciągu znaków podawanego jako argument wywołania skryptu. Sprawdź poprawność wygenerowanego hasha porównując go z wynikiem otrzymanym przy pomocy `md5sum` lub `openssl`.

1.10 Napisz skrypt w języku Python, w którym wygenerujesz hash SHA-1 dowolnego pliku podawanego jako argument wywołania skryptu. Sprawdź poprawność wygenerowanego hasha porównując go z wynikiem otrzymanym przy pomocy `sha1sum` lub `openssl`.

1.11 Mając dany początkowy ciąg znaków `R3iSrSNmgU9SFHxVekUD`, który następnie został zahashowany, określ, jaka funkcja skrótu została wykorzystana do utworzenia hasha: `48cab4b54bef42fddaa6353c68a20b369f40026e`.

1.12 Sprawdź, czy pliki `a.txt` oraz `b.txt` mają taką samą zawartość.

1.13 Wykonaj zadanie **1.6** za pomocą narzędzia `hashcat`.

1.14 Za pomocą słownika `rockyou.txt` oraz oprogramowania `JohnTheRipper` sprawdź jakie hasła mają użytkownicy `u1` i `u2` (plik `z2.shadow`). Hasła te mają charakter słów słownikowych.

1.15 Użytkownik ma hasło (plik `z5.shadow`) składające się z 3 dowolnych znaków. Jakie to hasło? Do rozwiązania zadania użyj oprogramowania `JohnTheRipper`.

1.16 Użytkownik ma hasło (plik `z6.shadow`) składające się z 5 dowolnych znaków. Jakie to hasło? Do rozwiązania zadania użyj oprogramowania `JohnTheRipper`.

1.17 W pliku `z7.shadow` jest wiersz z hasłem użytkownika `user1`. Hasło ma charakter słownikowy, przy czym jest zapisane w taki sposób, że niektóre litery zamienione są na cyfry i tak: każde wystąpienie małego ***a*** zamienione jest na ***@***, małego ***i*** na ***1***, małego ***e*** na ***3***. Ponadto hasło ma jeszcze na początku i na końcu znak ***#*** (hash). Do rozwiązania zadania użyj oprogramowania `JohnTheRipper`.

1.18 Wykonaj zadanie **1.14** za pomocą narzędzia `hashcat`.

1.19 Wykonaj zadanie **1.15** za pomocą narzędzia `hashcat`.

1.20 Wykonaj zadanie **1.16** za pomocą narzędzia `hashcat`.

1.21 Wykonaj zadanie **1.17** za pomocą narzędzia `hashcat`.

Linki

- https://pl.wikipedia.org/wiki/Potok_(Unix)
- https://manpages.ubuntu.com/manpages/impish/pl/man1/head.1.html
- https://manpages.ubuntu.com/manpages/impish/pl/man1/cut.1.html
- https://manpages.ubuntu.com/manpages/impish/pl/man1/base64.1.html
- https://manpages.ubuntu.com/manpages/impish/pl/man1/tr.1.html
- http://linuxwiki.pl/wiki/Tr
- https://linux.die.net/man/4/urandom
- https://unix.stackexchange.com/questions/324209/when-to-use-dev-random-vs-dev-urandom
- https://tools.kali.org/password-attacks/crunch
- https://www.openwall.com/john/
- https://www.openwall.com/john/doc/EXAMPLES.shtml
- https://www.openwall.com/john/doc/OPTIONS.shtml
- https://www.varonis.com/blog/john-the-ripper/
- https://www.soisk-me.pl/klasa-iii-linux/plik-passwd-i-shadow
- https://chameleonstales.blogspot.com/2019/04/co-to-jest-za-plik-etcpasswd-i-etcshadow.html
- https://docs.python.org/3/library/hashlib.html
- https://hashcat.net/hashcat/
- http://manpages.ubuntu.com/manpages/impish/man1/fcrackzip.1.html
- https://www.cyberpratibha.com/blog/add-kali-linux-repository/
- https://hashcat.net/wiki/doku.php?id=example_hashes
- https://miloserdov.org/?p=5477
- https://countuponsecurity.files.wordpress.com/2016/09/jtr-cheat-sheet.pdf

Odpowiedzi

1.1 Generowanie hasła:
`openssl rand -base64 4`

Skrót MD5 hasła:
`openssl rand -base64 4 | openssl dgst -md5`

1.2 Generowanie hasła:
`cat /dev/urandom | base64 | head -n 1 | tr -dc '[:alnum:]' | cut -c -16`

Skrót MD5 hasła:
`cat /dev/urandom | base64 | head -n 1 | tr -dc '[:alnum:]' | cut -c -16 | openssl dgst -md5`

Kodowanie Base64:
`base64` lub `base64 -e`

Wypisz `n` pierwszych linii z pliku:
`head -n 1`

Usuń znaki (`-d`), użyj zbioru dopełnienia (`-c`), `[:alnum:]` - wszystkie litery i cyfry
`tr -dc '[:alnum:]'`

Pobierz określoną liczbę znaków:
`cut -c -16`

1.3 Generowanie haseł:
`crunch 3 3 1234567890 -o test.txt`

Haszowanie haseł:
`while read line; do echo -n ''$line'' | openssl dgst -sha1; done < test.txt`

1.4 Generowanie haseł:
`crunch 5 5 -t %a^b% -o test.txt`

Haszowanie haseł:
`while read line; do echo -n ''$line'' | openssl dgst -sha3-224; done < test.txt`

```
, for all uppercase letters
@ for all lowercase letters
% for all numeric characters
^ for all special characters
```

1.5 Generowanie haseł:

```
crunch 3 3 abc + 468 ?%: -t @%^
```

Haszowanie haseł:

```
while read line; do echo -n ''$line'' | openssl dgst -sha3-224; done < test.txt
```

1.6 Lokalizacja pliku ze słownikiem:

```
/usr/share/wordlists/rockyou.txt.gz
gzip -d rockyou.txt.gz
```

Złamanie hasha:

```
john --format=raw-md5 --wordlist=''wordlist.txt'' hash.txt
john --show --format=Raw-MD5 hash.txt
```

1.7 Lokalizacja pliku ze słownikiem:

```
/usr/share/wordlists/rockyou.txt.gz
gzip -d rockyou.txt.gz
```

Złamanie hasha:

```
john --format=raw-SHA256 --wordlist=''wordlist.txt'' hash.txt
john --show --format=Raw-SHA256 hash.txt
```

1.8 Wygenerowanie hasha pliku i porównanie z hashem na stronie:

```
openssl dgst -md5 gparted-live-1.3.1-1-amd64.iso
```

1.9 Skrypt:

```
#!/bin/env/python
import sys
```

```
import hashlib

h = hashlib.md5()
h.update(sys.argv[1].encode("utf-8"))
h.digest()
print(h.hexdigest())
```

1.10 Polecenia:

```
sha1sum test.txt
```

```
openssl dgst -sha1 test.txt
```

Skrypt:

```
#!/bin/env python
import sys
import hashlib

fname = sys.argv[1]

with open(fname) as f:
    lines = f.readlines()

fcontent = " ".join(lines)

h = hashlib.sha1()
h.update(fcontent.encode("utf-8"))

print(h.hexdigest())
```

1.11 Skrypt:

```
#!/bin/env/python
import hashlib

# algs = hashlib.algorithms_available
```

```
# print(algs)

cleartxt = "R3iSrSNmgU9SFHxVekUD".encode("utf-8")
hashtxt = "48cab4b54bef42fddaa6353c68a20b369f40026e"

for alg in hashlib.algorithms_available :
        try:
                h = hashlib.new(alg)
                h.update(cleartxt)
                if h.hexdigest() == hashtxt :
                        print(alg)
        except:
                pass
```

1.12 Polecenie:

`md5sum a.txt`, `md5sum b.txt`

W pliku `a.txt` za pomocą steganografii został ukryty inny tekst, dlatego nie są one takie same, chociaż na pierwszy rzut oka (`cat a.txt b.txt`) tak wyglądają:

```
stegsnow -C -m "UMCS 2021"b.txt a.txt
stegsnow -C a.txt
```

1.13 Polecenie:

```
hashcat -a 0 -m 0 --force ex1.6.txt /usr/share/wordlists/rockyou.txt
hashcat --show ex1.6.txt
```

`-a 0` - atak słownikowy, tu nazywany atakiem straight
`-m 0` - łamanie MD5
`--show` - pokaż wynik

Hasło: `8afa847f50a716e64932d995c8e7435a:princess`

1.14 Polecenie:

```
john z2.shadow --wordlist=/usr/share/wordlists/rockyou.txt
john z2.shadow --show
```

```
u1:google:16026:0:99999:7:::
u2:onelove:16026:0:99999:7:::

2 password hashes cracked, 0 left
```

1.15 Polecenie:

```
sudo gedit /etc/john/john.conf
```

```
[Incremental:ZADANIE]
File = $JOHN/utf8.chr
MinLen = 3
MaxLen = 3
CharCount = 196
```

```
john --incremental:ZADANIE z5.shadow
john z5.shadow --show
```

Hasło:

```
u9:123:16026:0:99999:7:::
u10:dhw:16026:0:99999:7:::

2 password hashes cracked, 0 left
```

1.16 Polecenie:

```
sudo john --make-charset=charset.chr
sudo gedit /etc/john/john.conf
```

```
[Incremental:ZADANIE2]
File = $JOHN/utf8.chr
MinLen = 5
MaxLen = 5
CharCount = 196
```

```
john --incremental:ZADANIE2 z6.shadow
john z6.shadow --show
```

Hasło:

1.17 Polecenie:

```
sudo gedit /etc/john/john.conf
```

```
[List.Rules:PRZYKLAD]
^[#]sa@se3si1$[#]
```

```
john --wordlist=/usr/share/wordlists/rockyou.txt z7.shadow --rules=PRZYKLAD
john z7.shadow --show
```

Hasło to:

```
#p1n3@ppl3#
```

1.18 Polecenia:

```
cp z2.shadow z8.shadow
```

Pozostawienie w pliku jedynie hasha

```
hashcat -m 1800 -a 0 z8.shadow /usr/share/wordlists/rockyou.txt
```

```
hashcat -m 1800 -a 0 z8.shadow /usr/share/wordlists/rockyou.txt --show
$6$jTfZyjJr$xzqXw3CFldUMA3JESiGMyE2N2jr9YEO62otJsiwLSWn9yWc/n0J0UszKzia/3IFnPh6c7ZSUaahgnRP/cuAYJ.:google
$6$pmqtr7vg$j3NPrwFohrNYY3VTTVAlYdWja.pNnrce7nNbP.Uiq8WksCUkfFFtRJ3udehVjk8rVpanXxYFmlHHMzouP2Iyv.:onelove
```

1.19 Polecenie:

```
cp z5.shadow z9.shadow
```

Pozostawienie w pliku jedynie hasha

```
hashcat -m 1800 -a 3 z9.shadow -i --increment-min=3 --increment-max=3
hashcat -m 1800 -a 3 z9.shadow -i --increment-min=3 --increment-max=3 --show
```

Hasło:

```
$6$3tVyi5Or$TvxtOe7bNTtJE7QmYSWC7HTLlxxhaOXHgDiOfRceOBnsFppOCue/zkz21gO7wPEUimLCEhd33oWF7HD4JUns21:123
$6$FPMAFD62$GhrAXEUS359cBy3bh0.uY6VKqZx/Byx91M9wyFPLMYMTltSbfT9WU6sFtjUHMl8rTfijWeSLplFDkyY6YY9WE.:dhw
```

1.20 Polecenie:

```
cp z6.shadow z10.shadow
```

Pozostawienie w pliku jedynie hasha

```
hashcat -m 1800 -a 3 z10.shadow -i --increment-min=5 --increment-max=5
hashcat -m 1800 -a 3 z10.shadow -i --increment-min=5 --increment-max=5 --show
```

1.21 Polecenie:

```
cp z7.shadow z11.shadow
gedit z11.shadow
```

Pozostawienie w pliku tylko hashy

```
nano rules.txt
^# sa@ se3 si1
hashcat -m 1800 -a 0 z11.shadow /usr/share/wordlists/rockyou.txt -r rules.txt hashcat -m 1800 -a
0 z11.shadow /usr/share/wordlists/rockyou.txt -r rules.txt --show
```