udp Escuela de Informática y Telecomunicaciones

"Criptografia y seguridad en redes"

Laboratorio 3: Hash

Sebastian González

Profesor: Nicolas Boettcher
Profesor lab:Victor Manriquez
Ayudantes:
Felipe Condore
Brayan Espina
Pablo Becerra

24 de junio del 2022

Índice

1. Desarrollo

1.1. Caso de uso

Se realizará un proyecto que logre enviar a la base de datos un hash, logrando darle al usuario seguridad, ya que gran cantidad de páginas chilenas envían sus datos en texto plano por lo cual, al realizar una intercepción en el tráfico, uno puede obtener la contraseña de un usuario muy fácilmente. Por lo cual con este nuevo sistema, la contraseña del usuario será enviada a través de este nuevo hash, provocando que al interceptar el tráfico sea mucho más difícil de averiguar esta contraseña y también logrando ser más difícil de romper a través de fuerza bruta.

Por otro lado este sistema de hash solucionara los problemas que tienen las contraseñas de los usuarios, donde la gran mayoría usa solo números,fechas de cumpleaños, nombres, etc. las cuales son muy fáciles de descubrir, logrando incluso que el usuario tenga más seguridad.

1.2. Funcionamiento del codigo

1.2.1. Código

Para que la contraseña sea transformada a un hash se utiliza "hashhçomo función principal, la cual dentro tiene las distintas funciones que crean el hash.

```
def hashh(passw):
  #tama o de la contrase a texto plano
  tama opassw=len(passw)

  #pasar al hexadecimal
  hexad=hexadecimal(passw)

  #cambiar las letras del hexadecimal a numero
  textonumeros=letrasxnumeros(hexad)

  #se aumentan el numero  multiplicandolo por el tama o de la contrase a
  newtextonumeros=aumentarTama o(textonumeros,tama opassw)
  convertirHash =convertir(newtextonumeros)
  #print(passw+" - "+convertirHash)
  return convertirHash
```

- La primera función es hexadecimal que realiza un cambio de los caracteres a hexadecimal.

```
def hexadecimal(passw):
  hexa=passw.encode('utf-8').hex()
  return hexa
```

- La segunda función recibe la contraseña en hexadecimal y cambia las letras por números, las letras cambiadas siempre serán de la a hasta la f. Una vez remplazada quedara un numero bastante grande, por ejemplo: 909091292391231901329.

```
def letrasxnumeros(palabraHexa):
  palabraHexa=str(palabraHexa)
  palabraHexa=palabraHexa.replace("a","1")
  palabraHexa=palabraHexa.replace("b","2")
  palabraHexa=palabraHexa.replace('c','3')
  palabraHexa=palabraHexa.replace("d","4")
  palabraHexa=palabraHexa.replace("e","5")
  palabraHexa=palabraHexa.replace("f","8")
  return palabraHexa
```

- Esta tercera función recibe el número creado en la anterior función y es multiplicado por el largo de la contraseña puesta por el usuario, esta se multiplicará hasta que el dígito formado tenga un largo mayor a 110 caracteres.

```
def aumentarTama o(palabraNumber,tama o):
  true=False
  conta=0
  while true!=True:
    if (len(palabraNumber)>110):
      if conta==0:
        palabraNumber=int(palabraNumber)
```

```
        palabraNumber=palabraNumber*tama o
        palabraNumber=str(palabraNumber)
      break
    palabraNumber=int(palabraNumber)
    palabraNumber=palabraNumber*tama o
    palabraNumber=str(palabraNumber)
    conta=conta+1

  return palabraNumber
```

- Por último la cuarta función tiene una matriz de 10x10 la cual tiene 90 tipos distintos de caracteres(base 90). Su función consiste en tomar el valor numérico que tiene más de 110 caracteres, con el cual se comienza a tomar pares de números donde un dígito corresponderá a una columna y el otro a una fila de la matriz base 90 entregando un carácter, este proceso se repetirá 55 veces obteniendo así el hash.

```
def convertir(hexaxhash):
  base90=[["0","1","2","3","4","5","6","7","8","9"],
          ["a","A","b","B","c","C","d","D","e","E"],
          ["f","F","g","G","h","H","i","I","j","J"],
          ["k","K","l","L","m","M","n","N","o","O"],
          ["p","P","q","Q","r","R","s","S","t","T"],
          ["u","U","v","V","w","W","x","X","y","Y"],
          ["z","Z","  ","  ","&","$","#","=","?","  "],
          ["+","-","%","(",")","[","]","{","}","."],
          [",","_","*","/","<",">",";",":","  ","!"],
          ["0","1","2","3","4","5","6","7","8","9"],
         ]
  palabra=""
  for i in range(0,110,2):
    palabra=palabra+base90[int(hexaxhash[i])][int(hexaxhash[i+1])]
  return palabra
```

1.3. Condiciones del gremio

El gremio solicita que el hash tenga las siguientes codiciones, donde a traves de images se puede comprobar que el hash creado cumple lo requerido.

- El número de caracteres final del texto procesado, debe ser fijo y no menor de 55 caracteres. Este número no debe variar, aunque el texto de entrada sea más largo.

```
Escriba su contraseña: hola
hola - CoO1Y97$;En?TDE})t!wS)RN8nO#ig[2<;rxOIs37lVgL96j=)4M1KB
CoO1Y97$;En?TDE})t!wS)RN8nO#ig[2<;rxOIs37lVgL96j=)4M1KB
```

Figura 1

- Se requiere que el programa pueda recibir como entrada, tanto un String o texto mediante STDIN, cómo archivos. En este último caso, se pide que el programa pueda trabajar con el contenido de los archivos (procesando línea por línea el contenido del archivo), y con el archivo en su conjunto.

```
1- Escribir una contraseña
2- Agregar un archivo con contraseñas(.txt)
3- Salir
1
Escriba su contraseña: hola
hola - CoO1Y97$;En?TDE})t!wS)RN8nO#ig[2<;rxOIs37lVgL96j=)4M1KB
CoO1Y97$;En?TDE})t!wS)RN8nO#ig[2<;rxOIs37lVgL96j=)4M1KB
**********
```

Figura 2

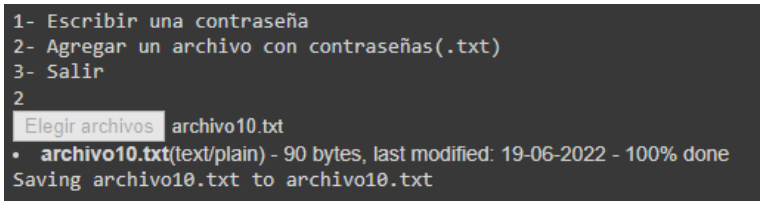

Figura 3

- Cualquier ligero cambio al texto de entrada, debe cambiar el resultado final del algoritmo, aunque se debe mantener el mismo valor de Hash para el mismo String ingresado (Aunque sea por un tiempo determinado).

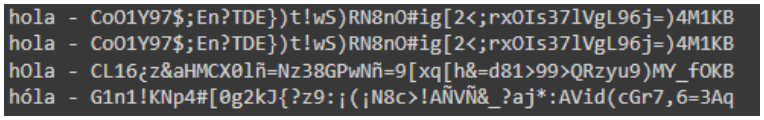

Figura 4

- El procesamiento del texto de entrada, debe ser rápido, no debe tomar mucho tiempo, independiente de la cantidad de texto de entrada. * Se recomienda el uso de operaciones matemáticas al texto de entrada, a fin de optimizar la velocidad de procesamiento del texto de entrada*

Adicionalmente se requiere que posea a lo menos dos opciones:

- una que procese la entrada y calcule el Hash de estas entradas (sea por STDIN, cómo mediante un archivo).
- Y otra opción que sólo calcule la entropía del texto de entrada. En este último caso, debe arrojar mediante STDOUT que entropía posee cada texto de entrada (El formato de la salida a STDOUT, debe mostrar tanto el texto analizado, cómo la entropía calculada separada por un delimitador que permita diferenciar los diferentes campos).

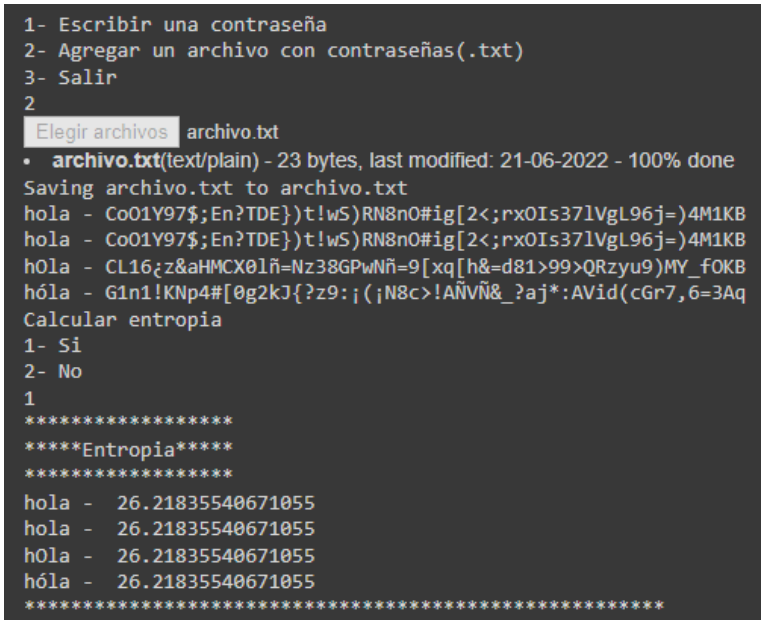

Figura 5

1.4. Tablas comparativas

HASH	TAMAÑO	BASE	ENTROPIA
MD5	32	16	128
SHA1	40	16	160
HASHH	55	90	357.05
SHA256	64	16	256

Cuadro 1: Tabla entropia

Palabra	Entropia	Tiempo hash (s)	Tiempo MD5 (s)	Tiempo SHA1 (s)	Tiempo SHA256 (s)
123456	39.5097754	0.00029969215	1.478195190e-05	4.52995300e-06	6.9141387e-06

Cuadro 2: Tabla de tiempos de hashs 1 entrada

Palabra	Entropia	Tiempo hash (s)	Tiempo MD5 (s)	Tiempo SHA1 (s)	Tiempo SHA256 (s)
12345	32.9248125	0.000212907	1.2636184e-05	3.5762786e-06	4.053115844e-06
123456789	59.2646625	0.000153303	8.1062316e-06	2.622604e-06	3.099441528e-06
password	52.6797000	0.000156641	3.0994415e-06	1.9073486e-06	1.907348632e-06
iloveyou	52.6797000	0.000143289	1.6689300e-06	9.5367431e-07	9.536743164e-07
princess	52.6797000	0.000140190	2.1457672e-06	9.5367431e-07	1.430511474e-06
1234567	46.0947375	0.000147342	1.4305114e-06	9.5367431e-07	1.192092895e-06
rockyou	46.0947375	0.000186443	3.5762786e-06	9.5367431e-07	1.907348632e-06
12345678	52.67970005	0.000188827	5.2452087e-06	1.9073486e-06	2.622604370e-06
abc123	39.5097750	0.000167608	2.6226043e-06	1.1920928e-06	1.192092895e-06
nicole	39.5097750	0.000177145	2.3841857e-06	1.4305114e-06	1.430511474e-06

Cuadro 3: Tabla de tiempos de hashs 10 entrada

Palabra	Entropia	Tiempo hash (s)	Tiempo MD5 (s)	Tiempo SHA1 (s)	Tiempo SHA256 (s)
daniel	39.5097750	0.000191926	2.1457672e-06	9.536743e-07	1.19209289e-06
babygirl	52.6797000	0.000144004	1.4305114e-06	9.536743e-07	9.53674316e-07
monkey	39.5097750	0.000169754	1.4305114e-06	9.536743e-07	1.19209289e-06
lovely	39.5097750	0.000189065	2.1457672e-06	1.192092e-06	1.43051147e-06
jessica	46.0947375	0.000251054	5.9604644e-06	2.145767e-06	3.09944152e-06
654321	39.5097750	0.000185489	6.9141387e-06	1.907348e-06	3.09944152e-06
michael	46.0947375	0.000166416	4.2915344e-06	1.430515e-06	2.14576721e-06
ashley	39.5097750	0.000177860	2.6226043e-06	1.668930e-06	1.19209289e-06
qwerty	39.5097750	0.000172138	2.1457672e-06	1.430511e-06	1.19209289e-06
111111	39.5097750	0.000278711	1.1920928e-05	3.576278e-06	3.33786010e-06
iloveu	39.5097750	0.000202178	6.9141387e-06	2.384185e-06	3.09944152e-06
000000	39.5097750	0.000206947	8.5830688e-06	2.861022e-06	2.62260437e-06
michelle	52.6797000	0.000162124	6.9141387e-06	2.145767e-06	2.62260437e-06
tigger	39.5097750	0.000204086	8.5830688e-06	2.384185e-06	2.86102294e-06
sunshine	52.6797000	0.000183343	1.2159347e-05	3.337860e-06	3.57627868e-06
chocolate	59.264662	0.000165224	1.1920928e-05	3.337860e-06	4.29153442e-06
password1	59.2646625	0.000246524	1.3351440e-05	3.814697e-06	4.29153442e-06
soccer	39.5097750	0.000210762	8.3446502e-06	2.861022e-06	2.86102294e-06
anthony	46.0947375	0.000187158	9.2983245e-06	2.622604e-06	4.29153442e-06
friends	46.0947375	0.000196218	4.0531158e-06	1.430511e-06	2.14576721e-06

Cuadro 4: Tabla de tiempos de hashs 20 entrada

Palabra	Entropia	Tiempo hash (s)	Tiempo MD5 (s)	Tiempo SHA1 (s)	Tiempo SHA256 (s)
butterfly	59.264662	0.0001728534	8.10623168e-06	2.14576721e-06	3.33786010e-06
purple	39.509775	0.0001947879	1.47819519e-05	2.86102294e-06	3.33786010e-06
angel	32.924812	0.0001902580	3.33786010e-06	1.66893005e-06	1.66893005e-06
jordan	39.509775	0.0001850128	1.07288360e-05	3.09944152e-06	3.57627868e-06
liverpool	59.264662	0.0001831054	8.34465026e-06	2.62260437e-06	2.86102294e-06
justin	39.509775	0.0001845359	1.09672546e-05	3.09944152e-06	4.291534423e-06
loveme	39.509775	0.0002110004	1.02519989e-05	3.09944152e-06	3.576278686e-06
fuckyou	46.094737	0.0001733303	3.57627868e-06	1.66893005e-06	1.668930053e-06
123123	39.509775	0.0001792907	3.33786010e-06	1.19209289e-06	1.192092895e-06
football	52.679700	0.0001976490	1.26361846e-05	3.33786010e-06	3.814697265e-06
secret	39.509775	0.0001869201	1.04904174e-05	3.09944152e-06	3.576278686e-06
andrea	39.509775	0.0002787113	2.69412994e-05	5.24520874e-06	5.245208740e-06
carlos	39.50977	0.0003452301	8.34465026e-06	3.09944152e-06	3.337860107e-06
jennifer	52.67970	0.0002615451	6.19888305e-06	3.09944152e-06	3.337860107e-06
joshua	39.50977	0.0002601146	5.48362731e-06	2.86102294e-06	2.861022949e-06
bubbles	46.09473	0.0001988410	5.96046447e-06	1.90734863e-06	2.384185791e-06
1234567890	65.84962	0.0001437664	3.33786010e-06	1.66893005e-06	1.430511474e-06
superman	52.67970	0.0001621246	7.15255737e-06	2.14576721e-06	2.384185791e-06
hannah	39.50977	0.0001938343	8.34465026e-06	2.14576721e-06	2.861022949e-06
amanda	39.50977	0.0001926422	9.05990600e-06	2.38418579e-06	3.337860107e-06
loveyou	46.09473	0.0001628398	3.57627868e-06	1.43051147e-06	1.6689300535e-06
pretty	39.50977	0.0002012252	7.39097595e-06	2.14576721e-06	2.622604370e-06
basketball	65.84962	0.0001394748	3.09944152e-06	1.430511475e-06	1.43051147e-06
andrew	39.50977	0.0002131462	9.29832458e-06	2.622604370e-06	2.861022949e-06
angels	39.50977	0.0001986026	1.04904174e-05	2.622604370e-06	3.576278686e-06
tweety	39.50977	0.000206470	8.34465026e-06	2.861022949e-06	2.861022949e-06
flower	39.50977	0.000185489	3.57627868e-06	1.430511474e-06	1.668930053e-06
playboy	46.09473	0.000170946	7.15255737e-06	2.622604370e-06	2.622604370e-06
hello	32.92481	0.000211238	6.91413879e-06	2.622604370e-06	2.622604370e-06
elizabeth	59.26466	0.000164747	2.62260437e-06	1.668930053e-06	1.192092895e-06
hottie	39.50977	0.000194787	8.10623168e-06	2.622604370e-06	2.622604370e-06
tinkerbell	65.84962	0.000165939	1.14440917e-05	3.337860105e-06	3.576278686e-06
charlie	46.09473	0.000161170	3.57627868e-06	1.430511474e-06	1.192092895e-06
samantha	52.67970	0.000159502	2.38418579e-06	1.668930053e-06	1.192092895e-06
barbie	39.50977	0.000183343	2.622604375e-06	1.192092895e-06	1.430511474e-06
chelsea	46.09473	0.000186920	1.072883605e-05	3.099441528e-06	3.576278686e-06
lovers	39.50977	0.000191211	4.053115844e-06	1.90734863e-06	1.668930053e-06
teamo	32.92481	0.000192642	2.622604370e-06	1.430511474e-06	1.192092895e-06
jasmine	46.09473	0.000172138	9.059906005e-06	2.861022949e-06	2.861022949e-06
brandon	46.09473	0.000172376	7.629394531e-06	2.861022949e-06	3.337860107e-06
666666	39.50977	0.000202894	6.914138793e-06	2.384185791e-06	2.861022949e-06
shadow	39.50977	0.000177621	2.622604370e-06	1.668930053e-06	1.430511474e-06
melissa	46.09473	0.000171661	6.914138793e-06	2.861022949e-06	2.861022949e-06
eminem	39.50977	0.000179767	2.360343933e-05	2.622604370e-06	2.861022949e-06
matthew	46.09473	0.000168800	7.390975952e-06	2.622604370e-06	3.337860107e-06
robert	39.50977	0.000188589	3.099441528e-06	1.668930053e-06	1.430511474e-06
danielle	52.67970	0.00015234	2.384185791e-06	9.536743164e-07	1.192092895e-06
forever	46.09473	0.000267982	1.096725463e-05	3.337860107e-06	3.099441528e-06
family	39.50977	0.000206232	5.72204589e-06	1.668930053e-06	1.907348632e-06
jonathan	52.67970	0.000187158	9.775161743e-06	2.145767211e-06	2.384185791e-06

Cuadro 5: Tabla de tiempos de hashs 50 entrada

1.5. Analisis de tablas

1.5.1. Entropia

Se puede observar en en el cuadro 1 que las entropías dadas por cada hash son totalmente distintas, esto es debido al tamaño y la base que utiliza cada uno de los hash.

- En el caso de las bases se puede observar que MD5, SHA1 Y SHA256 utilizan una base 16 en cambio HASHH tiene una base 90 logrando aumentar la entropía de este, debido a que hay 90 posibles símbolos en un dígito, en cambio en los otros hash su base es de solo 16 símbolos, haciendo que sea más fácil encontrar una posible combinación en una contraseña.
- Se puede observar que el tamaño de cada hash va incrementado, lo cual hace que el que tenga el tamaño más grande sea el más seguro. Al complementarlo con las bases provoca que exista una mayor cantidad de combinaciones en ella haciendo mucho más difícil de romper el hash.

Por lo tanto el hash creado(HASHH) es mucho más seguro de usar debido a que tiene una mayor entropía y esto es causado por la base que utiliza siendo mayor que los otros hash comparados. En el caso de cambiar las bases de los otros hash a 90 provocaría un cambio siendo el de mayor seguridad SHA256, esto causado por la longitud en el que entrega el hash.

Por último MD5 y SHA1 al ser tan débiles en entropía y además por haber sido descubierto su funcionamiento, estos puede ser utilizado para verificar archivos y ver si están correctamente descargados comparando el hash original con el descargado. Por otro lado SHA256 y el hash creado (HASHH), puede ser utilizados para temas de seguridad e integridad de los datos, donde exista información más sensible, como por ejemplo en credenciales, esto es debido a que aún no han sido rotos, y por otro lado su entropía es bastante grande.

1.5.2. Rendimiento

Al observar las tablas de rendimiento se puede apreciar a continuación una lista con los hash el cual demora mas y cual menos.

- HASHH
- MD5
- SHA1
- SHA256

El HASHH es el que demora más, esto puede ser causado por que el hash MD5 y los SHA utilizan bits para lograr su funcionamiento, provocando que al realizar sus operaciones este sea más rápido de realizar, en cambio el HASHH tiene que convertir los números, caracteres y dígitos en bits cada vez que realiza una operación.

Por otro lado MD5 es más lento que SHA1, esto es causado porque en el algoritmo de md5 realiza una mayor cantidad de iteraciones causando un mayor retraso en el código .

2. Repositorio

- https://github.com/wolfzart/Lab-3-cripto

3. Bibliografía

- https://github.com/Zunawe/md5-c/blob/main/md5.c.
- https://github.com/ajalt/python-sha1/blob/master/sha1.py