Dr. Yanwei Wu
Department of Computer Science
Western Oregon University

November 4, 2017

LAB 3 — Password Cracking
In this lab, you will use use John the Ripper to crack passwords.

Due: 12:00am Thursday, October, 2017.

Problem 1. John the Ripper – a password cracker

John the Ripper is a fast, multi-platform password cracker.

1 Setup

1.1 Download John the Ripper

If you're using Kali Linux, this tool is already installed. Download John the Ripper free version here `http://www.openwall.com/john/`, such as John the Ripper 1.8.0 (sources, tar.gz, 5.2 MB). Once downloaded, extract it with the following linux command:

```
tar zxvf john1.8.0.tar.gz
```

1.2 Compile John the Ripper on a Unix-like system

Enter the directory into which you extracted the source code distribution of John. Enter the "src" subdirectory and invoke "make" to obtain a list of operating systems for which specific support exists:

```
cd src
make
```

Note the make target for your system and type:

```
make clean SYSTEM
```

where SYSTEM is the appropriate make target. Alternatively, if your system is not listed, use:

```
make clean generic
```

If everything goes well, this will create the executables for John and its related utilities under "../run/". You can change directory to there and start John, like this:

```
cd ../run
```

./john –test

Alternatively, you may copy the entire ”run” directory to anywhere you like and use John from there.

2 Sample usage of John the Ripper

(a) First, you need to get a copy of your password file. If your system uses shadow passwords, you may use John's "unshadow" utility to obtain the traditional Unix password file, as root:

  umask 077

  unshadow /etc/passwd /etc/shadow > mypasswd

  (You may need to replace the filenames as needed.)

  Then make "mypasswd" available to your non-root user account that you will run John under. No further commands will need to be run as root.

  If your system is ancient enough that it keeps passwords right in the world-readable /etc/passwd, simply make a copy of that file.

(b) Now, let's assume you've got a password file, "mypasswd", and want to crack it. The simplest way is to let John use its default order of cracking modes:

  john mypasswd

(c) If you've got some passwords cracked, they are stored in /JOHN/john.pot. The john.pot file is not meant to be human-friendly. You should be using John itself to display the contents of its "pot file" in a convenient format:

  john –show mypasswd

(d) You might prefer to manage the cracking modes manually. It is wise to start with "single crack" mode:

  john –single mypasswd

(e) To catch weak passwords not derived from readily available users' personal information, you should proceed with cracking modes demanding more processor time. First, let's try a tiny wordlist with word mangling rules enabled:

  john –wordlist=password.lst –rules mypasswd

(f)

3 Configuration file

See RULES for information on the syntax at `http://www.openwall.com/john/doc/RULES.shtml`.

(a) Let's assume that you notice that in some password file a lot of users have their passwords set to login names with "?!" appended. Then you just make a new "single crack" mode rule and place it somewhere near the beginning:

[List.Rules:Single]
Az"?!"

Hint: if you want to temporarily disable all of the default rules, you can simply rename the section to something John doesn't use and define a new one with the section's old name, but be sure to leave the "List." prefix of the name intact to maintain correct configuration file syntax.

All the same applies to wordlist mode rules as well.

(b) If you generate a custom charset file (described above) you will also need to define a configuration file section with the "incremental" mode parameters. In the simplest case it will be like this (where "Custom" can be replaced with any name you like):
[Incremental:Custom]
File = custom.chr

This way, John will only use characters from passwords used to generate the charset file only. To make John try some more characters, add:
Extra = !#$%

4 Sample Password Hashes

A group called KoreLogic used to hold DEFCON competitions to see how well people could crack password hashes. Their contest files are still posted on their site and it offers a great sample set of hashes to begin with. Download the password hash file bundle from the KoreLogic 2012 DEFCON challenge. Or use this mirror. Extract the file using this linux command:

```
tar jxf cmiyc_2012_password_hash_files.tar.bz2
```

This expands into 19 different hashdumps including des, md5, and ntlm type encryption. Each of the 19 files contains thousands of password hashes. This should be a great data set to test our cracking capabilities on.

5 Lab work

This will do a "single crack". It is the most basic cracking scheme john has. It will crack something like "admin:admin".
john –wordlist=password.lst passwd.crack

To find out what those rules actually are you'll want to check out the file john.conf.

Question: Create 10 passwords according to the rules and use john to crack at least 8 of the 10 passwords.

Turn in the screenshot via the hw3 submission link on moodle.

6 Extra work

This will use your wordlist with some additional default rules.
john –wordlist=password.lst –rules=Extra passwd.crack

A sample rule:

```
!?A >[1-6] l i\0[a-z]
```

* !?A : do not run on words with capital letters
* >[1-6] : for each 1 to 6, if the word is greater than that length do the following (note, John will not accept two digit numbers for either of the bounds)
* l : convert to lowercase (?)
* i : insert at the following index the following character
* \0 : references result of range [1-6]
* [a-z] : for each range [a-z]

Effectively this will insert all letters a-z from position 1-6 in the word, hash it, and compare it against the hashes you're cracking.

Question: Write at least one rule of your own and crack the password of this type.

Turn in the screenshot via the hw3 submission link on moodle.

Submitted by Dr. Yanwei Wu on November 4, 2017.