Generating Wordlists

Peleus

Cracking passwords has two aspects that need to be considered when taking into account how likely it is to reveal the information you need. They are defined as follows:

- **Efficiency** – The likelihood that your password set has the candidate password within it.
- **Power** – How many attempts / guesses you can make per second, minute / random time frame.

With the increase in GPU crackers, oclHashcat being my favorite, a large emphasis has increasingly been put on power as opposed to efficiency. People suspect that because they *can* throw a wordlist of 1 billion entries against a hash that it's the optimal solution. I'm not saying that you shouldn't try it as your last resort, but perhaps there is a better way to put the odds in your favour.

Obtaining a Relevant Password List

The best tool for this job is going to be CeWL (Custom Wordlist Generator). It has been designed to spider target websites for key words and compile them into a word list for usage later. You can have a lot of control over the spider such as how many links it should follow, the minimum word length and even supports different authentication schemes to crawl restricted area's you have access to. Let's use a example of this website and see what word lists we can generate.

```
cewl http://netsec.ws/ -d 1 -m 6 -w netsec.txt
```

Breaking this down we'll be crawling netsec.ws and (-d) 1 link layer deep from the main page. The minimum length of words we're going to be keeping is 6 characters, and we're saving the output to a text file netsec.txt. Testing the result we have accumulated a lot of passwords directly related to netsec.ws and it's content.

```
wc -l netsec.txt
1741 netsec.txt
```

Building Off a Solid Foundation

Now we have a solid list of candidate passwords we often want to build off this by mutating the passwords according to particular rules. John the ripper provides awesome functionality for this with their wordlist rules. They can be viewed and added to in the file located at /etc/john/john.conf under '#Wordlist mode rules'. Some examples are,

```
# Try words as they are
:
# Lowercase every pure alphanumeric word
-c >3 !?X l Q
# Capitalize every pure alphanumeric word
-c (?a >2 !?X c Q
# Lowercase and pluralize pure alphabetic words
<* >2 !?A l p
# Lowercase pure alphabetic words and append '1'
<* >2 !?A l $1
...
```

We can even add our own rules to the list according the john's syntax which can be read about here. As an example let's say we wanted to add on 1 or 2 numbers to the end of the passwords we have in our list. We can add the following.

```
# Add one number to the end of each password
$[0-9]
# Add two numbers to the end of each password
$[0-9]$[0-9]
```

We can now use john to perform modifications according to these rules with the following command.

```
john ---wordlist=netsec.txt --rules --stdout > netsec-mutated.txt
```

Now when we have a look at our new wordlist, we can see our password's have grown from 1741 to 273106 variations of them.

```
wc -l netsec-mutated.txt
273106 netsec-mutated.txt
```

With your new targeted wordlist often you'll have a much higher success rate against the hashes you're trying to crack. We've got machines that can make 1 million guesses a second, why not feed it in something good?

Filed Under: Passw ords
Tagged With: cew l, cracking, john, passw ords, w ordlist