Sharyl Lynn Riley

# Hacking Journal

July 28, 2020

**Target(s):**

Geheim.zip file.

**Mission Objective:**

Using any tools, you find helpful, within Kali or elsewhere, disclose the content of the given ZIP archive without having the password readily available. The solution must contain pictures of your work with appropriate annotation (at least one with each tool used) with the content of the zip file and a short summary how you solved the challenge.

All documentation must be provided in the Hacking Journal (found in student resources section) and uploaded to this assignment for grading. The methods are your technical approach and process for solving the challenge. ***Use at least three methods with three different tools.*** You are the professional here. You have been 'hired' to investigate and break this file as necessary. Document what you do and what you find accordingly.

**Tools Used:** fcrackzip; zippasswordcrackerprov1.4f; John the Ripper

Method 1: Dictionary Attack

```
root@kali:~# apt-get install fcrackzip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  fcrackzip
0 upgraded, 1 newly installed, 0 to remove and 213 not upgraded.
Need to get 28.2 kB of archives.
After this operation, 81.9 kB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 fcrackzip amd64 1.0-9 [2
8.2 kB]
Fetched 28.2 kB in 1s (43.4 kB/s)
Selecting previously unselected package fcrackzip.
(Reading database ... 352552 files and directories currently installed.)
Preparing to unpack .../fcrackzip_1.0-9_amd64.deb ...
Unpacking fcrackzip (1.0-9) ...
Setting up fcrackzip (1.0-9) ...
Processing triggers for man-db (2.8.6.1-1) ...
```

```
root@kali:~# fcrackzip --help

fcrackzip version 1.0, a fast/free zip password cracker
written by Marc Lehmann <pcg@goof.com> You can find more info on
http://www.goof.com/pcg/marc/

USAGE: fcrackzip
          [-b|--brute-force]            use brute force algorithm
          [-D|--dictionary]             use a dictionary
          [-B|--benchmark]              execute a small benchmark
          [-c|--charset characterset]   use characters from charset
          [-h|--help]                   show this message
          [--version]                   show the version of this program
          [-V|--validate]               sanity-check the algortihm
          [-v|--verbose]                be more verbose
          [-p|--init-password string]   use string as initial password/file
          [-l|--length min-max]         check password with length min to max
          [-u|--use-unzip]              use unzip to weed out wrong passwords
          [-m|--method num]             use method number "num" (see below)
          [-2|--modulo r/m]             only calculcate 1/m of the password
          file...                     the zipfiles to crack

methods compiled in (* = default):
```

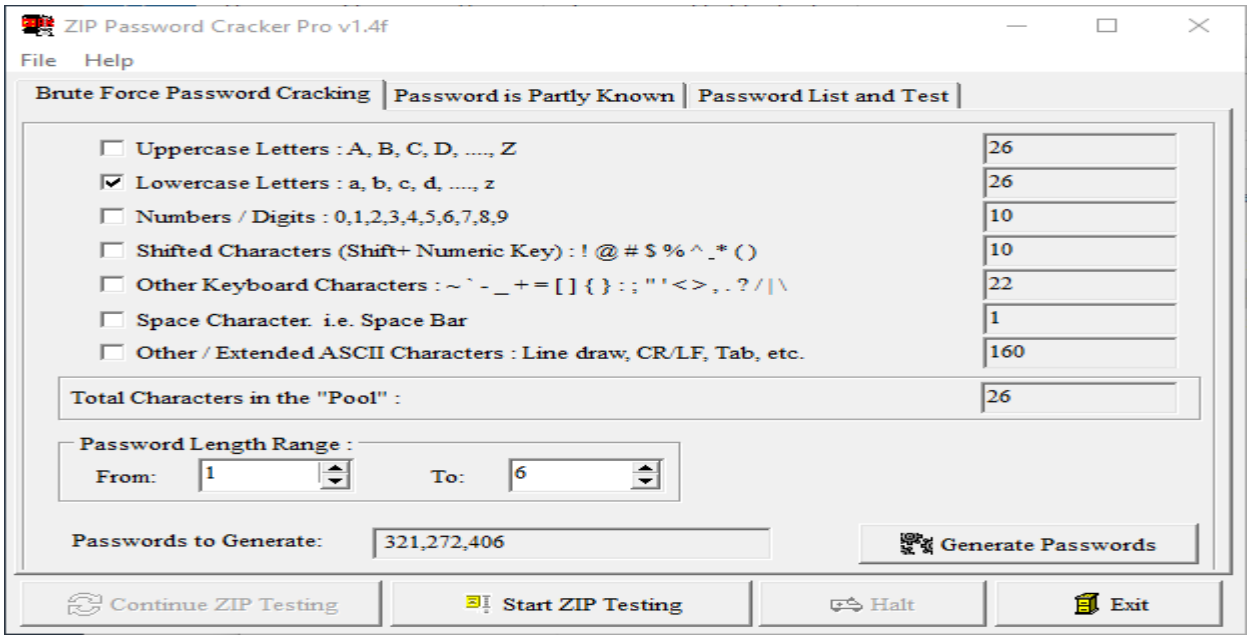
```
root@kali:~/Downloads# crunch 5 5 -f /usr/share/crunch/charset.lst lalpha -o passlist.txt
Crunch will now generate the following amount of data: 71288256 bytes
67 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 11881376

crunch: 100% completed generating output
```

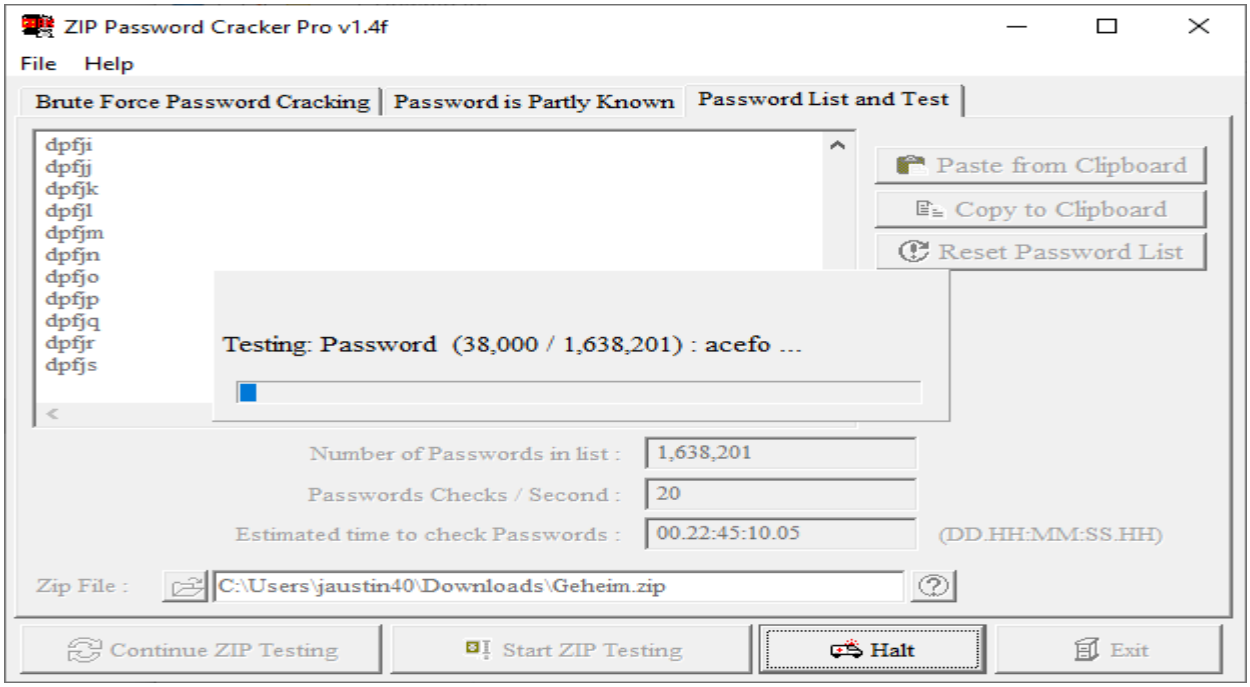
```
root@kali:~/Downloads# fcrackzip -D -p passlist.txt Geheim2.zip
possible pw found: aaabr ()
possible pw found: aaazh ()
possible pw found: aabav ()
possible pw found: aabip ()
possible pw found: aabkm ()
possible pw found: aabsl ()
possible pw found: aabwx ()
possible pw found: aacgh ()
possible pw found: aacph ()
possible pw found: aacrp ()
```

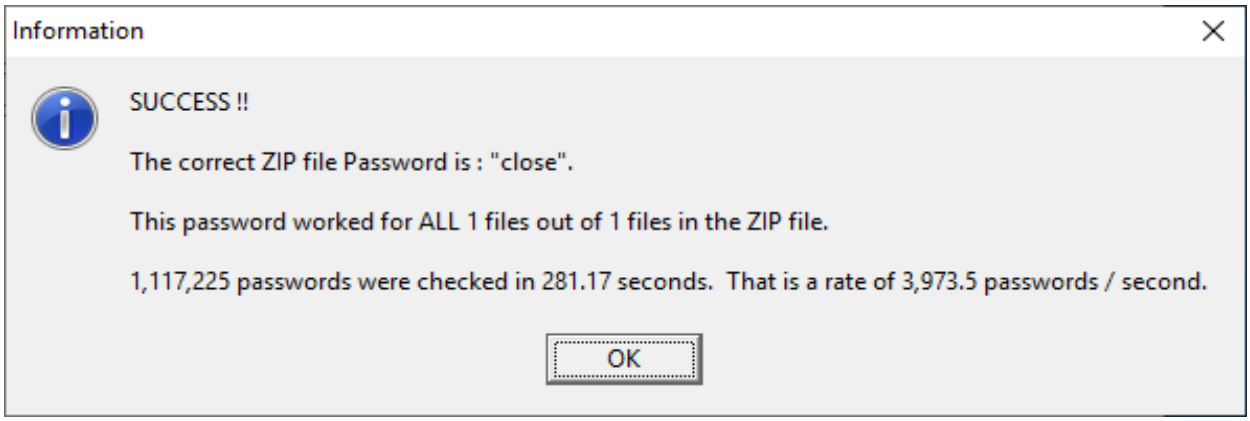
```
root@kali:~/Downloads# fcrackzip -v -u -D -p passlist.txt Geheim2.zip
found file 'Geheim.txt', (size cp/uc     57/    45, flags 1, chk cdce)
checking pw cexhn

PASSWORD FOUND!!!!: pw == close
```

```
root@kali:~/Downloads# cat passlist.txt
aaaaa
aaaab
aaaac
aaaad
aaaae
aaaaf
aaaag
aaaah
aaaai
aaaaj
aaaak
aaaal
aaaam
aaaan
aaaao
aaaap
aaaaq
aaaar
aaaas
aaaat
aaaau
aaaav
aaaaw
```

Method 2 Brute Force:


ZIP Password Cracker Pro v1.4f
File
Help
Brute Force Password Cracking
Password is Partly Known
Password List and Test
Uppercase Letters : A, B, C, D, ...., Z
26
Lowercase Letters : a, b, c, d, ...., z
26
Numbers / Digits : 0,1,2,3,4,5,6,7,8,9
10
Shifted Characters (Shift+ Numeric Key) : ! @ # $ % ^ _* ( )
10
Other Keyboard Characters : ~ ` - _ + = [ ] { } : ; " ' < > , . ? / | \
22
Space Character. i.e. Space Bar
1
Other / Extended ASCII Characters : Line draw, CR/LF, Tab, etc.
160
Total Characters in the "Pool" :
26
Password Length Range :
From:
1
To:
6
Passwords to Generate:
321,272,406
Generate Passwords
Continue ZIP Testing
Start ZIP Testing
Halt
Exit


ZIP Password Cracker Pro v1.4f
File
Help
Brute Force Password Cracking
Password is Partly Known
Password List and Test
dpfji
dpfjj
dpfjk
dpfjl
dpfjm
dpfjn
dpfjo
dpfjp
dpfjq
dpfjr
dpfjs
Paste from Clipboard
Copy to Clipboard
Reset Password List
Testing: Password (38,000 / 1,638,201) : acefo ...
Number of Passwords in list :
1,638,201
Passwords Checks / Second :
20
Estimated time to check Passwords :
00.22:45:10.05
(DD.HH:MM:SS.HH)
Zip File :
C:\Users\jaustin40\Downloads\Geheim.zip
Continue ZIP Testing
Start ZIP Testing
Halt
Exit




Method 3: Pass the Hash

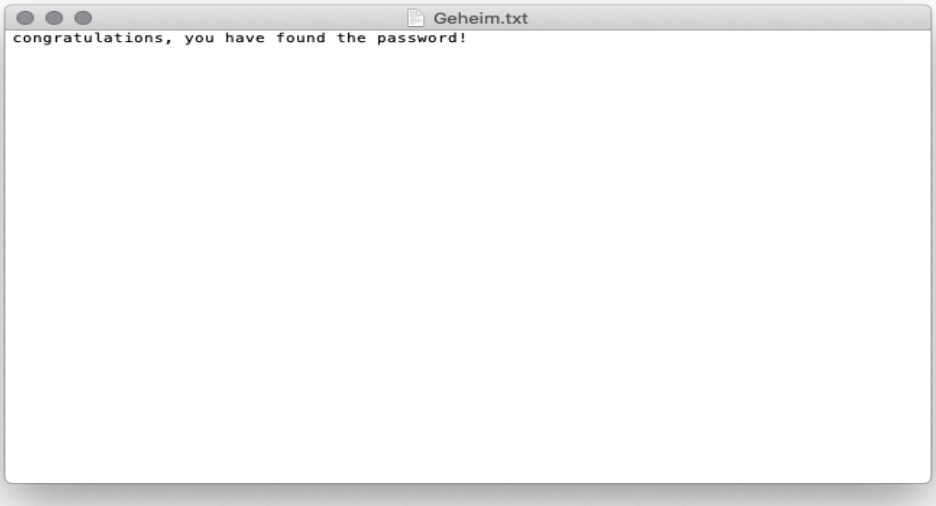
```
root@kali:~/Downloads# zip2john
Created directory: /root/.john
Usage: zip2john [options] [zip file(s)]
Options for 'old' PKZIP encrypted files only:
 -a <filename>   This is a 'known' ASCII file. This can be faster, IF all
    files are larger, and you KNOW that at least one of them starts out as
    'pure' ASCII data.
 -o <filename>   Only use this file from the .zip file.
 -c This will create a 'checksum only' hash.  If there are many encrypted
    files in the .zip file, then this may be an option, and there will be
    enough data that false positives will not be seen.  If the .zip is 2
    byte checksums, and there are 3 or more of them, then we have 48 bits
    knowledge, which 'may' be enough to crack the password, without having
    to force the user to have the .zip file present.
 -m Use "file magic" as known-plain if applicable. This can be faster but
    not 100% safe in all situations.
 -2 Force 2 byte checksum computation.

NOTE: By default it is assumed that all files in each archive have the same
password. If that's not the case, the produced hash may be uncrackable.
To avoid this, use -o option to pick a file at a time.
```

```
root@kali:~/Downloads# zip2john Geheim.zip > zip-hash.txt
ver 1.0 Geheim.zip/Geheim.txt PKZIP Encr: cmplen=57, decmplen=45, crc=CDCEEB7B
```

```
root@kali:~/Downloads# cat zip-hash.txt
Geheim.zip/Geheim.txt:$pkzip2$1*1*2*0*39*2d*cdceeb7b*0*28*0*39*cdce*4520*b4e2f48
6b7ac79336589b0fa5841db48d2d737fecb303e1d54e6f6a69844a14d3996d8ec8698688dbcb32e1
d45378e61b2a9be7d1c9f92ce8d*$/pkzip2$:Geheim.txt:Geheim.zip::Geheim.zip
```

```
root@kali:~/Downloads# john zip-hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for p
erformance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for p
erformance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
close        (Geheim.zip/Geheim.txt)
1g 0:00:00:01 DONE 3/3 (2019-12-05 20:51) 0.5617g/s 714474p/s 714474c/s 714474C/
s cezhp...closh
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

**Overall Summary of challenge procedures:** The overall level of difficulty of the tools were somewhat easy to learn but you do need to know the commands within the application tool. I personally liked the GUI brute force the most. I do like to use the terminal, but I liked the look of the GUI better. The outcome of the attacks is the password CLOSE – see screenshot of the contents of the text file:




Interestedly, these tools are taught on YouTube and websites, but you need to have some background foundational ethical hacking experience to be able to understand some of tool's functions, as they require writing scripts to be effective. The pros of these tools: they come pre-installed, and there are thousands more to download in their repositories should you have needed to download one of them. The menus are beautifully organized. The Widgets are very handy and easy to read. Whereas the cons

of the tools are that some must downloaded from repositories. I used Kali on a Virtual Machine that was easy to install and keep updated.