

Pydictor工具介绍

1、工具介绍

这个小工具是用来生成字典的，比如可以生成8位纯数字的字典，用来跑一些密码的时候会用到。（这个的可定制性很高，所以使用就很方便），目前最方便的生成密码的工具之一。

比如我只记得密码后面六位是123456，前面两位是什么忘记了，就可以使用这个工具很方便的生成一个**密码字典**，来进行找回密码的工作。生成满足上面条件的密码字典文件就只需要下面的一行命令：

```
python pydictor.py -base Lc --len 2 2 --tail 123456 -base Lc
```

指定所有大写和小写字母，**len**表示长度，**tail**表示以123456结尾。

pydictor优点

- 你可以用pydictor生成普通**爆破字典**、基于网站内容的自定义字典、**社会工程学字典**等等一系列高级字典；
- 你可以使用pydictor的内置工具，对字典进行安全删除、合并、去重、合并并去重、高频词筛选；
- 除此之外，你还可以输入自己的字典，然后使用**handler**工具，对字典进行各种筛选，编码或加密操作；
- 可定制性强，你可以通过修改多个配置文件、加入自己的字典、选用**leet mode**模式、长度选择、各类字符数量筛选、各类字符种类数筛选、正则表达式筛选，甚至可通过修改 `/lib/fun/encode.py`文件，自定义加密方法等高级操作；按照API编写标准，在**plugins/**文件夹下添加自己的插件脚本，在**tools/**目录下添加自己的工具脚本等。生成独一无二的高度定制、高效率和复杂字典，生成密码字典的好坏和你的自定义规则、能不能熟练使用pydictor有很大关系；
- 强大灵活的配置解析功能；
- 兼容性，不管你是使用的python 2.7版本还是python 3.4 以上版本，pydictor都可以在Windows、Linux 或者是Mac上运行；

2、工具安装

在github上搜索，直接下载即可

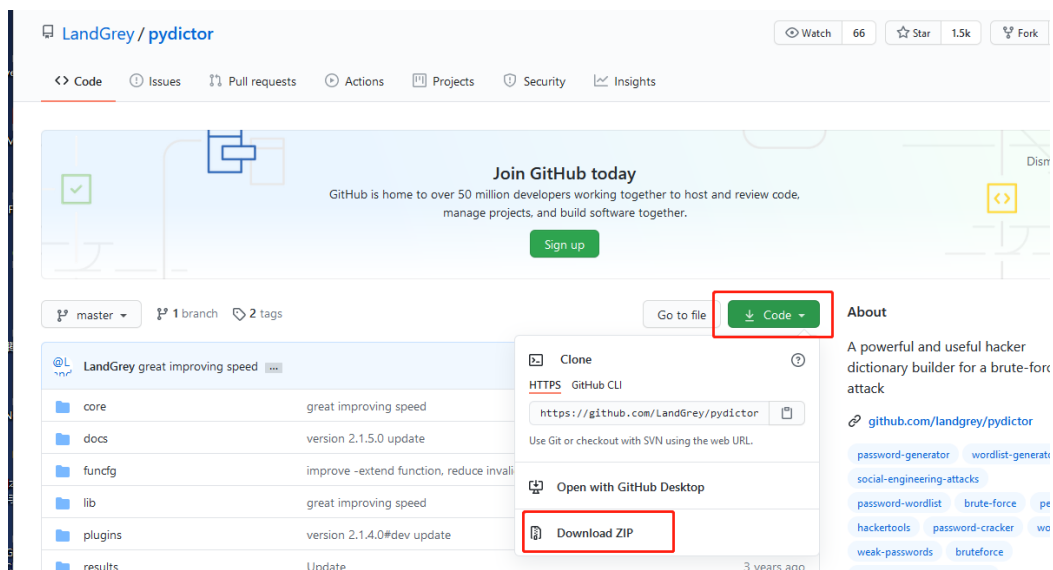
pydictor

build Python 2.7&3.4 release License

README.md 中文版

pydictor — A powerful and useful hacker dictionary builder for a brute-force attack

Email: LandGrey@qq.com



另外通过git工具也可以实现下载

```
git clone --depth=1 --branch=master
https://www.github.com/landgrey/pydictor.git
cd pydictor/
chmod 755 pydictor.py
python pydictor.py
```

下载后修改pydictor目录下的文件，设置权限为755，增加文件可执行操作。

3、工具使用

使用工具，先看帮助文档，

```
root@kali:~/home/kai/pydictor# python pydictor.py

usage:
pydictor.py [options]
  -base [type]
  -char [custom_char]
  -chunk [chunk1] [chunk2] ...
  -extend [string_or_file]
  -plug [birthday,pid6,ftp,pid8,pid4,scratch]
  --conf [expression_or_file]
  --sedb
  -o, --output [directory]
  -tool [shredder,uniqbiner,handler,uniqifer,comparer,hybrider,counter,combiner]
  --len [minlen] [maxlen]
  --head [prefix_string]
  --tail [suffix_string]
  --encode [none,shal,b64,url,execjs,des,rsa,b32,b16,test,sha256,sha512,hmac,md516,md5]
  --occur [letter] [digital] [special]
  --types [letter] [digital] [special]
  --repeat [letter] [digital] [special]
  --regex [regex]
  --level [code]
  --leet [code]
  --dmy

另外通过git 2.1.4.1#dev下载
git clone --depth=1 --branch=master
https://www.github.com/landgrey/pydictor.git
cd pydictor/
chmod 755 pydictor.py
python pydictor.py

下载后修改pydictor目录下的文件，设置权限为755，增加可执行操作。
```

模块和选项还是比较多，对初学者或者英文不好者可能不是特别友好，接下来针对功能划分为三部分：核心功能字典、插件型字典、内置工具三部分来讲解该工具，希望通过讲解能够把该工具学透彻。

```
基础  ← -base [type]
自定义 ← -char [custom_char]
排列组合 ← -chunk [chunk1] [chunk2] ...
扩展 ← -extend [string_or_file]
插件 ← -plug [birthday,pid6,ftp,pid8,pid4,scratch]
社会工程 ← --conf [expression_or_file]
工具 ← --sedb
        -o, --output [directory]
        -tool [shredder,uniqbiner,handler,uniqifer,comparer,hy
        --len [minlen] [maxlen]
        --head [prefix_string]
        --tail [suffix_string]
        --encode [none,shal,b64,url,execjs,des,rsa,b32,b16,test,s
        --occur [letter] [digital] [special]
        --types [letter] [digital] [special]
        --repeat [letter] [digital] [special]
        --regex [regex]
        --level [code]
        --leet [code]
        --dmy
```

核心模块

归属	类别	标识符	描述	支持功能代号
core	base	C1	基础字典	F1 F2 F3 F4
core	char	C2	自定义字符集字典	F1 F2 F3 F4
core	chunk	C3	排列组合字典	ALL
core	conf	C4	配置语法生成字典	ALL
core	extend	C5	规则扩展字典	ALL
core	sedb	C6	社会工程学字典	ALL

（1）核心功能字典

常用的选项

pydictor字典操作功能及说明对照表

功能	功能代号	说明
len	F1	定义长度范围
head	F2	添加前缀
tail	F3	添加后缀
encode	F4	编码或自定义加密方法
occur	F5	字母、数字、特殊字符出现次数范围筛选
types	F6	字母、数字、特殊字符种类数范围筛选
regex	F7	正则筛选
level	F8	字典级别筛选
leet	F9	1337 模式

编码函数支持编码和加密方式

pydictor支持的编码或加密方式

方式	描述
none	默认方式, 不进行任何编码
b16	base16 编码
b32	base32 编码
b64	base64 编码
des	des 算法, 需要根据情况修改代码
execjs	执行本地或远程js函数, 需要根据情况修改代码
hmac	hmac 算法, 需要根据情况修改代码
md5	md5 算法输出32位
md516	md5 算法输出16位
rsa	rsa 算法 需要根据情况修改代码
sha1	sha-1 算法
sha256	sha-256 算法
sha512	sha-512 算法
url	url 编码
test	一个自定义编码方法的示例

occur 功能

用法：--occur [字母出现次数的范围] [数字出现次数的范围] [特殊字符出现次数的范围]

示例：--occur ">=4" "<6" "=="

types 功能

用法：--types [字母种类的范围] [数字种类的范围] [特殊字符种类的范围]

示例：--types "<=8" "<=4" "=="

regex 功能

用法：--regex [正则表达式]

示例：--types "^z.*?g\$"

level 功能

用法：--level [level]

1 示例：--level 4 /funcfg/extend.conf配置文件中level大于等于4的项目会被启用

1) 基础字典

```
python pydictor.py -base L --len 2 3 --encode b64
python pydictor.py -base dLc --len 1 3 -o /awesome/pwd
python pydictor.py -base d --len 4 4 --head Pa5sw0rd
```

基本用法：【-base】

-base 用来生成字典所需的数字、小写字母、大写字母，可以排列组合，顺序需要注意。

-base Type	Choose from (d, L, c, dL, dc, Lc, dLc)	
d	digital	[0 - 9]
L	lowercase letters	[a - z]
c	capital letters	[A - Z]
dL	Mix d and L	[0-9 a-z]
dc	Mix d and c	[0-9 A-Z]
Lc	Mix L and c	[a-z A-Z]
dLc	Mix d, L and dL	[0-9 a-z A-Z]

--len 表示生成字典的长度，可以采取默认（min=0,max=4），也可以自己指定，如果min=max说明字典长度恒定；

--head 在生成的字典前加固定字符串；

--tail 在生成的字典末尾加固定字符串

--encode 对生成的字典进行编码，支持常见的编码类型


```
root@kali:/home/kai/pydictor# python pydictor.py -base d --len 4 4 --head Pa5sw0r
2.1.4.1#dev
[+] A total of :10000 lines
[+] Store in :/home/kai/pydictor/results/base_002023.txt
[+] Cost :0.0626 seconds
root@kali:/home/kai/pydictor# more /home/kai/pydictor/results/base_002023.txt
Pa5sw0r0000
Pa5sw0r0001
Pa5sw0r0002
Pa5sw0r0003
Pa5sw0r0004
Pa5sw0r0005
```

2) 自定义字符集字典

自定义用法【-char】

自定义字符集生成字典，例如需要在指定的字符串“asdf123._@”中，随机排列组合生成长度为2-3的密码，且密码尾部包含固定字符串“@site.com”；

```
python pydictor.py -char "asdf123._@" --len 2 3 --tail @site.com
```

```
root@kali:/home/kai/pydictor# python pydictor.py -char "asdf123._@" --len 2 3 --tail @site.com
2.1.4.1#dev
[+] A total of :1452 lines
[+] Store in :/home/kai/pydictor/results/char_002343.txt
[+] Cost :0.0331 seconds
root@kali:/home/kai/pydictor# more /home/kai/pydictor/results/char_002343.txt
aa@site.com
as@site.com
ad@site.com
af@site.com
a1@site.com
a2@site.com
a3@site.com
```

3) 排列组合字典

排列组合用法【-chunk】

自定义生成字典，要求在“abc” “123” @ . _ “kitty” “root” “0809”这些字符串中按照排列组合生成字典，且字典的头部为字符a，尾部字符为@pass，且要求字典为md5加密；

```
python pydictor.py -chunk "abc" "123" @ . _ "kitty" "root" "0809"
--head a --tail @pass --encode md5
```

```

root@kali:/home/kai/pydictor# python pydictor.py -chunk "abc" "123" @ . _ "kitty" "root" "0809" --head a --tail
@pass --encode md5
pydictor 2.1.4.1#dev
[+] A total of :40320 lines
[+] Store in :/home/kai/pydictor/results/chunk_002728.txt
[+] Cost :0.8459 seconds
root@kali:/home/kai/pydictor# more /home/kai/pydictor/results/chunk_002728.txt
>
69863266e49b5ff8a7799efed53e0682
dc25f4c64e2b76fc7d1692bc7a042a26
8cc710fbd8d5b884f1c4bc253aacc10
4784cb1a1a73317c71ba5b6ce0f0a3ad
e3a161850ea4221fb6d3a04dfalc5cef
099fc6a44b3b5a7f7c9709e06083a15d

```

4) 语法引擎解析字典

语法引擎字典【--conf】

利用一些语法来生成解析字典，生成要求更加复杂的字典要求：

```

python pydictor.py --conf                                用默认
的"/funcfg/build.conf"文件建立字典
python pydictor.py --conf "[0-9]{6,6}<none>[a-f,abc,123,\\!\\@#]{1,1}
<none>" --encode md5 --output parsing.txt

```

举例1：--conf参数默认利用/funcfg/build.conf中的语法生成，在pydictor目录下查看该文件的内容如下

```

root@kali:/home/kai/pydictor/funcfg# ls
build.conf  extend.conf  leet_mode.conf  scratch_blacklist.conf  scratch_sites  sedb_tricks.conf
root@kali:/home/kai/pydictor/funcfg# more build.conf
# pydictor parser configuration file
# Build By LandGrey
# Copyright (c) 2016-2017 LandGrey (https://github.com/LandGrey/pydictor)
# License: GNU GENERAL PUBLIC LICENSE Version 3
#####
#####
#####
[boob,boob,B0B]{1,1}<none>_[0-9]{4,4}<none>@passwd
root@kali:/home/kai/pydictor/funcfg#

```

按照指定的语法生成字典截图如下所示

```

root@kali:/home/kai/pydictor# python pydictor.py --conf
abc,123,[0#]
pydictor 2.1.4.1#dev
pydictor目录下查
[+] A total of :30000 lines
[+] Store in :/home/kai/pydictor/results/conf_014231.txt
[+] Cost :0.5953 seconds
root@kali:/home/kai/pydictor# more /home/kai/pydictor/results/conf_014231.txt
bob_0000@passwd
bob_0001@passwd
bob_0002@passwd
bob_0003@passwd
bob_0004@passwd
bob_0005@passwd
bob_0006@passwd

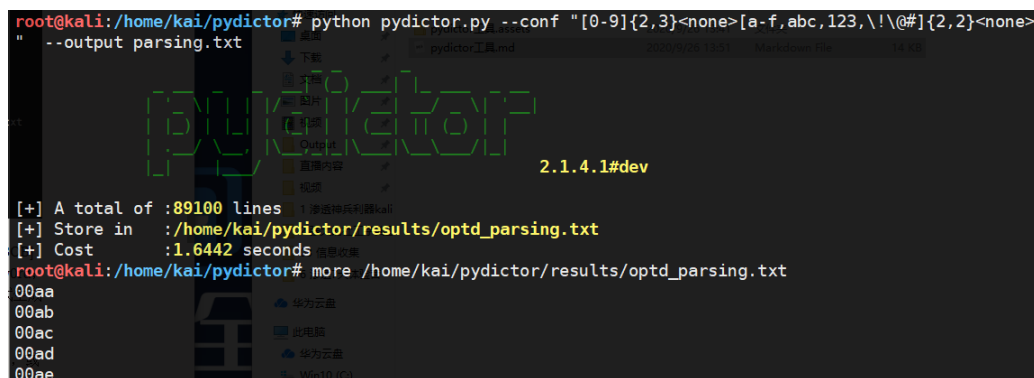
```


解释：可以按照类似正则表达式的样子生成对应的字典，可以看出[bob,b0b,BOB]字符三者其一，然后0-9的数字，构造长度为4的字符，最后末尾接上@passwd：组成所需的字典，该场景可以应用比较复杂的需求，写好表达式，按照表达式生成字典。

举例2：字典要求：开始为0-9数字构成，长度为2-3位，剩余部分为a-f字母中，或者abc, 123, !@#中随机一个构成，然后对生成字典md5加密

```
python pydictor.py --conf "[0-9]{2,3}<none>[a-f,abc,123,!@#]{1,1}<none>" --encode md5 --output parsing.txt
```

以下为不加密的情况：



```
root@kali: /home/kai/pydictor# python pydictor.py --conf "[0-9]{2,3}<none>[a-f,abc,123,!@#]{2,2}<none>" --output parsing.txt
pydictor
2.1.4.1#dev
[+] A total of :89100 lines
[+] Store in : /home/kai/pydictor/results/optd_parsing.txt
[+] Cost :1.6442 seconds
root@kali: /home/kai/pydictor# more /home/kai/pydictor/results/optd_parsing.txt
00aa
00ab
00ac
00ad
00ae
```

5) 规则扩展字典

扩展用法【-extend】

```
python pydictor.py -extend bob --level 4 --len 4 12
python pydictor.py -extend liwei zwell.com --more --leet 0 1 2 11
21 --level 2 --len 6 16 --occur "<=10" ">0" "<=2" -o
/possbile/wordlist.lst
```

6) 社会工程学字典

社会工程学【--sedb】

```
python pydictor.py --sedb
```

```

      _ _ _ _ _
    _ _ _ _ _ | ( ) _ _ | _ _ _ _ _
    | ' _ \ | | | / _ ' | | / _ _ \ | ' _ | | | | | | |
    | | ) | | | | ( | | | ( | | | ( | |
    | . _ / \ _ , | \ _ , | | \ _ \ _ \ | |
    | _ | _ | _ /
```

Social Engineering Dictionary Builder

Build by LandGrey

```
-----[ command ]-----
----
[+]help desc          [+]exit/quit          [+]clear/cls
[+]show option        [+]set option arguments [+]rm option
[+]len minlen maxlen  [+]head prefix        [+]tail suffix
[+]encode type        [+]occur L d s         [+]types L d s
[+]regex string       [+]level code         [+]leet code
[+]output directory   [+]run

-----[ option ]-----
---
[+]cname              [+]ename              [+]sname
[+]birth              [+]usedpwd            [+]phone
[+]uphone             [+]hphone             [+]email
[+]postcode           [+]nickname           [+]idcard
[+]jobnum             [+]otherdate          [+]usedchar

pydictor SEDB>>
```

根据以下信息:

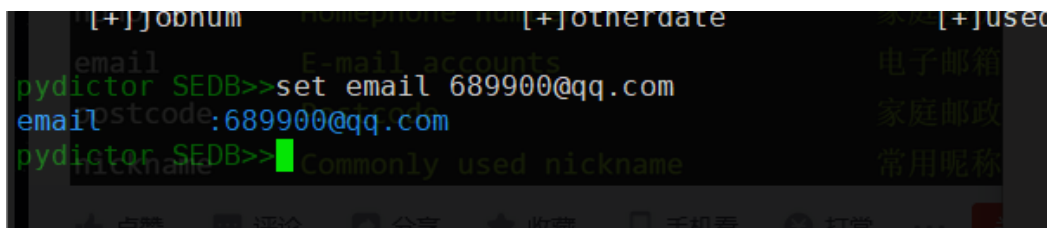
INFORMATION ITEMS	VALUE
chinese name	李伟
pinyin name	liwei
simple name	lw
simple name	Lwei
english name	zwell
birthday	19880916
used password	liwei123456.
used password	liwei@19880916
used password	lw19880916_123
used password	abc123456
phone number	18852006666
used phone number	15500998080
home phone	76500100
company phone	010-61599000
email account	33125500@qq.com
email account	13561207878@163.com
email account	weiweili@gmail.com
email account	wei010wei@hotmail.com
home postcode	663321
now place postcode	962210
common nickname	zlili
id card number	152726198809160571
student id	20051230

INFORMATION ITEMS	VALUE
job number	100563
father birthday	152726195910042816
mother birthday	15222419621012476X
boy/girl friend brithday	152726198709063846
friend brithday	152726198802083166
pet name	tiger
crazy something	games of thrones
special meaning numbers	176003
special meaning chars	m0n5ter
special meaning chars	ppdog

使用命令

```
python pydictor.py --sedb
set cname liwei
set sname lw Lwei
set ename zwell
set birth 19880916
set usedpwd liwei123456. liwei@19880916 lw19880916_123
set phone 18852006666
set uphone 15500998080
set hphone 76500100 61599000 01061599000
set email 33125500@qq.com
set email 13561207878@163.com
set email weiweili@gmail.com
set email wei010wei@hotmail.com
set postcode 663321 962210
set nickname zlili
set idcard 152726198809160571
set jobnum 20051230 100563
set otherdate 19591004 19621012
set otherdate 19870906 19880208
set usedchar tiger gof gamesthrones 176003 m0n5ter ppdog
```

例如设置邮箱email，如



设置sname值

```
pydictor SEDB>>set email 689900@qq.com
email:689900@qq.com
pydictor SEDB>>set sname 'shan shan'
sname : 'shan shan'
pydictor SEDB>>
```

show 查看已经设置的所有属性和值

```
pydictor SEDB>>show
len      : minlen: [no-limited] maxlen: [no-limited]
head     : [none]
tail     : [none]
encode   : [none]
occur    : letter: [ <=99 ] digital: [ <=99 ] special: [ <=99 ]
types    : letter: [ >=0 ] digital: [ >=0 ] special: [ >=0 ]
repeat   : letter: [ >=0 ] digital: [ >=0 ] special: [ >=0 ]
level    : [3]
leet     : [none]
regex    : [.*?]
cname    : Chinese name's phonetic
ename    : English name
sname    : 'shan shan'
birth    : Simple spellings phonetic
usedpwd  : Birthday [YYYYMMDD]
phone    :
uphone   : Used password
hphone   : Cell phone number
email    : 689900@qq.com
postcode :
nickname :
idcard   :
jobnum   :
otherdate :
usedchar :
pydictor SEDB>>
```

run 生成字典

```
pydictor SEDB>>run
[+] A total of :4346 lines
[+] Store in   :/home/kai/pydictor/results/sedb_062101.txt
[+] Cost      :0.1413 seconds
pydictor SEDB>>
```

(2) 插件型字典

plugin	birthday	P1	生日日期字典插件	ALL
plugin	ftp	P2	关键词生成ftp密码字典插件	ALL
plugin	pid4	P3	身份证后四位字典插件	ALL
plugin	pid6	P4	身份证后六位字典插件	ALL
plugin	pid8	P5	身份证后八位字典插件	ALL
plugin	scratch	P6	网页原始关键词字典插件	ALL

1) 一段时间内生日字典

生成长度6-8位的字典，要求生成字典是在19800101-20001231之间；

```
python pydictor.py -plug birthday 19800101 20001231 --len 6 8
```

```
root@kali:/home/kai/pydictor# python pydictor.py -plug birthday 19800101 20001231 --len 6 8
生成字典是在19800101-20001231之间;
py -plug birthday 19800101 20001231 --len 6 8
2.1.4.1#dev
[+] A total of :21270 lines
[+] Store in :/home/kai/pydictor/results/birthday_022446.txt
[+] Cost :0.5917 seconds
root@kali:/home/kai/pydictor# more /home/kai/pydictor/results/birthday_022446.txt
19800101
800101 pid4
198011 pid6 --encode b64
19800102 d8 --encode sha1 -o pid8.txt
800102
198012
19800103
```

2) 身份证后4/6/8位字典

```
python pydictor.py -plug pid4
python pydictor.py -plug pid6 --encode b64
python pydictor.py -plug pid8 --encode sha1 -o pid8.txt
```

```
root@kali:/home/kai/pydictor# python pydictor.py -plug pid4
more /home/kai/pydictor/results/birthday_022446.txt
2.1.4.1#dev
[+] A total of :10890 lines
[+] Store in :/home/kai/pydictor/results/pid4_022635.txt
[+] Cost :0.2299 seconds
root@kali:/home/kai/pydictor# more /home/kai/pydictor/results/pid4_022635.txt
0100 plug pid8 --encode sha1 -o pid8.txt
0101
0102
0103
0104
0105
0106
0107
0108 plug scratch
0109 文件中的多行 url 作为输入
```

3) 网页原始关键词字典

python pydictor.py -plug scratch 用/funcfg/scratch.sites 文件中的多行 url 作为输入；【不要求，基本不用】

```
python pydictor.py -plug scratch http://www.example.com
```

```
root@kali:/home/kai/pydictor# python pydictor.py -plug scratch http://www.baidu.com
pydictor 2.1.4.1#dev
[+] A total of :648 lines
[+] Store in :/home/kai/pydictor/results/scratch_022815.txt
[+] Cost :1.3606 seconds
root@kali:/home/kai/pydictor# more /home/kai/pydictor/results/scratch_022815.txt
Compatible
bdstatic
bdsug
important
shadow
underline
space
pcDirect
```

(3) 内置工具

tool	combiner	T1	字典合并工具	
tool	comparer	T2	字典比较相减工具	ALL
tool	counter	T3	词频统计工具	ALL
tool	handler	T4	筛选处理原有字典工具	ALL
tool	uniqbiner	T5	先合并后去重工具	ALL
tool	uniqifer	T6	字典去重工具	ALL
tool	hybrider	T7	多字典文件组合工具	F1 F2 F3 F4

1) 字典合并工具

经常从网上下载或者收藏各种的字典，利用字典的合并工具，可以把相同类型的字典合并成一个大的字典，利用工具生产2个字典（可提前把result目录下的生成的字典删除）

```
root@kali:/home/kai/pydictor# python pydictor.py -base dLc --len 2 3
pydictor 2.1.4.1#dev
[+] A total of :242172 lines
[+] Store in :/home/kai/pydictor/results/base_023212.txt
[+] Cost :0.7855 seconds
root@kali:/home/kai/pydictor# python pydictor.py -base dLc --len 2 2
pydictor 2.1.4.1#dev
[+] A total of :3844 lines
[+] Store in :/home/kai/pydictor/results/base_023218.txt
[+] Cost :0.0414 seconds
root@kali:/home/kai/pydictor# cd results/
root@kali:/home/kai/pydictor/results# ls
base_023212.txt base_023218.txt
root@kali:/home/kai/pydictor/results# cd
```

采用-tool combiner可以实现字典的合并，后面的参数为指定的目录，目录中存放的字典为默认要合并的字典

```
python pydictor.py -tool combiner ./results/
```

```
root@kali:/home/kai/pydictor# python pydictor.py -tool combiner ./results/
small.txt
[+] A total of :246016 lines
[+] Store in :/home/kai/pydictor/results/combiner_023242.txt
[+] Cost :0.0476 seconds
root@kali:/home/kai/pydictor#
root@kali:/home/kai/pydictor#
```

2) 字典比较工具

```
python pydictor.py -tool comparer big.txt small.txt
```

字典比较，比较2个字典的不同之处，用得相对较少；

3) 词频统计工具

用于统计功能还是比较实用，根据合并的字典中重复率最高的前多少条生成新的字典；

其中counter主要有2个参数，参数v和s，v表示对应的

```
-tool arg [arg ...]
combiner [dir]
comparer [minuend file] [subtrahend file]
counter ['v','s','vs'] [file] [view_num]
handler [file]
hybrider [file1] [file2] ...
shredder [file_or_dir]
uniqbiner [dir]
uniquifer [file]
```

```
python pydictor.py -tool counter s huge.txt 1000
python pydictor.py -tool counter v /tmp/mess.txt 100
python pydictor.py -tool counter vs huge.txt 100 -o fre.txt
```

```
root@kali:/home/kai/pydictor# python pydictor.py -tool counter vs ./results/base_023212.txt 100 -o
.txt
mess.txt 100
.txt 100 --encode
Word:1P -> 2 times
Word:1F -> 2 times
Word:1L -> 2 times
Word:Y1m -> 1 times
Word:Y1l -> 1 times
Word:Y1o -> 1 times
Word:Y1n -> 1 times
Word:SwU -> 1 times
Word:Y1h -> 1 times
```

4) 字典处理工具

```
python pydictor.py -tool handler raw.txt --tail @awesome.com --
encode md5
python pydictor.py -tool handler raw.txt --len 6 16 --occur "" "=6"
"<0" --encode b64 -o ok.txt
```

5) 安全擦除字典工具

```
python pydictor.py -tool shredder                                擦除当前输出目录
下所有字典文件
python pydictor.py -tool shredder base                          擦除当前输出目录
下所有以"base"开头的字典文件
python pydictor.py -tool shredder /data/mess
python pydictor.py -tool shredder D:\mess\1.zip
```

6) 合并去重工具

```
python pydictor.py -tool uniqbiner /my/all/dict/
```

7) 字典去重工具

```
python pydictor.py -tool uniqifer /tmp/dicts.txt --output
/tmp/uniq.txt
```

8) 多字典文件组合工具

```
python pydictor.py -tool hybrider heads.txt some_others.txt
tails.txt
```


(4) 使用场景

1) 字典合并

字典都不是凭空捏造或生成的，一般都会参考前辈们公布的字典。所以，先收集百八十个字典，放到一个目录下，把字典合并起来吧。

```
1. 合并目录/网站路径爆破字典2. 合并子域名字典3. 合并用户名字典4. 合并弱密码字典5. 其它各式各样的字典
python pydictor.py -tool combiner /my/dict/dirpath -o comb.txt
```

2) 词频统计

但是有时候我们通常不需要那么大的字典，选合并后字典的出现频率最高的前1000条保存吧。筛选出

```
最常用的网站路径/子域名/用户名/弱密码/...
```

修改lib/data/data.py中counter_split变量指定的分隔符(默认"\n")，也可以统计其它字符分隔的字典词频。

```
python pydictor.py -tool counter vs comb.txt 1000
```

3) 去除重复项

面对合并后的超大字典，还是不舍得只要频率高的词，路径字典有时候还是多多益善。去重下，照单全收

```
python pydictor.py -tool uniqifer comb.txt --output uniq.txt
```

或者直接合并加去重

```
python pydictor.py -tool uniqbiner /my/dict/dirpath --output
uniq.txt
```

4) 枚举数字字典

准备好字典了，拿最基础的试试手

```
1. 爆破4位或6位数字手机短信验证码2. 爆破用户名ID值
```

生成4位纯数字字典

```
python pydictor.py -base d --len 4 4
```

5) 简单用户名字典

不能确定是否存在某用户时，试试1位到3位的拼音字典，加上123456这样的几个弱口令，说不定就有意外收获：

```
python pydictor.py -base L --len 1 3 -o dict.txt
```

6) 后台管理员密码字典(明文传输)

经常遇到的测试场景了，就是一个登录页，把收集到的信息都用上，生成后台爆破字典，比如

```
域名:test.land.com.cn编辑名:张美丽、Adaor、midato公司名:上海美丽大米有限责任公司(如有雷同纯属巧合)座机:568456地址:xxx园区A座312室
```

把自己常用的弱口令字典复制到wordlist/Web 目录下，最终生成的字典会包含它们：

然后把下列信息写入/data.txt

```
testlandzhangmeilimeilizmlAdaormidatomeilidamimldmshmlm568456A312
```

生成字典：

```
python pydictor.py -extend /data.txt --level 3 --len 4 16
```

弱口令字典 + 部分信息 + 生成规则 + level 3，最终生成了七万多条密码，一部分密码如下：

7) 台管理员密码字典(前台普通加密)

有时候网站的密码可能不是直接明文传输过去的，程序员会用js简单加密下再传输过去，比如base64编码、md5加密，这时候可以用--encode参数生成加密字典

```
python pydictor.py -extend /data.txt --level 3 --len 4 16 --encode  
b64 python pydictor.py -extend /data.txt --level 3 --len 4 16 --  
encode md5
```

8) 后台管理员密码字典(前台js自定义加密)

高级点的程序员，还喜欢前端自定义个js加密方法，把用户名和密码加密后传输过去，比如

这时候，普通爆破工具基本都无能为力了，但是却依旧可以通过pydictor来生成字典：

修改/lib/fun/encode.py 文件的 test_encode()函数，用python语法仿照上图的加密方式再实现一遍加密：

```
def test_encode(item):    c = chr(ord(item[0]) + len(item))    for
i in range(1, len(item)):    c += chr(ord(item[i]) + ord(item[i
- 1]))    return quote(c)
```

然后运行命令，生成按照前端js加密方法加密后的密码字典，可以直接用burpsuite加载

```
python3 pydictor.py -extend /data.txt --level 3 --len 4 16 --encode
test
```

最后通过这种方式生成符合前端加密方法的用户名字典，先探测出存在的用户名，再结合几个弱密码，爆破出来100多个弱口令。

需要注意的是，一般生成加密字典前要生成一个没加密的字典，因为每一项在文件中的顺序是一致的，所以爆破出来密码后，可以通过行数对照去没加密的字典中查找明文。

9) 复杂格式的字典

例如，你通过shoulder hack和一些信息，猜到别人的密码大概是

```
Cxhai【三位或四位数字】_abc123@【qq,163,wy,mail中的一个】，然后md5加密的值
```

这种复杂格式的字典，pydictor也可以轻松的生成

```
python pydictor.py --conf "Cxhai[0-9]{3,4}
<none>_abc123@[qq,163,wy,mail]{1,1}<none>" --encode md5
```

没加密前的字典：

最终加密后的字典：

10) 社会工程学字典

通过配置文件定义的规则的一部分内置代码逻辑，你可以输入一些关于个人的信息，生成关于某个人可能用的密码，比如，我只知道一个的如下信息

```
姓名： 景林生日：1997年7月16日以前用过密码：jlin520
```

然后一波操作，生成了四万多条密码

嫌密码太多了？没事，只要长度6-16的，级别设置大点，密码会少很多；

查看下当前配置，重新生成字典，只有三千多条了

11) 处理自己的字典

退一万步来讲，上面的字典都帮不了你，但是pydictor的handler功能还是可以帮你根据具体的使用场景来处理自己的字典，让自己原本的字典适用各种场合。

比如：

对方密码策略要求是6到16位；必须有数字和字母，不允许有特殊字符；前端js对密码base64编码后传输到后端。

可以用下面的命令处理自己原先的字典raw.txt，生成符合本次爆破场景的字典：

```
python pydictor.py -tool handler /wordlist/raw.txt --len 6 16 --  
occur ">0" ">0" "<=0" --encode b64 -o /wordlist/ok.txt
```

