Angel Rodriguez
CS 486 - Cryptography
Project 1 part 2

~~Password Cracking on Kali Linux part 2~~
p4s5w0rD kR4<kin 0n k4L! L!NuX p4rT 2

Index

1 - Terminology

> **Depth**: Refers to the number of clicks you need to reach a specific page from the homepage using the shortest path. A page directly linked to the homepage is at depth 2 (the homepage itself is always 1!).

> **theScore**: theScore creates mobile-first sports experiences, connecting fans to what they love through an addictive combination of real-time news, scores, fantasy information and alerts while creating and curating content that is mobile optimized, comprehensive, customizable and seamlessly shareable. It focuses on mobile experience but also has a website with all the information also given in the application.

2 - Idea

We are going to be looking at more tools and methods to polish our password cracking techniques, given that we already know how to use the material given in the first part of this password cracking series. **We will focus on generating our own custom word lists that we can input on John The Ripper / Johnny, instead of using the default ones on those toolkits.**

3 - Requirements

Very much like the first part, you will need Kali Linux OS and a strong understanding on how to use the command line.

4 - Objectives

The main objective for part 2 of the series is to further reinforce the need for complex, significant and thoughtful passwords knowing that attackers can be way more elaborate on how to hack passwords given the tools and methods that will be explained in this document.

5 - Tools and Materials Used

For the second part of the project, we are going to need two more toolkits already provided by Kali Linux, CeWL and Crunch. We are also going to be using the website www.thescore.com as an example. I will be again be running Kali Linux as a virtual image of VirtualBox software on my MacBook Pro as the host.

6 - Warnings and Disclaimer

Proceed at your own risk, Kali Linux is a professional environment with industry-level tools. Do not attempt the following procedure for malicious ends, this is purely educational penetration testing. I am not responsible for any damages on your side.

7 - CeWL

CeWL (Custom Word List generator, pronounced “cool” and written by author “Robin Wood”) is a ruby app which spiders a given url to a specified depth, optionally following external links, and returns a list of words which can then be used for password crackers such as **John the Ripper.** CeWL also has an associated command line app, FAB (Files Already Bagged) which uses the same meta data extraction techniques to create author/creator lists from already downloaded. A manual for CeWL can be seen by typing the command `man cewl` in the terminal.

8 - Procedure for CeWL

Let’s pretend we are trying to hack the password of one of our co-worker called Carlos. We have already tried the methods and used the tools explained in part 1 but we were not successful because the user was knowledgeable on the topic and had a stronger password that the ones provided as examples. Say that we decided to further investigate Carlos and have found out that he is a passionate follower of sports, and that he goes on the website www.thescore.com more often than not. We can use CeWL to crawl theScore’s website and create a custom wordlist to then use it on the tools explained on part 1 to have a better chance of success at hacking our victim.

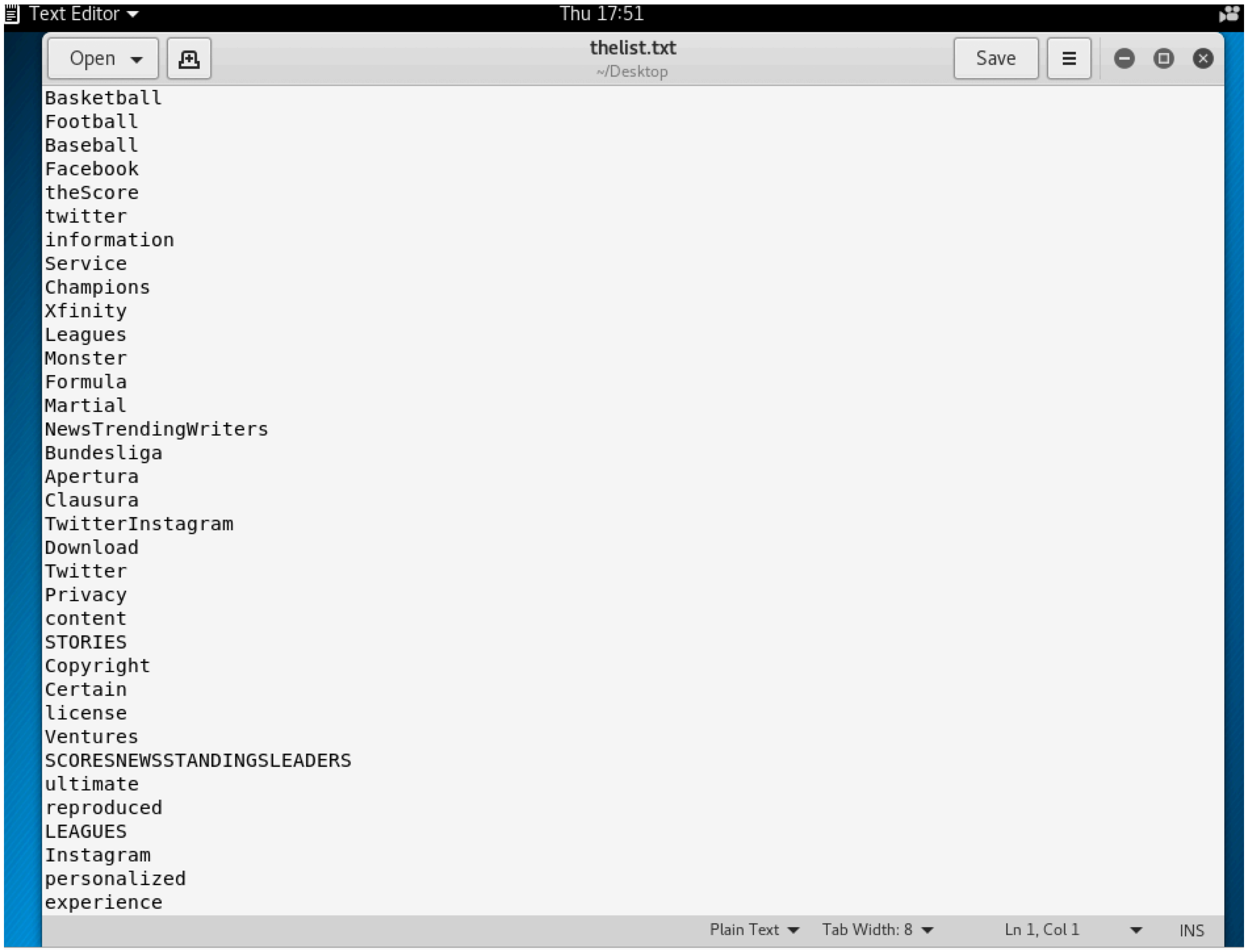
>Create a custom word list:
We can do so typing the following command in the terminal (**make sure you are in the Desktop directory, you can do so by typing the command `cd Desktop`**):

```
cewl -w thelist.txt -d 1 -m 7 https://www.thescore.com
```

By typing this command, we are asking CeWL to crawl theScore’s website with a depth of 1 and a minimum word length of 7 characters. It will then output the word list to a file called “thelist.txt” in the current directory (should be Desktop in this case). **Note**: It will take some time for this to be done, be patient. You will know when the process is ready when the terminal gives you access to typing commands again.

9 - Results for CeWL

Once you open the file "thelist.txt" located in your Desktop, it should show something like this:



```
Basketball
Football
Baseball
Facebook
theScore
twitter
information
Service
Champions
Xfinity
Leagues
Monster
Formula
Martial
NewsTrendingWriters
Bundesliga
Apertura
Clausura
TwitterInstagram
Download
Twitter
Privacy
content
STORIES
Copyright
Certain
license
Ventures
SCORESNEWSSTANDINGSLEADERS
ultimate
reproduced
LEAGUES
Instagram
personalized
experience
```

We have now successfully created a word list filled with words related to sports crawled from theScore's website that we can use on polishing our attack.

But wait, there is more. Enter Crunch.

10 - Crunch

Crunch is a wordlist generator developed by author "bofh28" where you can specify a standard character set or a character set you specify. Crunch can generate all possible combinations and permutations. The basic syntax for crunch is the following:

```
crunch <min> <max> <characterset> -t <pattern> -o <output filename>
```

- <min> = The minimum password length.
- <max> = The maximum password length.
- <characterset> = The character set to be used in generating the passwords.
- -t <pattern> = The specified pattern of the generated passwords. For instance, if you knew that the target's birthday was 0728 (July 28th) and you suspected they used their birthday in their password (people often do), you could generate a password list that ended with 0728 by giving crunch the pattern @@@@@@@0728. This word generate passwords up to 11 characters (7 variable and 4 fixed) long that all ended with 0728.
- -o <outputfile> = This is the file you want your wordlist written to.

A manual for crunch with more commands can be seen by typing the command `man crunch` in the terminal.

11 - Procedure for Crunch

Let's say that we know Carlos' birthday, December 17th for example. His password might contain the numbers "1217", referring to his birthday, as explained in the previous section of the document.

We can use the powerful -t pattern command to specify that we would like a list containing the characters "1217" at the end. For demonstration purposes, let's say that the password is 5 characters long, so we would type the command:

```
crunch 5 5 -t @1217
```

This would give us a list (in the terminal, use "-o" command if you want them in a file) of 26 items ranging from "a@1217" to "z@1217". Of course we can be more elaborate and use different character sets with the characterset command. If we wanted our list to be of 2 characters followed by the number "1217", we would type: `crunch 6 6 -t @@1217`

If we want Crunch to use the character set of "Football" only, we would type:

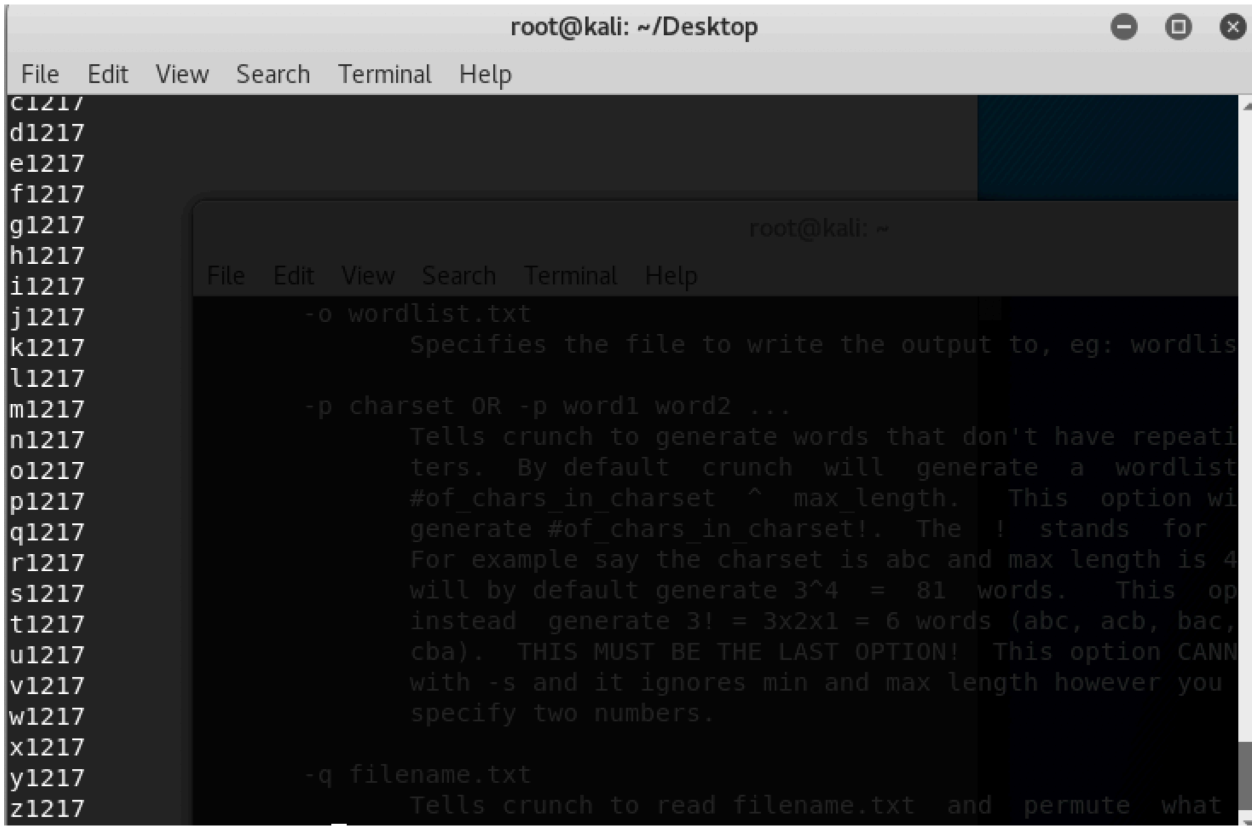
```
crunch 12 12 Football -t @@@@@@@@1217
```

Using this command, we would get a massive list (1679616 lines) of permutations of the word “Football” followed by the number “1217”

The possibilities are endless here. We can generate all of these lists and combine them together for a more effective one. **Note**: again, you have to be patient here, generating some of these lists depending on the criteria can take even several hours.

12 - Results for Crunch

Output for: crunch 5 5 -t @1217


root@kali: ~/Desktop
File Edit View Search Terminal Help

```
c1217
d1217
e1217
f1217
g1217
h1217
i1217
j1217
k1217
l1217
m1217
n1217
o1217
p1217
q1217
r1217
s1217
t1217
u1217
v1217
w1217
x1217
y1217
z1217
```

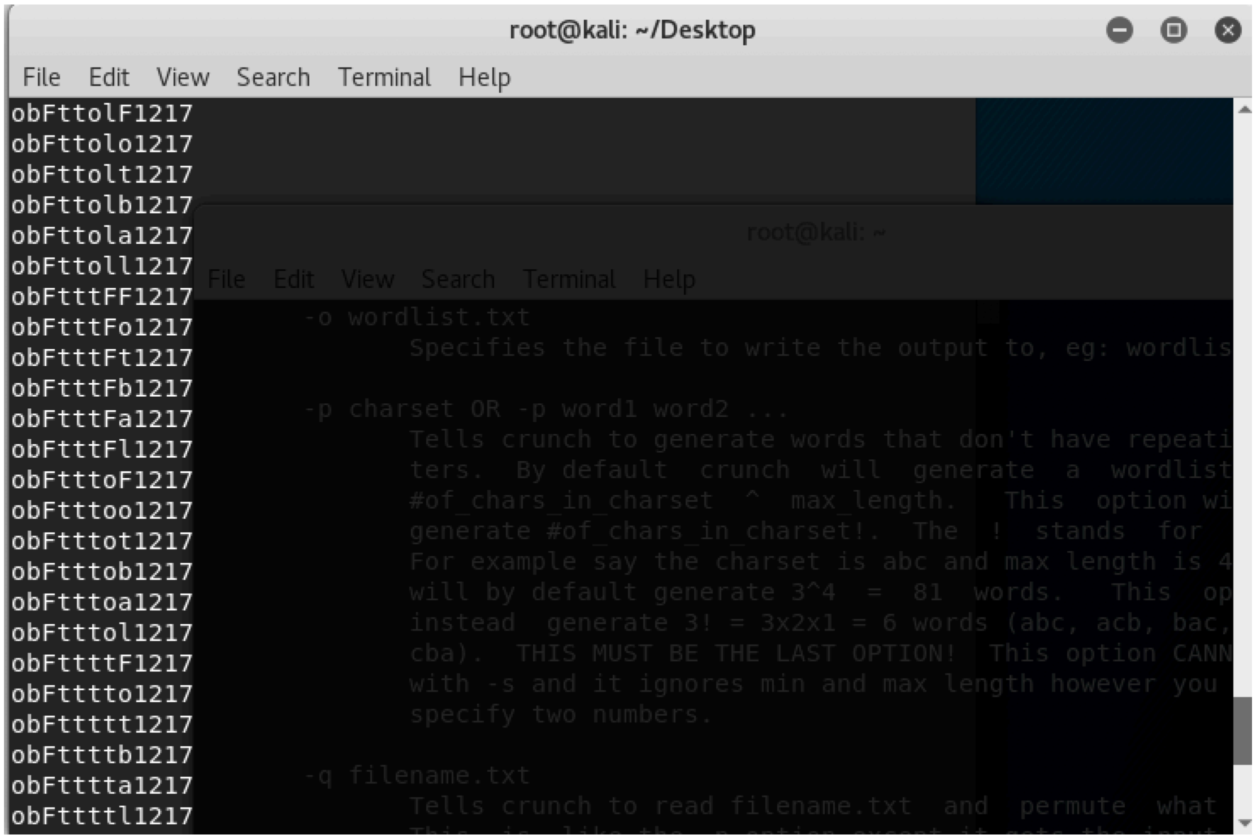
root@kali: ~
File Edit View Search Terminal Help

```
       -o wordlist.txt
              Specifies the file to write the output to, eg: wordlis

       -p charset OR -p word1 word2 ...
              Tells crunch to generate words that don't have repeati
              ters.  By default  crunch  will  generate  a  wordlist
              #of_chars_in_charset  ^  max_length.   This  option wi
              generate #of_chars_in_charset!.  The  !  stands  for
              For example say the charset is abc and max length is 4
              will by default generate 3^4  =  81  words.   This  op
              instead  generate 3! = 3x2x1 = 6 words (abc, acb, bac,
              cba).  THIS MUST BE THE LAST OPTION!  This option CANN
              with -s and it ignores min and max length however you
              specify two numbers.

       -q filename.txt
              Tells crunch to read filename.txt  and  permute  what
```

Output for: crunch 12 12 Football -t @@@@@@@@1217


root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali: ~
File Edit View Search Terminal Help

```
obFttolF1217
obFttolo1217
obFttolt1217
obFttolb1217
obFttola1217
obFttoll1217
obFtttFF1217
obFtttFo1217
obFtttFt1217
obFtttFb1217
obFtttFa1217
obFtttFl1217
obFtttoF1217
obFtttoo1217
obFtttot1217
obFtttob1217
obFtttoa1217
obFtttol1217
obFttttF1217
obFtttto1217
obFttttt1217
obFttttb1217
obFtttta1217
obFttttl1217
```

```
       -o wordlist.txt
              Specifies the file to write the output to, eg: wordlis

       -p charset OR -p word1 word2 ...
              Tells crunch to generate words that don't have repeati
              ters.  By default  crunch  will  generate  a  wordlist
              #of_chars_in_charset  ^  max_length.   This  option wi
              generate #of_chars_in_charset!.  The  !  stands  for
              For example say the charset is abc and max length is 4
              will by default generate 3^4  =  81  words.   This  op
              instead  generate 3! = 3x2x1 = 6 words (abc, acb, bac,
              cba).  THIS MUST BE THE LAST OPTION!  This option CANN
              with -s and it ignores min and max length however you
              specify two numbers.

       -q filename.txt
              Tells crunch to read filename.txt  and  permute  what
```

13 - Conclusion

We can conclude that passwords are something that should not be taken in a slight manner. Attackers could investigate and gather more information on a person and use that as valuable data to carry a more effective attack on the victim. A very valuable lesson here is also to keep in mind that our passwords should not be related to something that someone could find out about us. For example, if I love chess and I am very outspoken about it, an attacker could use that data to carry out a strong attack on me, and the attacker might discover that my password is something like “Rook0323” which would be the name of a piece of chess followed by my hypothetical birthday. Keep in mind the attacker could also generate several lists from these two toolkits discussed in this document, CeWL and Crunch, and mix and combine them in different ways using something like python scripts or simple terminal commands (as seen in part 1). Attackers can get very creative and generate a very strong list of possibilities that could be very effective in an attack against a victim compared to John The Ripper / Johnny default password lists.

It is preferable to have seemingly random passwords, following the notions of part 1, such as “Vb4.M-165$Qfa0rTt!” (this is randomness at an exaggerated level but I hope you get the idea) that is not related to anything specially something that can be found out about you.

I have also created a video tutorial explaining the procedures for the use of both tools discussed in this document, the link is the following: https://youtu.be/gThGCgfbUd0

14 - References

Kali Linux Official Documentation:
> https://docs.kali.org/introduction/what-is-kali-linux

Crunch:
> https://tools.kali.org/password-attacks/crunch
> https://null-byte.wonderhowto.com/how-to/hack-like-pro-crack-passwords-part-4-creating-custom-wordlist-with-crunch-0156817/
> http://manpages.ubuntu.com/manpages/bionic/man1/crunch.1.html

CeWL:
> https://tools.kali.org/password-attacks/cewl
> https://null-byte.wonderhowto.com/how-to/hack-like-pro-crack-passwords-part-5-creating-custom-wordlist-with-cewl-0158855/
> https://www.oncrawl.com/oncrawl-seo-thoughts/page-depth-how-it-affects-your-seo-performance/

TheScore:
https://www.thescore.com