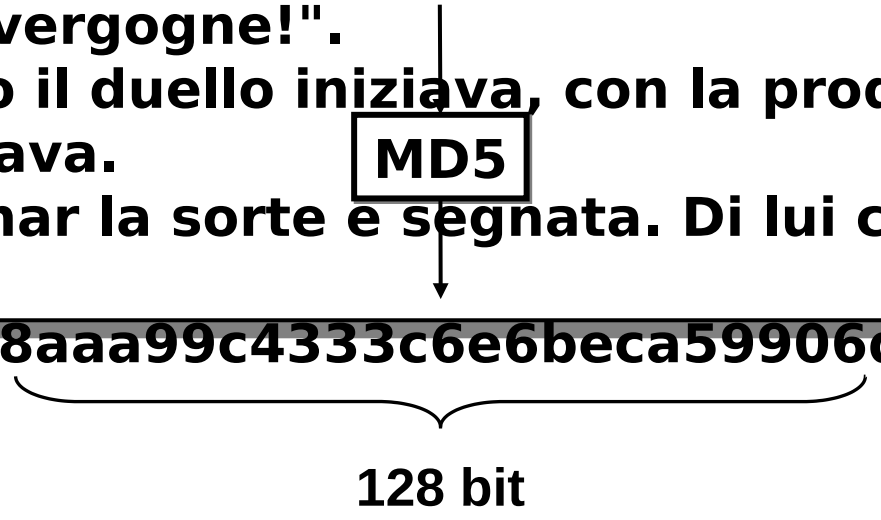
CyberChallengeIT 2020
Università degli studi di Pisa

Funzioni Hash e Steganografia

Esempio di Hash

La storia di Ragnar inizia così, senza che nessuno lo attendesse quel dì.
Entrò tracotante brandendo la lama, urlando spavaldo di gloria e di fama.
Ma poi tutt'un tratto il tono scemò, quando di Matilda lo sguardo incrociò.
"Siam stanchi di udire siffatte menzogne, orsù diamo un limite a queste vergogne!".
Così d'un baleno il duello iniziava, con la prode Matilda che parava e affondava.
Del povero Ragnar la sorte è segnata. Di lui ci rimane una testa mozzata!



Alcuni utilizzi per le Hash

Le Funzioni hash sono utilizzate in vari ambiti, in particolare, a noi ne interessano due:

1. Controllo sull'integrità di un messaggio:
   - Per evitare che il messaggio arrivi corrotto si attacca al messaggio il digest del messaggio stesso. Il ricevente estrae il messaggio, ne calcola il digest e poi lo confronta con quello che ha ricevuto insieme al messaggio: se sono uguali, il ricevente intuisce che non ci sono stati problemi (o attacchi) durante la trasmissione. *Se riuscissimo a cambiare sia il messaggio che il digest corrispondente, potremmo ingannare il ricevente.*
2. Password:
   - I siti affidabili, per questioni di sicurezza, non memorizzano mai la vostra password in chiaro sui loro server, ma ne memorizzano il digest. Questo viene fatto perché **dovrebbe** essere molto difficile risalire al testo originale partendo dal digest. Anche gli archivi protetti da password ne memorizzano il digest!
     Se riuscissimo a recuperare un DB di digest di password (o il digest che protegge un archivio) che usi una funzione hash debole, potremmo riuscire a recuperare qualche coppia email-

JohnTheRipper

JohnTheRipper è un tool a linea di comando utile per crackare hash di diversi algoritmi (MD5, SHA1…)

Potete installarlo con aptitude o scaricare le sorgenti da github

```
apt-get install aptitude
sudo aptitude install john
```

Con ‘john --help’ potete vedere tutte le sue opzioni.

JohnTheRipper ha 2 principali modalità di attacco:

JohnTheRipper – Wordlist Attack

L’idea è abbastanza intuitiva. Fornite a john un dizionario (wordlist) di parole che voi vi aspettiate possano aver generato le hash che state cercando di rompere.
Le parole devono essere in un file di testo, una per riga. È consigliabile ordinarle alfabeticamente.
Esistono molti dizionari pronti all’uso, il più famoso è rockyou.txt .

È possibile provare anche delle varianti delle parole, dette «mangled», storpiate.
Esempio di parola storpiata: topolino t0p0l1n0

JohnTheRipper – Wordlist Attack

Non c’è bisogno di inserire tutte le possibili varianti storpiate delle parole all’interno della wordlist. Tramite un set di regole (per john, ovviamente, RULES) potete decidere quali storpiature applicare al vostro dizionario.

Potete decidere di fare più set di regole, inserendole alla fine del file ‘/etc/john/john.conf’.

John ha dei set predefiniti di regole, la più potente dei quali è la JUMBO rule.
Inoltre, è possibile definire delle regole personalizzate.

JohnTheRipper – Wordlist Attack

```
[List.Rules:Example]
cAz"[0-9]"
```

Questa regola rende maiuscola la prima lettera delle parola all'interno della wordlist, e «postfigge» (append) un numero alla stringa così generata. In questo modo, se nella wordlist ci fosse soltanto la parola «ciao», john testerebbe:

- ciao
- Ciao0
- Ciao1
- Ciao2
- Ciao3
- Ciao4
- Ciao5
- Ciao6
- Ciao7
- Ciao8
- Ciao9

JohnTheRipper – Incremental Attack

Questa è la modalità bruteforce, proverà ogni singola combinazione del set di caratteri fornito. Per usare questa modalità bisogna definire dei parametri (lunghezza massima della chiave, set di caratteri consentiti ecc…). Queste definizioni si trovano ne file di configurazione, sotto la voce "[Incremental:XXXX]" dove al posto delle XXXX dovete inserire una stringa identificativa, che sarà il nome col quale invocherete la modalità incrementale.

Esistono diverse modalità pre-impostate:

- "ASCII" (tutti i 95 caratteri stampabili ASCII)
- "Alnum" (tutti i 62 caratteri alfanumerici)
- "Alpha" (tutte le 52 lettere, maiuscole e minuscole)
- "LowerNum" (lettere minuscole e cifre, 36 caratteri totali)
- "UpperNum" (lettere maiuscole e cifre, 36 caratteri totali)
- "LowerSpace" (lettere minuscole più lo spazio, 27 caratteri totali)
- "Lower" (lettere minuscole, 26 caratteri totali)

JohnTheRipper - sources

Cracking modes:
https://www.openwall.com/john/doc/MODES.shtml

Sintassi per la composizione delle regole:
https://www.openwall.com/john/doc/RULES.shtml

Esempi di utilizzo di JohnTheRipper:
https://www.openwall.com/john/doc/EXAMPLES.shtml

JohnTheRipper cheatsheet:
https://countuponsecurity.files.wordpress.com/2016/09/jtr-cheat-sheet.pdf

Introduzione alla creazione di regole personalizzate:
https://www.gracefulsecurity.com/custom-rules-for-john-the-ripper/

Esempio di 3 rule set (potete copiare ed incollare nel vostro john.conf):
https://www.gracefulsecurity.com/custom-rules-for-john-the-ripper-

JohnTheRipper – script di “estrazione hash”

Nell’archivio scaricabile da github c’è la cartella run.

Al suo interno potete trovare numerosi script in python utili per estrarre le hash di password che proteggono file in diversi formati.

Questi script appaiono nel formato

<estensione>2john.py

Tra i più utili troviamo

zip2john.py
pdf2john.py
rar2john.py
7z2john.py

JohnTheRipper – Esempio cracking di Zip

Scaricate il file zip «April29» da telegram.

Da terminale:

```
zip2john april29.zip > hash_zip.txt
```

Scrive nel file hash_zip l'hash della password per sbloccare lo zip.

```
john hash_zip.txt > solution.txt
```

Esegue il cracking in modalità default, prima con la wordlist inclusa (molto piccola), poi in modalità incrementale ASCII. La stringa che genera quell'hash è salvata in solution.txt

Gli stessi passaggi possono essere fatti con altri tipi di file utilizzando altri script che trovate nella cartella run.

JohnTheRipper – Shadow file

Il file /etc/shadow contiene informazioni riguardanti gli utenti del sistema, uno per riga, la quale è composta da 9 campi separati dal carattere «duepunti» ‘:’
I campi indicano, nell'ordine:

1. Il nome utente.

2. La password crittografata, oppure un asterisco per indicare che non è possibile effettuare direttamente il login come quell’utente.

3. La data dell'ultima modifica della password, in giorni trascorsi dal 1º gennaio 1970.

4. Il minimo di giorni che devono trascorrere prima di poter modificare la password.

5. Il massimo di giorni che devono intercorrere tra una modifica e l'altra di una password.

6. I giorni di preavviso della scadenza della password.

7. I giorni dopo i quali una password scaduta comporta la disattivazione dell'account.

JohnTheRipper – Esempio cracking di password unix

Supponete di avere accesso ad un computer condiviso con altri utenti.
Potete utilizzare john per trovare le loro password!

Da terminale (aggiungiamo un utente fasullo):

```
useradd -r fantoccio
passwd fantoccio
```

Unix vi chiederà di inserire una password, vi suggerisco di inserirne una facile, come «hello» oppure «1612», per non perdere troppo tempo in questa prova.

```
john /etc/shadow > usersPwd.txt
```

Nel file usersPwd.txt potrete trovare le password che john è riuscito a crackare.

Hashcat

Hashcat è un famoso software utilizzato per crackare diversi tipi di hash, supporta esecuzione su GPU ed è disponibile su diversi sistemi operativi.
Installazione: `sudo aptitude install hashcat`
Le opzioni più importanti per hashcat sono:

-a attack mode, 3 per bruteforce

-m hash type, 0 per md5

--force per forzare l'esecuzione su alcune piattaforme

-w per settare le performance 1 basse, 4 massime

-o specifica il file di output dove memorizzare i risultati.

-i attacco incrementale, seguiti da parametri:
--increment-min <min> --increment-max <max>

Hashcat - maschere

Oltre alle opzioni viste, hashcat ha un'altra feature molto utile, anche se spesso poco utilizzabile: le maschere.

Sapere la struttura di una password, può essere molto utile. Per esempio, provate ad hashare con md5 la password jack1970

Ed invocate hashcat con il comando

```
hashcat -m 0 -a 3 -w 4 --force e4340d4fdb5705d65b8284802b683727
```

Quanto ci mette? Troppo…

Adesso provate ad invocare lo stesso comando, ma inserendo alla fine la seguente stringa:

```
?l?l?l?l?d?d?d?d
```

Il cracking ci impiega sensibilmente meno! In 3 minuti ha trovato la

Hashcat - charset

Le maschere funzionano con set di caratteri:

?l → abcdefghijklmnopqrstuvwxyz
?u → ABCDEFGHIJKLMNOPQRSTUVWXYZ
?d → 0123456789
?h → 0123456789abcdef
?H → 0123456789ABCDEF
?s → !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
?a → ?l?u?d?s
?b → 0x00 - 0xff

Potete anche definire i vostri custom character set (da 1 a 4). Se adesso provate a fare l'hash della stringa «Jack1970», potete utilizzare il seguente comando:

```
hashcat <varie opzioni ed hash> -4 ?u?l ?4?l?l?l?d?d?d?d
```

Per vedere tutte le password crackate da hashcat:

```
cat .hashcat/hashcat.potfile
```

Corkami/collisions

Corkami è un insieme di script in python che permette di modificare leggermente due file in modo che generino entrambi lo stesso hash MD5 (collisione).

Potete scaricare Corkami da:
https://github.com/corkami/collisions

E trovate gli script in
collisions-master/scripts

Corkami/collisions funzionamento

Scaricate da internet due immagini qualsiasi in formato png. Controllatene l'hash MD5. A meno di una straordinaria coincidenza, dovrebbero tornare diversi!

Adesso eseguite il comando:
`python png.py file1.png file2.png`

Il programma dovrebbe generare due file «collision1.png» e «collision2.png»

Se controllate le hash di queste due immagini risultanti, vedrete che saranno uguali!

Potete eseguire lo stesso procedimento con file in formato jpg, pdf, ed mp4.

STEGANOGRAFIA

Tecniche Steganografiche: Steganografia LSB

Si basa sulla teoria secondo la quale l'aspetto di un'immagine digitale ad alta definizione non cambia se i colori vengono modificati in modo impercettibile.

Ogni pixel è rappresentato da un colore differente, cambiando i bit meno significativi di ogni pixel, il singolo colore non risulterà variato in modo significativo e il contenuto dell'immagine sarà preservato nonostante questa manipolazione.

Nello standard bitmap a 24 bit di profondità ogni pixel è codificato con:

- 1 byte per codificare il rosso (valore 0 – 255)
- 1 byte per codificare il verde (valore 0 – 255)
- 1 byte per codificare il blu (valore 0 – 255)

Tecniche Steganografiche: Steganografia LSB

Il colore rosso, per esempio è codificato con la tripletta (255,0,0) mentre il colore lavanda è codificato con la tripletta (230, 230, 250).

Rosso

R	1	1	1	1	1	1	1	1
G	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0

Lavanda

R	1	1	1	0	0	1	1	0
G	1	1	1	0	0	1	1	0
B	1	1	1	1	1	0	1	0

ENC_LSB 2 = RES

R	1	1	1	1	1	1	1	1
G	0	0	0	0	0	0	1	1
B	0	0	0	0	0	0	1	1

DEC_LSB 2 = RES

R	1	1	0	0	0	0	0	0
G	1	1	0	0	0	0	0	0
B	1	1	0	0	0	0	0	0

Tecniche Steganografiche: Steganografia LSB

Il colore rosso, per esempio è codificato con la tripletta (255,0,0) mentre il colore lavanda è codificato con la tripletta (230, 230, 250).

Rosso

R	1	1	1	1	1	1	1	1
G	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0

Lavanda

R	1	1	1	0	0	1	1	0
G	1	1	1	0	0	1	1	0
B	1	1	1	1	1	0	1	0

ENC_LSB 4 = RES

R	1	1	1	1	1	1	1	0
G	0	0	0	0	1	1	1	0
B	0	0	0	0	1	1	1	1

DEC_LSB 4 = RES

R	1	1	1	0	0	0	0	0
G	1	1	1	0	0	0	0	0
B	1	1	1	1	0	0	0	0

Tecniche Steganografiche: Steganografia LSB

Link utili:

Steganografia LSB, esempio:
https://www.petitcolas.net/steganography/image_downgrading/

Steganografia LSB guida python:
https://towardsdatascience.com/steganography-hiding-an-image-inside-another-77ca66b2acb1

Pillow

Pillow è un modulo python che vi permette di manipolare immagini di diversi formati.

`Link utili:`
https://pillow.readthedocs.io/en/stable/index.html
https://pillow.readthedocs.io/en/stable/handbook/tutorial.html

Stegsolve

Stegsolve è un'applicazione java utile per analizzare i «piani» di un'immagine in vari formati: png, jpeg, gif, bitmap...

Per scaricare l'applicazione:

```
wget http://www.caesum.com/handbook/Stegsolve.jar -O stegsolve.jar
```

Per renderla eseguibile:

```
chmod +x stegsolve.jar
```

Per Eseguirla:

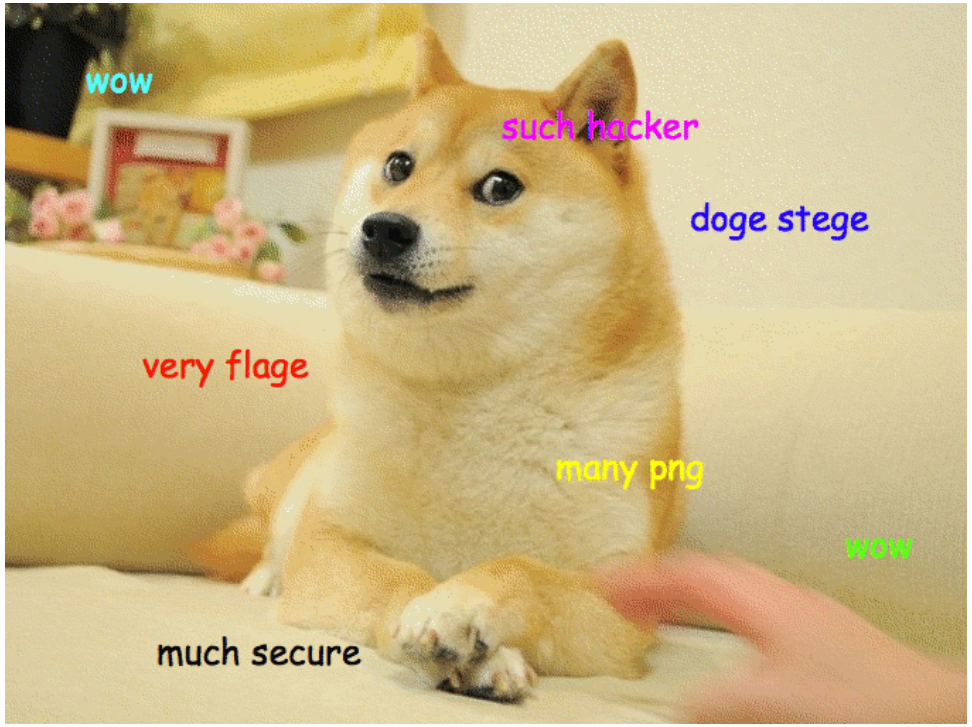
```
java -jar stegsolve.jar
```

GIMP

GIMP fa, tra le altre cose, un lavoro molto simile a stegsolve.
Per installare GIMP:
apt-get install gimp

Utilizzo:

- Nella cartella doge c’è doge.png
- Apritelo in GIMP.
- Navigate in colors set colormap
- Provatele tutte finché non trovate



Steghide

Steghide è un'applicazione a linea di comando tramite la quale si possono nascondere informazioni in immagini e tracce audio dei seguenti formati JPEG,JPG,BMP,WAV,AU.

Installazione:

```
apt-get install steghide
```

Informazioni:

```
steghide --help
```

Come esempio, provate a scaricare un'immagine e create un file di testo «segreto.txt».
Per creare un'immagine steganografica:

```
steghide embed -cf immagine.jpg -ef segreto.txt
```

Il programma vi chiederà anche di inserire una password.
ATTENZIONE! L'IMMAGINE VERRA' MODIFICATA!

Per estrarre un segreto:

Stegcracker

Stegcracker è un’applicazione a linea di comando pensata per estrarre e crackare le password usate con steghide.

Installazione:

```
Sudo pip3 install stegcracker
```

Usarlo è semplice:

```
stegcracker <immagine> <dizionario>
```

Zsteg

Zsteg è uno strumento per scoprire informazioni nascoste in PNG e BMP.

Si può scaricare zsteg da https://github.com/zed-0xff/cd ../..
Ed installare:

```
gem install zsteg
```

Per informazioni su zsteg e le sue opzioni:

```
zsteg -h
```

Per eseguire zsteg basta semplicemente invocarlo con il nome del file da analizzare:

```
zsteg flower_rgb3.png
```

Se utilizzate il comando di prima con l'opzione «–a» troverete più risultati. Zsteg –a infatti utilizza tutti i metodi conosciuti a prescindere dai risultati che ottiene.

Binwalk

Binwalk è uno strumento di steganalisi che rileva archivi o altri file «embeddati», inclusi, in altri file.

Lo potete installare con:
`sudo aptitude install binwalk`

Con il comando
`binwalk «nomefile»`
Il programma elenca i contenuti nascosti nel file

Con il comando
`binwalk -e «nomefile»`
Il programma cerca di estrarre i contenuti nascosti.