

### EJERCICIO 2.3.1. Ataques de Fuerza bruta y elaboración de diccionarios con comandos especiales.

- Creación de diccionarios a partir de una página web. Para ilustrar el proceso, vamos a utilizar una página web de referencia con el comando `cewl`. Este comando permite crear un diccionario con todas las palabras encontradas en la página web objetivo y que podemos registrarlo en un fichero: `cewl https://www...> lista.txt`
- ☑ Elegir una web cualquiera y hacer una prueba y contar las líneas dentro del fichero: `cat fichero.txt | wc -l`

```
[vicky@parrot]~$ cewl https://gobrunch.com > lista.txt
[vicky@parrot]~$ cat lista.txt | wc -l
919
```

- Existen recursos web especializados en diccionarios. Como es el caso de la página: <ftp://ftp.openwall.com/pub/wordlists> ... o bien, diccionarios más avanzados como la página: <https://packetstormsecurity.com/Crackers/wordlists>
- Además del uso de diccionarios comunes podemos utilizar una herramienta llamada `crunch`. Es un guión donde especificamos los caracteres que queremos en las contraseñas, y crea todas las combinaciones posibles de estas contraseñas.
- El formato del comando para usar `crunch` es:  
☉ `crunch [min] [max]`.
- El `[min]` es el número mínimo de caracteres de la contraseña que queremos crea y el `[max]` es el número máximo de caracteres en la contraseña. Por ejemplo, especificar los caracteres que queremos usar en las contraseñas, de modo que podemos especificar `abcdefg`, todas las minúsculas, y luego podemos escribir las mayúsculas; poner números y símbolos. La opción `-t` es muy útil si conocemos parte de la contraseña.
- ☑ Vamos a crear contraseñas de un mínimo de 4 y un máximo de 6 caracteres donde solamente vamos a incluir poner `123ab` y almacenarlo en una lista. Se puede comprobar con `cat` el listado generado.

```
[vicky@parrot]--[~]
$crunch 4 6 123ab -o lista2
Crunch will now generate the following amount of data: 131250 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 19375
crunch: 100% completed generating output
```

```
GNU nano 5.4 lista2 * I
1111
1112
1113
111a
111b
1121
1122
1123
112a
112b
1131
1132
1133
113a
113b
11a1
11a2
11a3
11aa
11ab
[ línea 1/19376 (0%), col 1/5 (20%), car 0/131250 (0%) ]
^H Ayuda ^O Leer fich. ^R Reemplazar ^V Pegar ^G Ir a línea ^Y Rehacer
^X Salir ^F Buscar ^K Cortar ^T Ejecutar ^Z Deshacer M-A Poner marca
```

☐ Crunch se usa principalmente en ataques masivos a páginas web de inicio de sesión.

☒ Para probar los patrones, vamos a crear contraseñas con cinco caracteres. Primero pondremos los caracteres 123ab (como antes), y agregamos opción -t para indicar que la contraseña comienza con a y termina con b, y queremos todas las combinaciones posibles de los caracteres entre a y b usando @. Luego, vamos a especificar el archivo de salida -o:

```
$ sudo nano lista2
[vicky@parrot]~$
[vicky@parrot]~$ crunch 5 5 123ab -t a@@@b -o lista3
Crunch will now generate the following amount of data: 750 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 125

crunch: 100% completed generating output
[vicky@parrot]~$
$
```

```
[vicky@parrot]~$
$ cat lista3
a111b
a112b
a113b
a11ab
a11bb
a121b
a122b
a123b
a12ab
a12bb
a131b
a132b
a133b
a13ab
a13bb
a1a1b
a1a2b
a1a3b
a1aab
a1abb
a1b1b
a1b2b
```

- Probar crear diccionarios con JOHN THE RIPPER y Hashcat. Vamos a probar a crear un diccionario con una de estas utilidades cogiendo como base algunas palabras y permutándolas.

- ☑ Por ejemplo, creamos un fichero de texto que contenga una palabra "hola" con `echo hola > diccionario.txt` para ver las combinaciones utilizamos John the Ripper sobre el fichero con el comando: `john -stdout -w:diccionario.txt --rules`

```
[vicky@parrot]-[~]  
└─$ john -stdout -w:diccionario.txt --rules  
Created directory: /home/vicky/.john  
Using default input encoding: UTF-8  
hola  
Hola  
holas  
hola1  
Hola1  
holahola  
aloh  
1hola  
HOLA  
hola2  
hola!  
hola3  
hola7  
hola9  
hola5  
hola4  
hola8  
hola6
```

- ☑ Probar lo mismo utilizando hascat: `hashcat --stdout -r /usr/share/hashcat/rules/leetspeak.rule diccionario.txt`
- Podemos mejorar el funcionamiento de JOHN THE RIPPER o HASCAT con el uso de nuevas reglas, tal es el caso por ejemplo de las reglas creadas por korelogic. Para manipular el directorio `/etc/john` nos descargamos el fichero `rules.txt` desde la página de korelogic <http://contest.korelogic.com/rules.html> con `wget` y luego, con el fichero descargado (según la versión disponible), se lo añadimos al final del fichero de configuración de john:

```
[x]-[vicky@parrot]-[~]  
└─$ hashcat --stdout -r /usr/share/hashcat/rules/leetspeak.rule diccionario.tx  
t  
█
```

```

[x]-[vicky@parrot]-[~]
└─$ hashcat --stdout -r /usr/share/hashcat/rules/leetspeak.rule diccionario.tx
t
se hol4
hol@
hola
hola
hola
hola
hola
hola
hola
hola
h0la
hola
hola
hola
hola
hola
h0l@
└─[vicky@parrot]-[~]
└─$

```

```

└─[vicky@parrot]-[~]
└─$ wget http://contest.korelogic.com/rules.html
--2023-08-09 09:53:19-- http://contest.korelogic.com/rules.html
Resolviendo contest.korelogic.com (contest.korelogic.com)... 205.134.174.174
Conectando con contest.korelogic.com (contest.korelogic.com)[205.134.174.174]:80... conectado.
Petición HTTP enviada, esperando respuesta... 302 Found
Localización: https://contest.korelogic.com/rules.html [siguiendo]
--2023-08-09 09:53:19-- https://contest.korelogic.com/rules.html
Conectando con contest.korelogic.com (contest.korelogic.com)[205.134.174.174]:443... conectado.
Petición HTTP enviada, esperando respuesta... 302 Found
Localización: https://contest-2010.korelogic.com/rules.html [siguiendo]
--2023-08-09 09:53:20-- https://contest-2010.korelogic.com/rules.html
Resolviendo contest-2010.korelogic.com (contest-2010.korelogic.com)... 205.134.174.174
Conectando con contest-2010.korelogic.com (contest-2010.korelogic.com)[205.134.174.174]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 24997 (24K) [text/html]
Grabando a: «rules.html»

rules.html          100%[=====] 24,41K  ---KB/s   en 0,1s
2023-08-09 09:53:21 (207 KB/s) - «rules.html» guardado [24997/24997]
└─[vicky@parrot]-[~]
└─$

```

```
← → 🏠 ↺ 🔒 https://contest-2010.korelogic.com/rules.txt 📄 ☆ ∞ 🔧 ☰
📁 Parrot OS 📁 Hack The Box 📁 OSINT Services 📁 Vuln DB 📁 Privacy and Security 📁 Learning Resources

[Incremental:RockYou-LanMan]
File = $JOHN/rockyou-lanman.chr
MinLen = 0
MaxLen = 7
CharCount = 69

[Incremental:rockyou]
File = $JOHN/rockyou.chr
MinLen = 0
MaxLen = 8
CharCount = 95

[Incremental:kore]
File = $JOHN/kore.chr
MinLen = 0
MaxLen = 8
CharCount = 95

#####
# KoreLogic Custom John the Ripper Rules:
#####

# Use this rule with 2EVERYTHING.dic or 3EVERYTHING.dic
[List.Rules:KoreLogicRulesPrependSeason]
A0"[Ss$][uU][mM][eE3][rR]"
A0"[Ww][iI][nN][tT+][eE3][rR]"
A0"[Ff][aA][lL][lL]"
A0"[Ss][pP][rR][iI][nN][gG]"
A0"[Aa][uU][tT][uU][mM][nN]"

# Use this rule with 2EVERYTHING.dic or 3EVERYTHING.dic
[List.Rules:KoreLogicRulesAppendSeason]
AZ"[Ss$][uU][mM][eE3][rR]"
AZ"[Ww][iI][nN][tT+][eE3][rR]"
AZ"[Ff][aA][lL][lL]"
AZ"[Ss][pP][rR][iI][nN][gG]"
AZ"[Aa][uU][tT][uU][mM][nN]"

[List.Rules:KoreLogicRulesPrependHello]
A0"[hH][eE][lL][lL][oO]"

[List.Rules:KoreLogicRulesPrependYears]
A0"20[0-1][0-9]"
A0"19[3-9][0-9]"
```

```
[vicky@parrot]~$
→ $sudo cat rules.txt >> /etc/john/john.conf
[vicky@parrot]~$
→ $
```

✓ Vamos ahora a probar a modificar JOHN THE RIPPER con nano / etc/john/john.conf para que durante el uso del diccionario se comporte como nosotros queramos de tal forma que por ejemplo añada al principio o al final de cada palabra alguna letra. Nos vamos a la configuración del programa y editamos su fichero de configuración para añadir una sección. Por ejemplo, la combinación TST con las líneas:



```
# Example: Override auto-tune for RAR format.
#rar_LWS = 128
#rar_GWS = 8192
[List.Rules:TSTRulesApendtst]
cAz"[tT][sS][tT]"
[List.Rules:TSTRulesPrependtst]
A0"[tT][sS][tT]"
[List.OpenCL:Drivers]
#Driver ; Description ; Recommendation
#AMD driver versions
938 , 2 ; 12.8 ;
1084, 4 ; 13.1 ;
1124 , 3 ; 13.4 ;
```

- Introducimos la regla al fichero anterior creado: `john -stdout -w:diccionario.txt --rules:TSTRulesPrependtst` y lo mismo con `Probamos lo mismo con Apendtst`.

```
[vicky@parrot]--[~]
$ john -stdout -w:diccionario.txt --rules:TSTRulesApendtst
Using default input encoding: UTF-8
Holatst
HolatsT
HolatSt
HolatST
HOLAst
HOLASt
HOLASt
HOLASt
HOLAst
HOLAst
HOLAst
HOLAst
HOLAst
HOLAst
HOLAst
8p 0:00:00:00 100,00% (2023-08-09 10:19) 114.2p/s HolaTST
[vicky@parrot]--[~]
$
```

```
[vicky@parrot]--[~]
$ john -stdout -w:diccionario.txt --rules:TSTRulesPrependtst
Using default input encoding: UTF-8
tT]sthola
tT]sThola
tT]Sthola
tT]SThola
4p 0:00:00:00 100,00% (2023-08-09 10:19) 66.66p/s tT]SThola
[vicky@parrot]--[~]
$
```

- ❑ En Kali tenemos una serie de diccionarios ya guardados que podemos ver en el siguiente directorio con el comando: `ls -lart /usr/share/wordlists/`
- ❑ Para ver el contenido de un fichero comprimido .gz se usa `zcat`: `zcat /usr/share/wordlists/rockyou.txt.gz`

```
[vicky@parrot]~  
$ ls -lart /usr/share/wordlists/  
total 52140  
-rw-r--r-- 1 root root 53357329 oct 13 2021 rockyou.txt.gz  
lrwxrwxrwx 1 root root 25 abr 28 12:03 wfuzz -> /usr/share/wfuzz/wordlist  
lrwxrwxrwx 1 root root 41 abr 28 12:03 nmap.lst -> /usr/share/nmap/nmaplib/data/passwords.lst  
lrwxrwxrwx 1 root root 46 abr 28 12:03 metasploit -> /usr/share/metasploit-framework/data/wordlists  
lrwxrwxrwx 1 root root 45 abr 28 12:03 fern-wifi -> /usr/share/fern-wifi-cracker/extras/wordlists  
lrwxrwxrwx 1 root root 41 abr 28 12:03 fasttrack.txt -> /usr/share/set/src/fasttrack/wordlist.txt  
lrwxrwxrwx 1 root root 35 abr 28 12:03 dnsmap.txt -> /usr/share/dnsmap/wordlist_TLAs.txt  
lrwxrwxrwx 1 root root 30 abr 28 12:03 dirbuster -> /usr/share/dirbuster/wordlists  
lrwxrwxrwx 1 root root 25 abr 28 12:03 dirb -> /usr/share/dirb/wordlists  
drwxr-xr-x 1 root root 164 abr 28 12:03 .  
drwxr-xr-x 1 root root 7912 ago 3 07:56 ..  
[vicky@parrot]~  
$
```

```
[x]--[vicky@parrot]--[usr/share/wordlists]  
$ zcat rockyou.txt.gz
```