# Advanced Password Cracking

During the cryptographic engineering lectures you have learned how to build a custom password cracker out of `passwd` command of `openssl`. It was clever but rather inefficient. We are going to learn how to use `john the ripper`. This is the motivation to work on web crawler and how to collect information on potential targets. It has an impact on people privacy but it is also very useful for economical intelligence...First, we learn how to use John the Ripper.

## 1. Installing John the Ripper

There are two versions of `john`:

- the standard version `http://www.openwall.com/john/j/john-1.8.0.tar.gz`
- and the community version called `jumbo` `http://www.openwall.com/john/j/john-1.8.0-jumbo-1.tar.gz`

You can use

```
tar zcvf john-1.8.0-jumbo-1.tar.gz
cd john-1.8.0-jumbo-1/src
```

You can now start to compile `jumbo` using `automake` and `autoconf` tool:

```
./configure
```

The `autoconf` tool analyzes your system to provide you with the best setup for `jumbo`. If you have tried to install the standard version you need to figure out the configuration by yourself. Some optional libraries may need to be installed to get extra functionalities and improved performance.

```
make clean && make -s
```

After a few minute of compiling, you will get all the executables located in the `run` directory of `john-1.8.0-jumbo-1`. If you want to modify your `PATH` to include the `run` directory of `john` (you can also move it but you need to be `root`). If not you will need to give the path of `john` or stay in the `run` directory and use `./john`.

```
echo $PATH
export PATH=$PATH:~/MYPATH/MYDIRECTORY
```

You need to add the previous line to the end of your `.bashrc` file to make permanent.

## 2. Using John the Ripper

As any good password cracker, `john` has two core engines: one to create password candidates and one to perform efficiently cryptographic operations.

Instead of having a single candidate generator, `john` has 4 :

- `single`,
- `wordlist`,
- `incremental`,
- `markov`.

Using `john` is relatively simple:

```
john --single --format=FORMAT fingerprint
```

`john` has three very important files:

- the "pot" in which all the guessed passwords are stored, it can be set to a given file using `-pot=mypot`;
- the setup file `john.conf` located in the `src` directory if you have built it yourself or in `/etc/john` if you have used a package;
- and the session file `john.rec` used to log `john` activity.

and

The first option of the previous command is the mode used to create candidates. The second option is the format of the fingerprint. The following command gives you all the fingerprint formats supported by `john`.

```
john --list=formats
john --list=format-details
```

**Question 1:** How many formats are supported by `john` ?

To see the passwords recovered by `john`, you must use the `-show` to observe the content of the pot.

```
john --help
john --show --format=FORMAT monchallenge
```

**Question 2:** You can start to attack the password contains in `challenge.md5.gz` and `challenge.sha1.gz`. What do you obtain ?

After `single`, you can switch to the `wordlist` mode. You can first download wordlist on `https://wiki.skullsecurity.org/index.php?title=Passwords` or on `https://crackstation.net/`. Be careful, some files are really large.

**Question 3:** From a file containing only the word "roch", find an option of `john` to print in the standard output all the password candidates tested by `john`. Same thing when you use the `-rules` option (mangling).

**Question 4:** Modify `john.conf` to change the mangling in the `wordlist` mode. Your new rules must happen a year between 1900 and 2016 at the end of the candidate (rules detailed at `http://www.openwall.com/john/doc/RULES.shtml`).

**Question 5:** How work `single`, `incremental` and `markov` from your understanding of `john.conf` ?

**Question 6:** How are the other cracker provided by `john` ?

Creating appropriate rules and good wordlist is critical for `john`. This is our main motivation to explore tomorrow web crawlers. Now, we need to work on `python`.