Use John The Ripper to crack password hashes

I often hear rumors about how fast a password hash (such as a linux passwd/shadow hash) can be cracked today by using modern GPU's. Basically what I heard recited by many was, that a kid with a fast gamer notebook with a decent graphics card could brute-force a password hash within hours by leveraging the power of the GPU.

I am not a security expert and by no means a Hacker, but i've wanted to try the script-kiddie approach myself hands on for a long time already. I was recently tasked with stress-testing some systems in our lab with significant compute power. One of them was a 128 core AMD ROME based server, the other one one was a GPU server with 8x NVIDIA RTX 2080Ti in it. So finally i took this opportunity to verify or bust the myth, that a teenager with a gamer notebook can crack your password hash in a few hours.

Well Spoiler alert: the myth is absolutely busted, IF your password is a safe password, meaning, it is not on a password list and of significant length. Once the wordlist based approach fails, the only option left to track is to brute-force the password, which means, to try all possible combinations of all possible characters of a password. Now if we know a password policy, that actually might help to narrow down the possible passwords and accelerate the brute force attack slightly by being able to remove un-allowed passwords (like for example all passwords that don't contain at least one upper and lowercase letter and digit, if our policy demands all three are present in a password). So ironically, the more restricting your policy, the less permutations are possible and the easier it is to crack your password.. so be careful what you wish for by publishing password policies ;)

So again, once we are past the wordlist approach and we need to start brute-forcing, things get time consuming. For an 8 character password without any policy limiting the possible permutations, we are already talking about **3'025'989'069'143'040 possible passwords**! At a rate of 1.5 Million passwords per second (that's what the 8 GPU server achieved using John the ripper), it would still take about 64 years to test all possible permutations for a single user. of course you might get lucky and your first guess is the right one, but you can see where this is going. **On average** this means, that an 8 character password will be cracked within about **32 years** of calculating hashes on the before mentioned system with 8 quite fancy (by today's standards) GPU's. So if you have a cluster of 32 of these machines, you may be able to crack the password within less than 2 years for sure. BUT: keep in mind the power consumption and the cost involved with this: the 8 GPU machine consumes 2800 Watts while calculating password hashes at the said 1.5 Million passwords per second. Running such a system for 32 years at a power cost of 18 cents per kWh would cost you **$141'281**. This number does not change if you add more systems to your cluster. You may get there faster, but you will still use the same amount of energy. If you need to cool your server with an AC, you can even multiply this number by about 1.5 to take the power consumption of your AC into account as well. And this number is only for the AVERAGE cost, not the maximum possible cost! So at the end of the day, if you just want to hack somebody's password in order to extort money from that person or company, make sure they have enough money to give and that their data is worth enough before you spend all that money an cracking a password. also make sure you get enough machines so you can beat the target's password lifecycle ;)

In any case, Ideally, you should start by first hacking into some large GPU based clusters in order to get massive calculating power for free, this will help with the profitability of such an attack :)

so can a kid with a gamer notebook crack your safe password within hours? most probably not!

Things look quite different, if your password is on one of those lists.. with 1.5 million passwords per second, those lists are processed quite fast ;) so you better choose a safe password for your critical applications. The longer the better of course. Even better than that: don't allow password logins in the first place.. especially not for users with fixed names such as `root` on linux.

In any case, here is the full description how i set up the various systems and what performance i got out of them using John the ripper, one of the most popular password hash cracking utilities.

https://github.com/openwall/john

prepare shadow and passwd files for john

first unshadow your `passwd` and `shadow` files

```
../run/unshadow ~/tmp/passwd ~/tmp/shadow > ~/tmp/unshadow
```

use john with openMP on multiple cores

```
export OMP_NUM_THREADS=8
john ~/tmp/unshadow
```

use john with multiple cores by forking the process

```
john --fork=8 ~/tmp/unshadow
```

query the results

```
john --show ~/tmp/unshadow
```

compile john with MPI support to run on clusters

ubuntu dependencies

```
sudo apt install libopenmpi-dev openmpi-bin
```

centos dependencies / preparation:

```
yum install openmpi-devel openmpi openssl-devel
```

```
module load mpi/openmpi-x86_64
```

get source and compile:

```
wget https://github.com/openwall/john/archive/1.9.0-Jumbo-1.tar.gz
tar xvf 1.9.0-Jumbo-1.tar.gz
cd john-1.9.0-Jumbo-1/src
./configure --enable-mpi
make -s clean && make -sj4
```

now run it :)

```
mpirun -np 8 -host
localhost,localhost,localhost,localhost,localhost,localhost,localhost,localh
ost ../run/john ~/tmp/unshadow
```

to check the progress run

```
kill -USR1 $(pidof mpirun)
```

compile with OpenCL support to run on NVIDIA (and other) GPU's (tested on CentOS)

download and install cuda if you haven't already

```
wget
https://developer.download.nvidia.com/compute/cuda/11.2.0/local_installers/c
uda_11.2.0_460.27.04_linux.run
sudo sh cuda_11.2.0_460.27.04_linux.run
wget https://github.com/openwall/john/archive/1.9.0-Jumbo-1.tar.gz
tar xvf 1.9.0-Jumbo-1.tar.gz
cd john-1.9.0-Jumbo-1/src
./configure LDFLAGS=-L/usr/local/cuda/targets/x86_64-linux/lib CPPFLAGS=-
I/usr/local/cuda/targets/x86_64-linux/include
```

the summary should show that OpenCL support is now enabled (yes)

```
make -s clean && make -sj4
```

now let's run it :) here is an example where I ran john the ripper on a server with 8x NVIDIA GeForce RTX 2080 Ti:

```
run/john --format=sha512crypt-opencl -dev=gpu -fork=8  ../root.unshadow
```

the `–format` option needs to be specified in order to use the GPU at all. without specifying a format, john will always default to the CPU implementation of the crypt algo. to figure out which format to use, start john without a format parameter and then look at the output to find the crypt that was used. now run

```
run/john --list=formats --format=opencl
```

to get a list of all crypts that support opencl. if you are lucky, the one you are looking for is in there as well :)

the `-dev=gpu -fork=8` options are there to use all cards in parallel. this will fork 8 individual processes each working on their own range of passwords at a time and each on a different GPU. If you have multiple hosts with GPU you may use MPI for that.

continue an interrupted session

John saves its status as it's working, so in case it crashes or you have to abort it because you need to work with your pc and don't want the cpu load on it for example, you can always resume the session and continue where John has left off.

BUT.. it is **important** that you specify the `–restore` option, otherwise john will start over again! To avoid accidently starting over and by doing so loosing all the previous compute time, it is advisable to give each session its own **session name** by using the `–session:<name>` parameter. By doing so, you can avoid overwriting the session in case you start another john session in the mean time.

so you would start john like this:

```
john --session:sess1 unshadow.txt
```

and then resume the session like so:

```
john --restore:sess1
```

also the other commands like for example `status` accept a session name parameter:

```
john --status:sess1
```

Performance examples

if you press any key during the run, you will get a status showing you c/s (crypts per second). here are a few numbers from the systems i had access to at the moment of writing this article:

CPU / GPU	c/s rate	method used
8-core i7-8809G CPU	7'500	OpenMP
128-core (2 socket) AMD ROME 7742	150'000	OpenMP
8 x GeForce RTX 2080 Ti	8×190'000	Fork