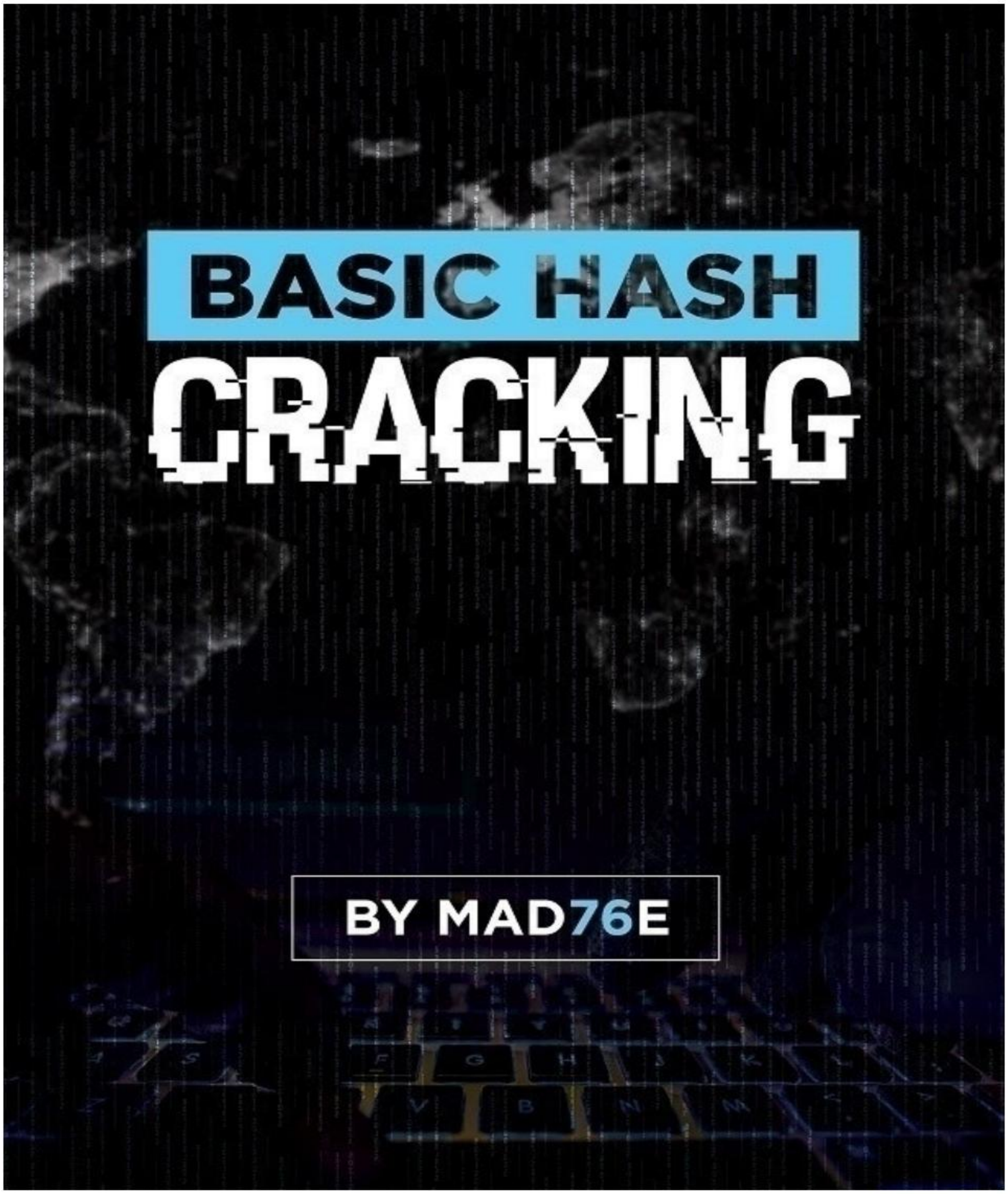
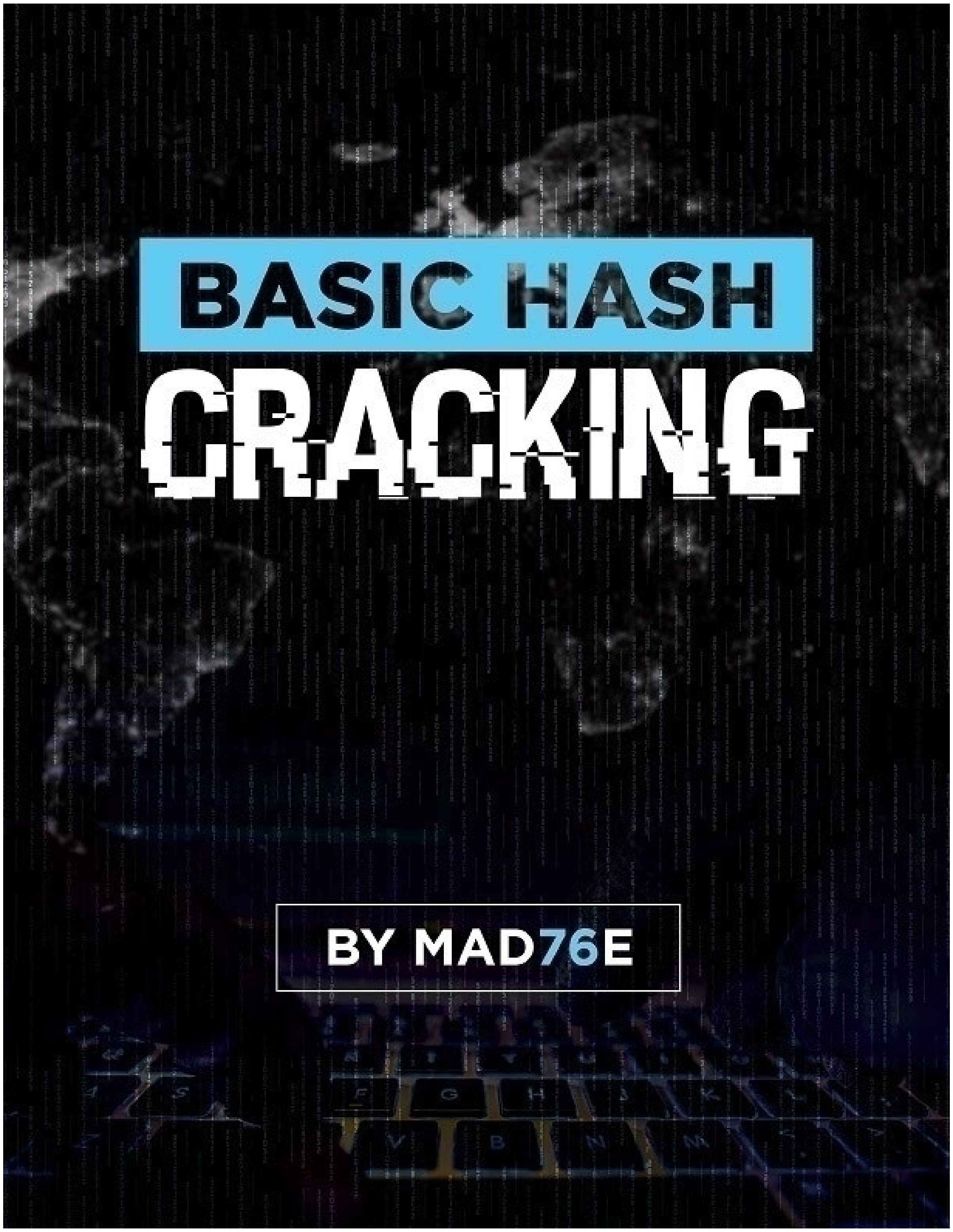
BASIC HASH CRACKING

BY MAD76E





BASIC HASH CRACKING

BY MAD76E

Basic Hash Cracking

Written by Mad76e

Dedicated to a group that use to be the Elite, but lost it due to

LEGION

WE - ARE - LEGION

"... I'm your dream, make you real
I'm your eyes when you must steal
I'm your pain when you can't feel
Sad but true

I'm your dream, mind astray
I'm your eyes while you're away
I'm your pain while you repay
You know it's sad but true, sad but true

You, you're my mask

You're my cover, my shelter
You, you're my mask
You're the one who's blamed...”

First printing 2016

978-1-365-59988-0

Mad76e

Please note:
This book is written from a hacker’s perspective, language, value and ideas may not be the same as the authors

Never try to crack anything without the owner’s written approval. All of the content in this book is 100% legal to do as long as you have permit from the business owner / affected user and the administrator to do so, this include old dumps of databases that's already well known, you need permission to crack those as well. This book is NOT an invitation for you to commit crimes, also before doing anything it's in your interest to check with your countries laws what applies

Introduction

This book is written to those who interested to learn basics of cracking, especially in and to use it to your favor to various hashes and I'm doing it in the old fashion way inside just for you to get a grip about the With this book we targeting ethical hackers that already got permission from the owner / administrator / user to test their passwords. Also it's a good reason for the administrator to check that he have set the right password policies.

Normally before a pentester do this, there's a legal document that both sides need to sign. In this document there are clear instructions about what we can do and what not to do, and this is very important. If breaking this written document the pentester might face heavy fines or jail time, its serious things at least in the

Now what is Hashcat, you may wonder? And the answer is that it's a multi platform password recovery tool that will crack almost anything between SHA1 to your Bitcoin wallet. We will also use other tools in our way to crack our password / key, more about them

This book is all about cracking, which means that we trying to guess or brute force our hash to get the password with different attacks. We will use mask attack, hybrid attacks and wordlist attacks against different types of hashes, with or without

To run Hashcat you will need at least one or preferably more high end ATI or NVIDIA graphics card with PCI-express slots on the motherboard. Preferably we need an I7 CPU that supports OpenCL instructions (but that is no requirement), let's talk about that later. Back in the day there were 2 types of one that was supported by the graphics drivers, called OclHashcat (ATI GPU version) or CudaHashcat (NVIDIA GPU version) and one without acceleration for CPU, called Hashcat. Well times tend to change things and now we got CPU with some lower type of graphics instructions integrated that made it possible to accelerate the speed at the CPU because of the "Intel OpenCL runtime" software which means that your CPU can work with OCL related without needing a GPU!! The crew that develop Hashcat decided to go with OpenCL, because the performance was better with OpenCL than the old way, so they changed the name of the product to just Hashcat. Nowadays both and newer are accelerated. You can choose to run both for best performance or just GPU. Both have its pros and cons. Older graphics cards are also supported in the new Hashcat 3.xx, however for some reason it delivers lesser speeds than old version 2.xx, so my recommendation is to stay with the old 2.01 if it's possible. Same commands that are used here are working with older but you need to change the executable file from **hashcat64.exe** to **oclhashcat64.exe** as example

Now, just a reminder, running with only raw CPU power and get a couple of thousand k/s is rather stupid when we can do this much faster with modern software. Please, for your own sake avoid old and ineffective ways to crack hashes update to latest Hashcat and install OpenCL runtime. The difference is enormous if your CPU supports

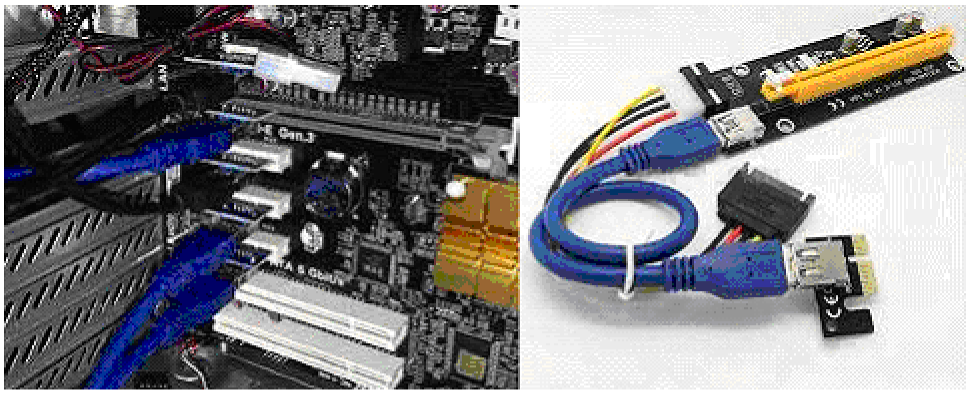
Hardware

I figured it was time to take a closer look on the computer itself that we will be using to crack our hashes with. I’m going to be very brief here. You will need a cracking server or a gaming computer with one or multiple GPUs (Graphical Processing Units) if you want to crack some of these hashes! If are running cracking without a graphics card you will discover that this can be a slow method compared to the other, depending on which hash you trying to crack, so we need to speed things up a bit. Now with a modern computer you can have up to 4 or more graphics card installed on your motherboard. It all has to do how much you’re willing to pay. My three graphics cards are slow compared to the standard but they get the job done, and that's what's More graphics cards or a faster graphics card plus a modern CPU will do the task much faster, however the cost will be I do recommend at least 2 graphics cars to start

First of all, we need a computer chassis or a metal frame, and a motherboard that has one or at least 2 PCIe sockets (PCI Express). <u>The speed on the PCI-express really matter, because</u>

were never going to use the whole bandwidth We also need one or more end graphics cards as AMD or NVIDIA. And that's the expensive part. We need at least 4GB in ram, preferably 8GB in Desirable is 4GB ram per graphics + 4GB to the **64bit OS** and perhaps the cheapest i7 or i5 CPU. We also need a hard drive just for the OS, and the wordlists, and we need raw power to keep the rig running, preferably one or two power supply units. here is a "Corsair HX1000i" which depends on how many graphic cards you are It's important to have a stable system, so leave at least 250watt to the system i.e. motherboard, CPU, hard drive, WiFi-stick etc. A high end graphics card usually wants around 250 to 450 watts each, depending on graphics model but it's better to check that out for sure. Goggle the vendor and wattage the specific card draws with full

We also must talk about PCI-Express risers. A PCI-express riser is a cable between the motherboards PCI-express socket and the graphics card. The reason we use this is the problem with heat that occur when using multiple graphics cards. The motherboard can't breathe properly, and the temperature on the motherboard Remember that each graphics card can reach about 75-90 degree Celsius. To avoid trapping heat we can build a frame and lift the cards a bit the motherboard, just to make the air flow better, plus that you can have greater distance between the cards. Now there's powered risers and non-powered ones. I suggest that you use an USB powered riser that only carries data back to the motherboard, reason for doing this is to avoid back feeding when running with multiple power supply units that can damage the motherboard.



Here is what I I use a 16x to 1x PCIe powered riser (not my computer on picture though)

If you damage the motherboard and the motherboard manufacture wants to know how that could the warranty might not be valid if you tell them you doing. So be careful playing around with multiple power supply units, unless the motherboard and the computer case have support for this or you're absolutely sure what you're

Above you see a mining rig. The same configuration is used when building a cracking rig, the only thing differ is the

software

So now we have almost all of the hardware. Now it's important to install some kind of network interface card to communicate with the network. To save energy I choose a WiFi-stick. My cracking rig has no keyboard or mouse or monitor, it's just a stand alone computer box.

More about RAM memory… Cracking doesn't use as much ram because most of the processes regarding cracking happens in the GPU or in the GDDR memory banks in the graphics however running with a big wordlist with rules or gigantic hash lists with millions of hashes it will affect memory badly, so it doesn't hurt to have a lot of RAM memory. If you're running with small databases and perhaps one graphics card you will be okay with 4gb, and you see any big difference in cracking speed between 4gb and 8gb as long as you don't run hundreds of thousands of hashes with salt at the same time or with rules, then 4gb isn't

My hardware

In this book I have chosen to work with two mid gain and low gain graphics card. you want something that really rocks your world you have to have those more expensive cards that are in the range of $350 and more especially NVIDIA. Reason for using a mid gain graphics card is the fact that you actually can crack with those as well, and you don't need to get poor to get reasonably performance, and I want to show to you all that you can crack something with these. Now I am an AMD fan, and AMD is a bit than NVIDIA nowadays, and I would go to NVIDIA long time ago if I was just running with one card. Normally nowadays I'm running with multiple graphics cards and I feel I'm not willing to waste a good mid gain graphics card.

Also I will provide a couple of below, with differences between a low gain graphics card and a medium gain graphics card. A note is that what I provide here is a rough guide to what you would expect of a graphics card in mid gain. So think of it as + -12%, depending on drivers, hardware and brand. To be honest all MD5 passwords using Latin script (our alphabet) and numbers 0-9 in a random order 8 characters long capital and small letters at random places in a password is cracked by this in 16 hours or so likewise a 13 characters long numeric MD5 password is cracked around 40 minutes, just to give it some thought how powerful these really are. A high gain card would do this extremely faster as you might understand. To be truthful a 9-10 character random password with numbers and letters isn't safe anymore, perhaps against one high end graphics card, but not against 4 or more graphics cards. A fellow cracker friend on a forum I often visit, runs with "8 old Radeon HD the graphic card itself isn't new but the card still gives a big punch, because still delivers good performance. He manages to pull out 78GH/s from those 8 graphics cards running MD5 with these 2 rigs. There's not a single MD5 password safe with fewer than 9 characters if he wanted to. Please note; running with high gain graphics card will use a lot of power. 150-350w for each card is not uncommon depending on model, so yeah my friend is actually using 2 PSU in every rig. I will do a fast comparison between a low gain graphics card and a medium gain graphics card, so you get an idea how big the difference really is. (R9 380x) against (R7 260X OC)...

OC)... OC)...
OC)...
OC)...

OC)...
OC)...

H/s = Hashes per second

kH/s = Kilo Hashes per second

MH/s = Mega Hashes per second

GH/s = Giga Hashes per second

Software

First off, we need to decide if were running this in Windows or Linux. Just simplify it for the beginner I'm running this in Windows7 64bit, and that's the main OS I prefer to use in here to simplify things. Also we need to download hashcat from their site. Also download BINARYS not source code

Hashcat
https://hashcat.net/hashcat/
We don't need to install anything, just simply unpack and it's ready to go. When writing this book, the latest hashcat version is 3.10

NVIDIA users requires ForceWare 346.59 drivers or later
http://www.geforce.com/drivers

AMD users requires 14,9 drivers or later. 15.12 are one of the best for this (even for new graphics cards for some reason.)
http://support.amd.com/en-us/download **<-- or from any other place**

Intel users require Intel OpenCL Runtime 14.2 or later
https://software.intel.com/en-us/articles/opencl-drivers

Keep an eye to the site and the drivers. There's still an ongoing work to perfect and improve cracking

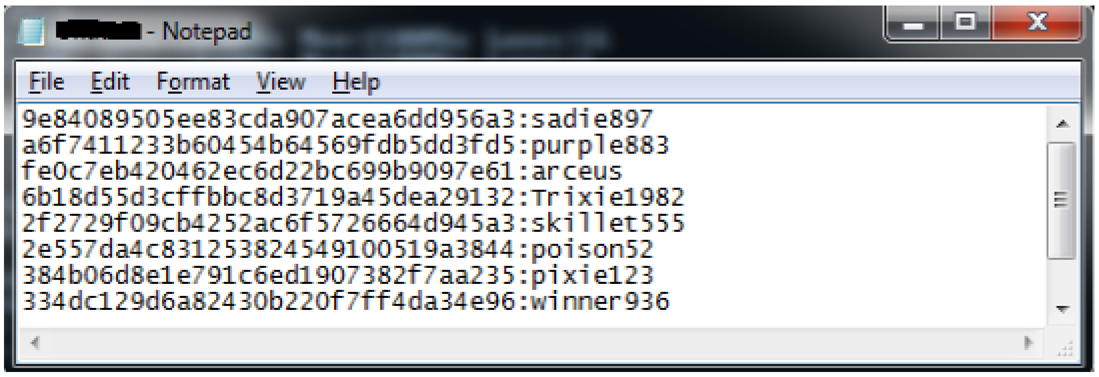
I do use so I can check the status from my laptop, so I'm not using any mouse or anything. Everything I do, as transfer files and the rig I do from it feels comfortable to connect to

your cracking rig from your mobile when you’re out of town,
just to check how the cracking is

First look at Hashcat

The installs are done and we have unpacked Hashcat in our folder of choice. Now in the hashcat folder you have a couple of files worth mention that are executable. Those files are hashcat64.exe / hashcat32.exe depending on if you are running a 32 bit OS or a 64 bit OS. You also have hashcat32.bin / hashcat64.bin that are intended for Linux users. Keep in mind that if you having more than 4GB in memory it's required to change OS to a 64bit, because the 32 bit OS can max allocate 4GB of memory. Folder of interests is the "rules" folder, more about that later, however it doesn't hurt take a look in that rule folders and open one of the files inside the folder to take a look at how a rule file may look like.

One important file were going to look at is the pot That file is located in the folder. All cracked passwords will be saved in the pot file. Look after a **hashcat.pot** file, there you will find hashes and passwords you This file should be empty right now but here is where all hashes and passwords is collected when were cracked something. Here I have cracked a couple of hashes just to show you an example



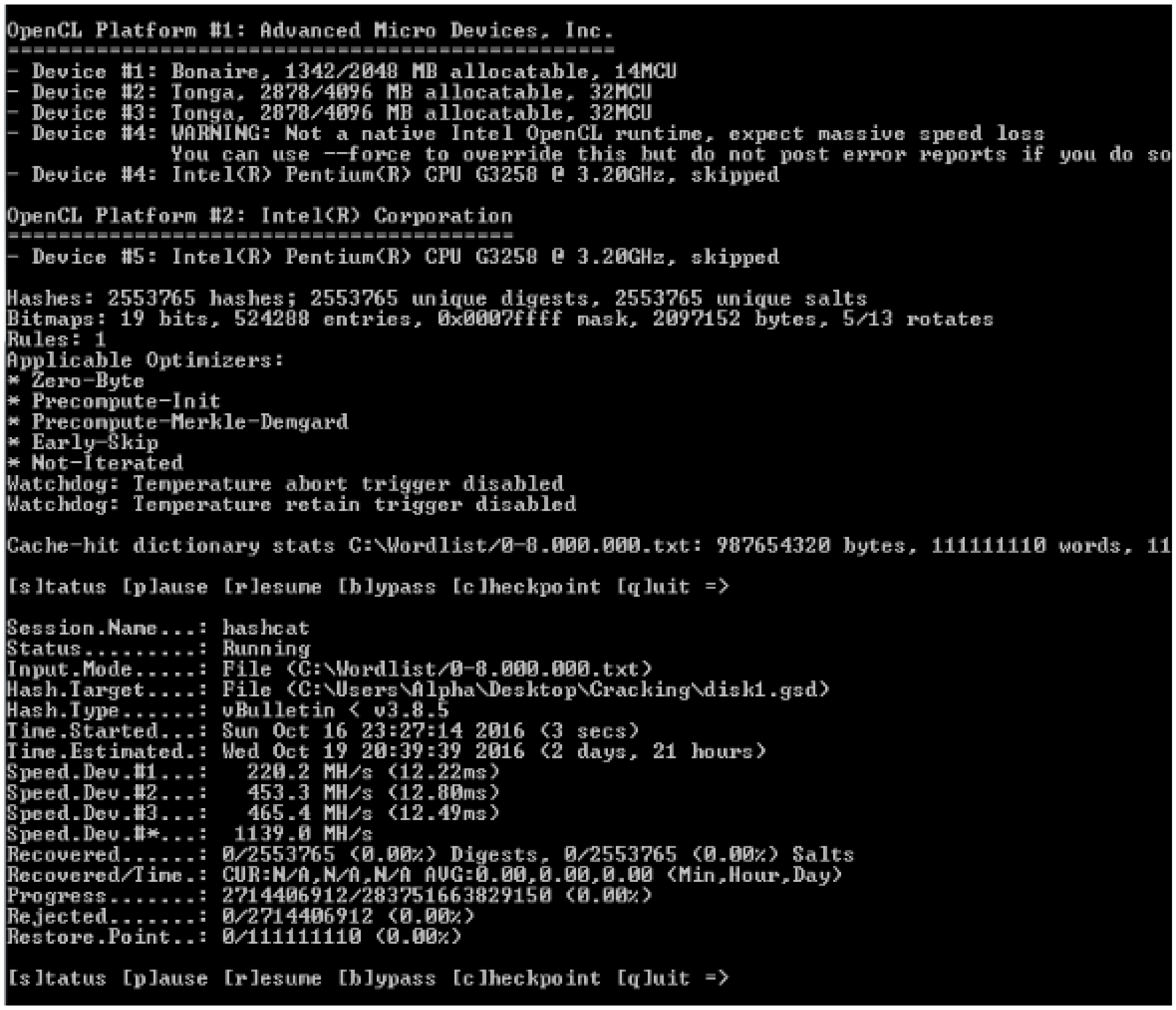
```
9e84089505ee83cda907acea6dd956a3:sadie897
a6f7411233b60454b64569fdb5dd3fd5:purple883
fe0c7eb420462ec6d22bc699b9097e61:arceus
6b18d55d3cffbbc8d3719a45dea29132:Trixie1982
2f2729f09cb4252ac6f5726664d945a3:skillet555
2e557da4c8312538245491005l9a3844:poison52
384b06d8e1e791c6ed1907382f7aa235:pixie123
334dc129d6a82430b220f7ff4da34e96:winner936
```

The time running you have a question to answer before you can use it, where you agree NOT to use this in malicious things (Write YES). The second is that all wordlist (that you have chosen to use) will be indexed / cached, and it's going to take a couple of minutes depending on size on wordlist. This only happens the first time you running with a new file

First time looking at the info Hashcat displays

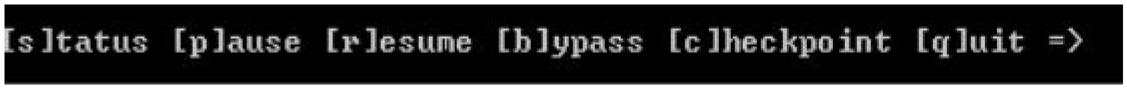
If you are an experienced user you can jump this chapter. This is for those who never used Hashcat before and don't know how it looks like. Using Hashcat for the very first time can be confusing. When looking at the picture below it seems very complicated but if you draw a line almost in the middle of the picture I'm going to explain it to you. We start with the upper part. Here are info about the hardware, which graphics cards finds for instance, and some info about the amount of hashes and the method that Hashcat uses to crack the hashes. The bottom part however is more interesting. Here we find that tell us how fast our graphics cards **Estimated"** and **"Progress"** that's always important to know right, and we must keep an eye to the **"Recovered"** that tells us how many hashes we cracked. is the current wordlist that were using and last the **"Hash Target"** that tells us which file with hashes (or single hash) using.

In the picture below there's in Windows 7 Command Prompt



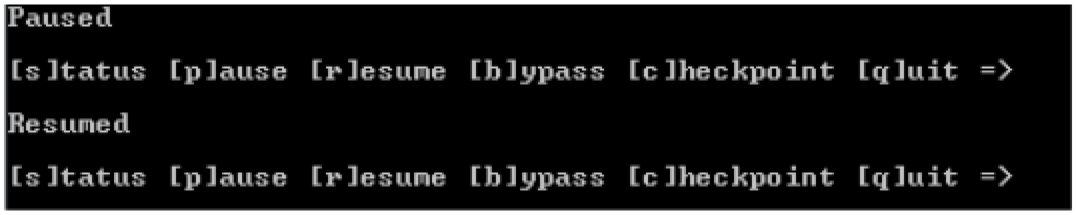
```
OpenCL Platform #1: Advanced Micro Devices, Inc.
================================================
- Device #1: Bonaire, 1342/2048 MB allocatable, 14MCU
- Device #2: Tonga, 2878/4096 MB allocatable, 32MCU
- Device #3: Tonga, 2878/4096 MB allocatable, 32MCU
- Device #4: WARNING: Not a native Intel OpenCL runtime, expect massive speed loss
             You can use --force to override this but do not post error reports if you do so
- Device #4: Intel(R) Pentium(R) CPU G3258 @ 3.20GHz, skipped

OpenCL Platform #2: Intel(R) Corporation
========================================
- Device #5: Intel(R) Pentium(R) CPU G3258 @ 3.20GHz, skipped

Hashes: 2553765 hashes; 2553765 unique digests, 2553765 unique salts
Bitmaps: 19 bits, 524288 entries, 0x0007ffff mask, 2097152 bytes, 5/13 rotates
Rules: 1
Applicable Optimizers:
* Zero-Byte
* Precompute-Init
* Precompute-Merkle-Demgard
* Early-Skip
* Not-Iterated
Watchdog: Temperature abort trigger disabled
Watchdog: Temperature retain trigger disabled

Cache-hit dictionary stats C:\Wordlist/0-8.000.000.txt: 987654320 bytes, 111111110 words, 11

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>

Session.Name...: hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist/0-8.000.000.txt)
Hash.Target....: File (C:\Users\Alpha\Desktop\Cracking\disk1.gsd)
Hash.Type......: vBulletin < v3.8.5
Time.Started...: Sun Oct 16 23:27:14 2016 (3 secs)
Time.Estimated.: Wed Oct 19 20:39:39 2016 (2 days, 21 hours)
Speed.Dev.#1...:   220.2 MH/s (12.22ms)
Speed.Dev.#2...:   453.3 MH/s (12.80ms)
Speed.Dev.#3...:   465.4 MH/s (12.49ms)
Speed.Dev.#*...:  1139.0 MH/s
Recovered......: 0/2553765 (0.00%) Digests, 0/2553765 (0.00%) Salts
Recovered/Time.: CUR:N/A,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 2714406912/283751663829150 (0.00%)
Rejected.......: 0/2714406912 (0.00%)
Restore.Point..: 0/111111110 (0.00%)

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

Now running we have a couple of keyboard buttons to use to do some basic functions as pause [p] , resume [r] to see the status [s] and quit [q], that's the most used. Checkpoint [c] makes a restore point to start from if I have to restart the server for some reason.



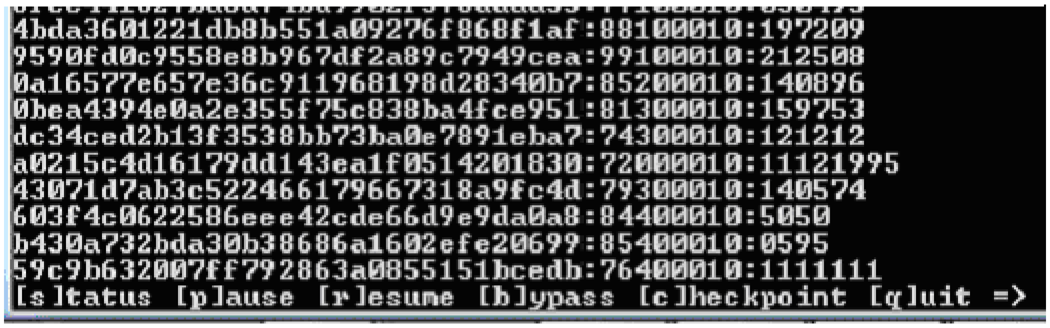
```
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

So if you want to quit or pause / resume or anything else, just use the keys on the keyboard. Also note that it's case sensitive, so watch out using caps lock. If using caps lock nothing will happening when you use the buttons



```
Paused
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
Resumed
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

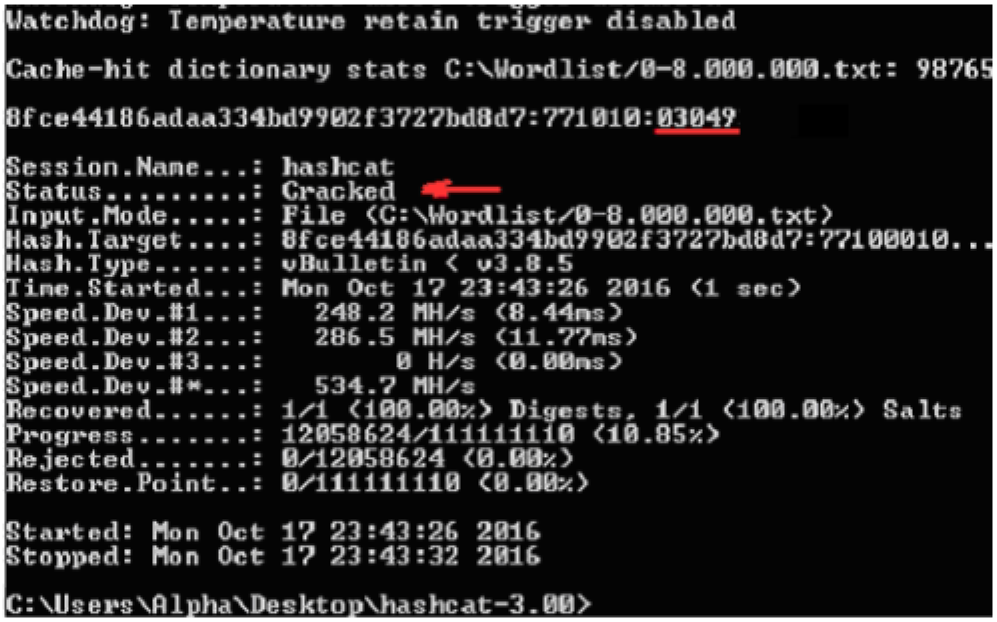
If Hashcat is cracking a text file with hashes it will display the cracked ones after each other like this (see below), the result you will find in the pot file (hashcat.pot) that you will find in the same folder as the main program. Remember that every hash you crack from now on will be added to that file. It's a good thing because if Hashcat stumble on the hash again in another file it won't add it to the pot file, it won't crack the same hash twice unless you remove the hash / password from the pot file. Also worth to mention is that it's possible to ask Hashcat to copy parts of cracked hashes to an output file of your choice, but some actions must be done first, more about that later.



```
4bda3601221db8b551a09276f868f1af:88100010:197209
9590fd0c9558e8b967df2a89c7949cea:99100010:212508
0a16577e657e36c911968198d28340b7:85200010:140896
0bea4394e0a2e355f75c838ba4fce951:81300010:159753
dc34ced2b13f3538bb73ba0e7891eba7:74300010:121212
a0215c4d16179dd143ea1f0514201830:72000010:11121995
43071d7ab3c52246617966 7318a9fc4d:79300010:140574
603f4c0622586eee42cde66d9e9da0a8:84400010:5050
b430a732bda30b38686a1602efe20699:85400010:0595
59c9b632007ff792863a0855151bcedb:76400010:1111111
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

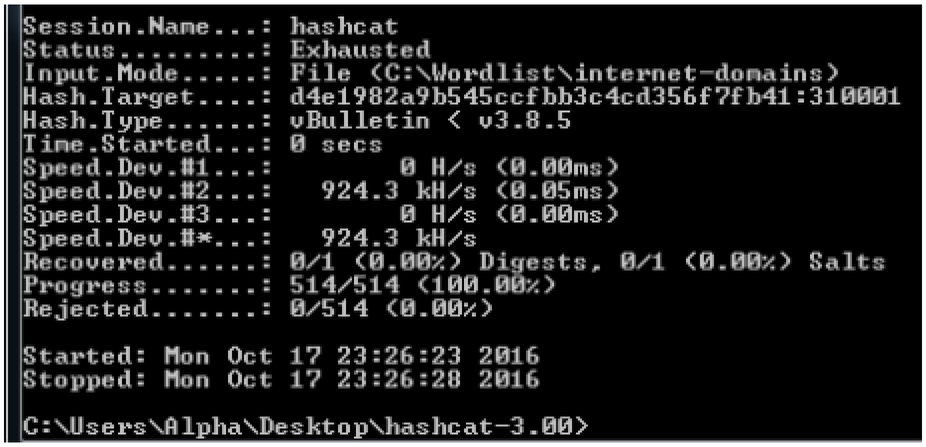
Now this kind of hash in the picture is called vBulletin, and you should read it like this **Hash: Salt: Password.** Now it's quite easy to understand I hope. More about salt and vBulletin

If cracks a single hash it will display the key like this



```
Watchdog: Temperature retain trigger disabled

Cache-hit dictionary stats C:\Wordlist/0-8.000.000.txt: 98765

8fce44186adaa334bd9902f3727bd8d7:771010:03049

Session.Name...: hashcat
Status.........: Cracked
Input.Mode.....: File (C:\Wordlist/0-8.000.000.txt)
Hash.Target....: 8fce44186adaa334bd9902f3727bd8d7:77100010...
Hash.Type......: vBulletin < v3.8.5
Time.Started...: Mon Oct 17 23:43:26 2016 (1 sec)
Speed.Dev.#1...:   248.2 MH/s (8.44ms)
Speed.Dev.#2...:   286.5 MH/s (11.77ms)
Speed.Dev.#3...:       0 H/s (0.00ms)
Speed.Dev.#*...:   534.7 MH/s
Recovered......: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.......: 12058624/111111110 (10.85%)
Rejected.......: 0/12058624 (0.00%)
Restore.Point..: 0/111111110 (0.00%)

Started: Mon Oct 17 23:43:26 2016
Stopped: Mon Oct 17 23:43:32 2016

C:\Users\Alpha\Desktop\hashcat-3.00>
```

If you look at the **"Status"** this will change from running to or depending on where in the running process Hashcat So cracking a text file with multiple hashes will give the same message running with one hash if not all hashes is found, and that is **"Exhausted"**



```
Session.Name...: hashcat
Status.........: Exhausted
Input.Mode.....: File (C:\Wordlist\internet-domains)
Hash.Target....: d4e1982a9b545ccfbb3c4cd356f7fb41:310001
Hash.Type......: vBulletin < v3.8.5
Time.Started...: 0 secs
Speed.Dev.#1...:        0 H/s (0.00ms)
Speed.Dev.#2...:   924.3 kH/s (0.05ms)
Speed.Dev.#3...:        0 H/s (0.00ms)
Speed.Dev.#*...:   924.3 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 514/514 (100.00%)
Rejected.......: 0/514 (0.00%)

Started: Mon Oct 17 23:26:23 2016
Stopped: Mon Oct 17 23:26:28 2016

C:\Users\Alpha\Desktop\hashcat-3.00>
```

Basic use

So how do we crack those hashes then, I will first cover some I intend to go through step by step here, so don't expect partially written commands to work, we will get fully defined commands later

hashcat64.exe -m [type]

-m [type] = Module type. Decide the type of hash you want to crack. In this book were going to talk about a fraction of them. You add a number after –m to specify what kind of hash you're interested to crack.

0 = MD5
10 = Salted MD5 ($pass$salt)
20 = Salted MD5 ($salt$pass)
100 = SHA1
110 = Salted SHA1 ($pass$salt)
120 = Salted SHA1 ($salt$pass)
400 = MD5(Wordpress), MD5(phpBB3), MD5(Joomla)
1400= SHA256
2500 = WPA/WPA2 cracking

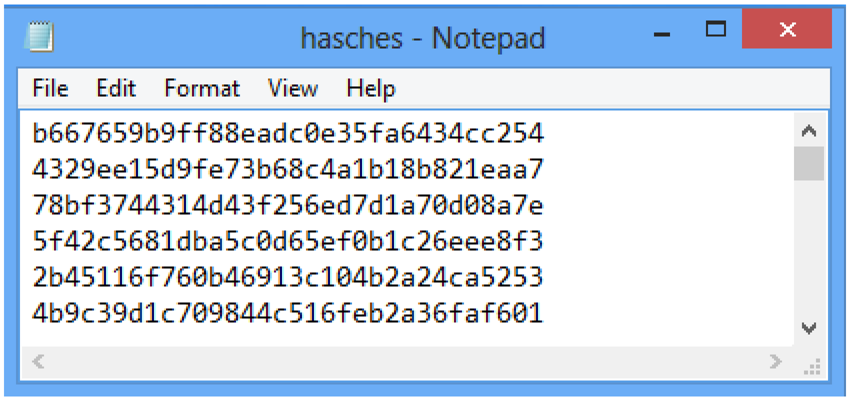
2600 = Double MD5
11300 = Bitcoin/Litecoin wallet.dat

So now we decided to run with MD5 (number 0 after -m). Next step is to tell Hashcat the actual hash or file

hashcat64.exe -m 0 or file]

[hash or file] = here you can choose either one hash or a text file with It will look something like this

hashcat64.exe -m 0
Or
hashcat64.exe -m 0 c:\hasches\hasches.txt



```
b667659b9ff88eadc0e35fa6434cc254
4329ee15d9fe73b68c4a1b18b821eaa7
78bf3744314d43f256ed7d1a70d08a7e
5f42c5681dba5c0d65ef0b1c26eee8f3
2b45116f760b46913c104b2a24ca5253
4b9c39d1c709844c516feb2a36faf601
```

Next is to verify what kind of attack you will be Running with wordlist is the same as attack mode 0, and we don't need to specify that, has the default setting at 0

hashcat64.exe -m 0 5a42e5aafdbaff0d65ef0b1c26eae8f3 -a 0
And
hashcat64.exe -m 0 5a42e5aafdbaff0d65ef0b1c26eae8f3 work just as good

As shown above, attack mode is specified with number behind -a

Attack modes:
0 = Straight
1 = Combination
3 = Brute-force
6 = Hybrid + mask
7 = Hybrid mask +

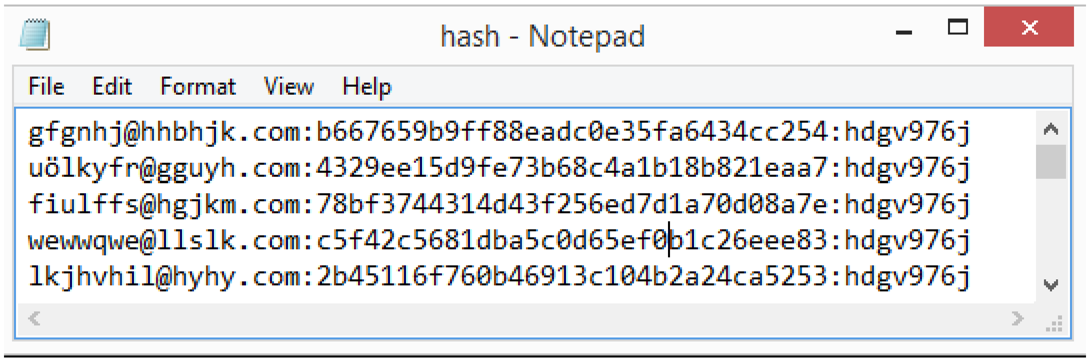
Next is to locate a password file, or a drive or a folder containing many wordlists, **here are three working examples**

hashcat64.exe -m 0 5a42e5aafdbaff0d65ef0b1c26eae8f3 d:\wordlist

hashcat64.exe -m 0 5a42e5aafdbaffod65efob1c26eae8f3 word1.txt
hashcat64.exe -m 0 c:\hasches\hasches.txt wordlist1 wordlist2 wordlist3

<u>Databases and cracking hashes</u>

Sometimes when we get our hands on a database we can't just crack it in a usual way. As you see in the picture usernames passwords and are in the same database. How do we crack this without removing anything, and get username (emails) and password? We must extract cracked passwords from the hashcat.pot file and put the usernames with the right password. Let me explain



```
gfgnhj@hhbhjk.com:b667659b9ff88eadc0e35fa6434cc254:hdgv976j
uölkyfr@gguyh.com:4329ee15d9fe73b68c4a1b18b821eaa7:hdgv976j
fiulffs@hgjkm.com:78bf3744314d43f256ed7d1a70d08a7e:hdgv976j
wewwqwe@llslk.com:c5f42c5681dba5c0d65ef0b1c26eee83:hdgv976j
lkjhvhil@hyhy.com:2b45116f760b46913c104b2a24ca5253:hdgv976j
```

Note! As you see above, usernames hashes and salts are separated by a ":" but that's not always the case. Other punctuation marks (separators) exist as these two as an example ; and a ,

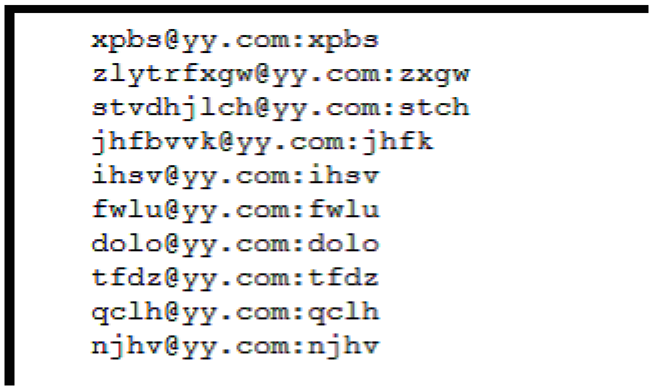
So let's start this

hashcat64.exe -m 10 c:\hashes.txt c:\wordlists\

So running of cause (-m 10), and tell to ignore one of the fields as “usernames” (emails) with the use of --username, and we point it to our wordlist Let it run through all your wordlists. Now we need to fuse the right password with the right username by combining our pot file with the original database with (usernames)

hashcat64.exe -m 10 --username --show -o c:\hash\Complete.txt --outfile-format 2 c:\hashes.txt

When we have done all of this we will have a file called and will create this file for you who contains usernames and passwords



```
xpbs@yy.com:xpbs
zlytrfxgw@yy.com:zxgw
stvdhjlch@yy.com:stch
jhfbvvk@yy.com:jhfk
ihsv@yy.com:ihsv
fwlu@yy.com:fwlu
dolo@yy.com:dolo
tfdz@yy.com:tfdz
qclh@yy.com:qclh
njhv@yy.com:njhv
```

Load

Now I will talk about the load. We can change the workload to press a couple of thousands more hashes per second out of the graphics card, it has its advantages and will give 5-10% extra raw power, however be aware that the graphics card develop more heat so keep an eye at your

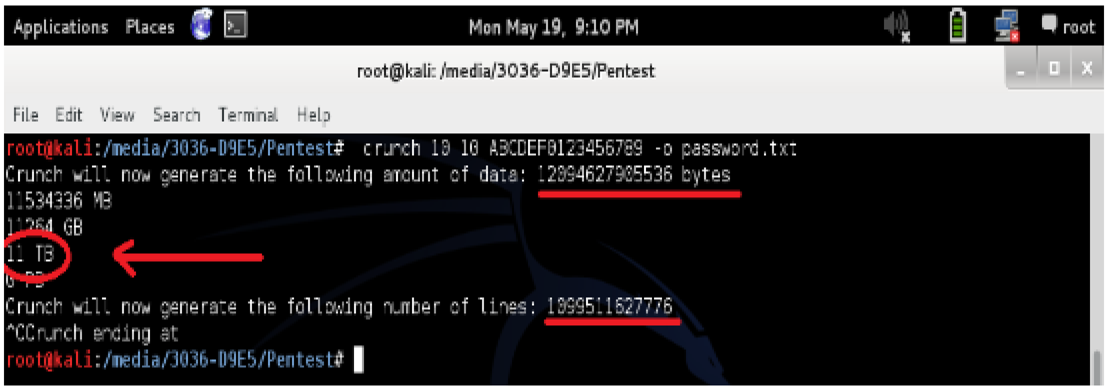
To use extra load add **-w [number]** at the end of the command like **-w 3** its using default settings (nr Old hashcat is only graded 1-3 but it's basically the same

Problem with heat

Some older GPUs have a tendency to report a faulty temperature which means that hashcat will exit after a it's an inbuilt safety mechanism in Either you can try modify the threshold for aborting current job by adding **--gpu-temp-abort [number of degrees Celsius]** in the command line or simply **--gpu-temp-disable** but the later one this is not recommended in

Mask attack / Hybrid attack /Mixed attack /Combination

So let's have a talk about the mask attack. What is it and how does it work? When attacking a hash that you know from the that it's a kind of key that's seems to be a random organized pattern with digits or letters or both. Then it's a good idea to use the mask attack. The best part is that it doesn't take more than a couple of kilobytes in the hard drive in worst case, to compare with if we decided to do a wordlist in crunch in Linux it would be many terabyte big as you can see in the picture. A Mask attack here would be a great way to save space. Also avoid using crunch as much as and if you use it, be aware that the size of a wordlist can fill your disk if you're not



```
root@kali:/media/3036-D9E5/Pentest#  crunch 10 10 ABCDEF0123456789 -o password.txt
Crunch will now generate the following amount of data: 12094627905536 bytes
11534336 MB
11264 GB
11 TB
0 PB
Crunch will now generate the following number of lines: 1099511627776
^CCrunch ending at
root@kali:/media/3036-D9E5/Pentest# 
```

So it's only when 100% certain that a key might have these type of pattern we can use this type of attack, you could do this to save space on our hard

going to look at the "built in character set" that we use in when using a mask attack. **<u>In this chapter I'm going to use WPA/WPA2 cracking as an example on all attacks.</u>** WPA cracking we will study later. We can use the mask attack when cracking WPA/WPA2, an example is shown

hashcat64.exe -m 2500 test-01.hccap -a 3 ?d?l?u?d?d?d?u?d?s?a

Now going to take a look at the mask attack and what this really means. First down the mask in pairs so it's more easy to follow, then were going to translate what they really means

?d ?l ?u ?d ?d ?d ?u ?d ?s ?a = 10 letters and digits long WPA key

```
^    ^   ^    ^    ^   ^     ^    ^   ^   ^
```

This above is the "mask", every pair represent a number or a letter or symbol, and it tells to try every combinations number / letters / ASCII and so on. As an example "?d" means that it will try the whole numeric charset (0-9) on that specific place and in the key.

Built-in charsets characters in hashcat

?l = abcdefghijklmnopqrstuvwxyz

?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ

?d = 0123456789

?s = «space»!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

?a = ?l?u?d?s

In new hashcat **?H** and **?h** will also be available. Uppercase ASCII HEX = ?H and lowercase ASCII HEX = h? Which means

?H = ABCDEF0123456789

?h = abcdef0123456789

Here are a couple of examples how a key may look like

Key=

0aC575G2/@

9zG432H0*K

EsA111W1$4

3wD001Q5+z

So if we down the command separate parts look like this.

hashcat64.exe -m 2500 test-01.hccap -a 3 ?d?l?u?d?d?d?u?d?s?a

hashcat64.exe = the program we run. In the same folder there's a hashcat64.exe for 64 bit OS and hashcat64.bin / hashcat64.bin for Linux
-m 2500 = the specific hash 2500 means WPA/WPA2
test-01.hccap = the converted *.cap file coming to that further down)
-a 3 = Attack mode, custom-character set (mask attack)
?d?l?u?d?d?d?u?d?s?a = The mask

If specific parts of the key are known, for example first, fifth and last character, you can include them in the mask on first, fifth and last place like shown below (for this example, the letter "Y" is known)

hashcat64.exe -m 2500 test-01.hccap -a 3 Y?d?d?dY?d?d?d?dY

And it will try every possibility with the letter "Y" present on place 1, 5 and 10 in the key

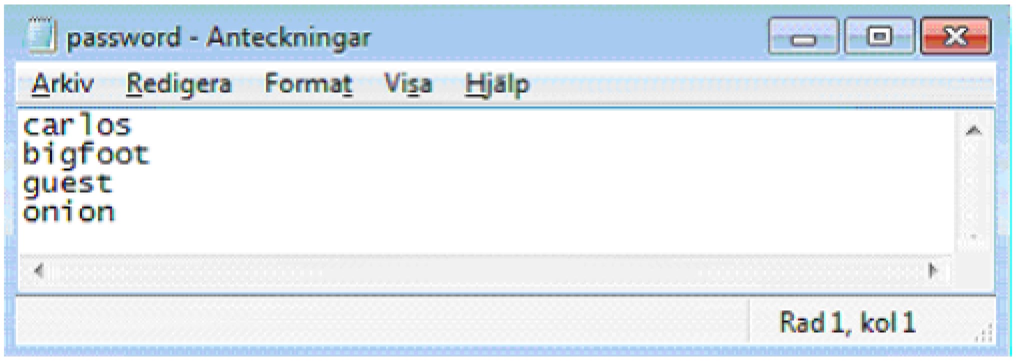
<u>Hybrid attack</u>

Also worth to mention is the hybrid attack. The hybrid attack combines a brute force wordlist with a mask attack, and can sometimes be useful. In hybrid attack what we actually do is we pass any specific string to manually, but automate it by passing a wordlist to Hashcat. Hashcat picks up words one by one and test them to the every password possible by the Mask defined.

hashcat64.exe -m 2500 test-01.hccap -a 6 password.txt ?d?l?d?l

-a 6 = the hybrid attack
password.txt = wordlist
?d?l?d?l = a mask (4 letters and numbers)

This wordlist in this example is a text file containing 4 random
carlos
bigfoot
guest
onion



```
carlos
bigfoot
guest
onion
```

Now it will use those words and combine it with the mask. When running the result will be like

carlos2e1c

bigfootoh1d

guest5p4a

onion1h1h

The fun part is that if you reverse the order, like this **-m 2500 test-01.hccap -a 7 ?d?l?d?l password.txt** the result will be like this

7a2ecarlos

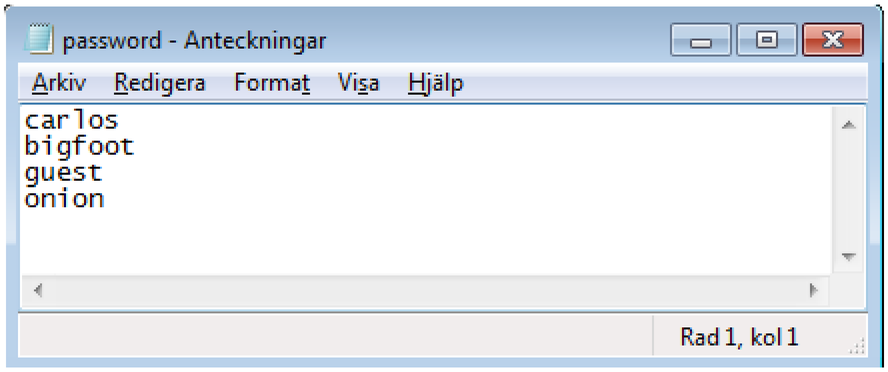
8j3abigfoot

0t3wguest

6a5jonion

<u>Combination attack.</u>

This attack combines 2 wordlist or one mixing words from one wordlist (by adding it again in command If we look at our old password it will go like this



```
carlos
bigfoot
guest
onion
```

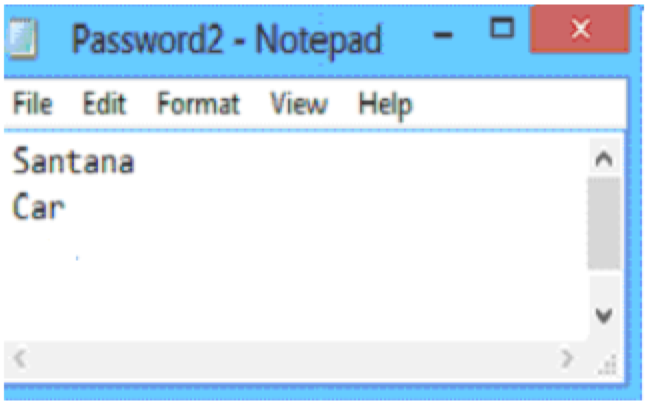
hashcat64.exe -m 2500 test-01.hccap -a 1 password.txt password.txt

The output will be

```
carloscarlos
carlosbigfoot
carlosguest
carlosonion
bigfootcarlos
bigfootbigfoot
bigfootguest

bigfootonion
```

And so

If we create 2 word list like this, the result would be Note this attack we are limited to 2 wordlists. This only works with 2 wordlists



```
Santana
Car
```

hashcat64.exe -m 2500 test-01.hccap -a 1 password.txt Password2

The output will be

<u>Specify charset</u>

Before we go in and talk about let's discover something How would you do the approach to crack a hash that you know is 8 keys long A-F and 0-9? Building this with crunch is relative simple but it would take 36-37GB on disk, so to
avoid this waste of space we must use the mask in a smarter way

hashcat64.exe -m 2500 test-01.hccap -a 3 -1 ABCDEF0123456789 ?1?1?1?1?1?1?1?1

hashcat64.exe = The program,

-m 2500 = The specific hash 2500 means WPA/WPA2
test-01.hccap = The converted *.cap file (were coming to that further down)
-a 3 = Attack mode, custom-character set (mask attack)
-1 = (a one) Specify charset
ABCDEF0123456789 = charset
?1?1?1?1?1?1?1?1 = the specified charset and the length of it

Instead as above, you see that we have a small bit of the alphabet, we could use multiple character set on the same place in the mask, like this

hashcat64.exe -m 2500 test-01.hccap -a 3 -1 ?s?a?u ?d?d?1?1?1?d?d?1

hashcat64.exe = The program,
-m 2500 = The specific hash 2500 means WPA/WPA2
test-01.hccap = The converted *.cap file (were coming to that further down)
-a 3 = Attack mode, custom-character set (mask attack)
-1 = (a one) Specify character set
a number between 0 to 9
chosen character set keys
d?d?1?1?1?d?d?1 = the specified character set and the length of it

Also working is

hashcat64.exe -m 2500 test-01.hccap -i -1 s?a?u ?d?d?1?1?1?d?d?1
-i = Increment

Difference is that we don't specify an attack mode. Giving an attack mode is always

<u>Mask attack in a hcmask file</u>
There is a second more powerful mask attack and it is to create a file in notepad and it as "my.hcmask" inside the folder, and inside the create your own kind of policies. Save the file inside the folder.

Let's say that you want to run a mask attack "Total 10 numbers and letters in a row" but to decimate cracking time you want to add a certain pattern to avoid some patterns. How do we solve this below?

"Total 10 random numbers and letters in a row, random A-F

Numbers from 7 numbers to 9

There can never be three letters after each other"

We create a text file that we name **my.hcmask**

In that file we can create our own rules / policies if we want to.
If we use this file we got additional 4 we can mix with plus those we already known since before. [?4,]

We could use it like this

ABC,?1?d?d?d?d?d?d?d?d?d ?1=ABC ?d =0-9
ABC,klm,?1?d?2?d?d?d?d?d?d?d ?1= ABC ?2=klm ?d=0-9
ABC,klm,XYZ,?1?d?2?d?3?d?d?d?d?d ?1= ABC ?2=klm ?
3=XYZ? d=0-9

You want more letters just add them until your satisfied perhaps like this
ABCDEFGHIJ,klmnop,XYZ,?1?d?2?d?3?d?d?d?d?d

It's also possible to mix with those characters we already known, as example
ABC,klm,XYZ,?1?d?2?d?3?d?u?s?a?d

Okay so back to the “my.hcmask” file that we have open in notepad. With this rules above we could just do it like this (I won’t go through the whole but you get the idea)

```
# 1 upper

ABCDEF,?1?d?d?d?d?d?d?d?d?d
ABCDEF,?d?1?d?d?d?d?d?d?d?d
ABCDEF,?d?d?1?d?d?d?d?d?d?d
ABCDEF,?d?d?d?1?d?d?d?d?d?d
ABCDEF,?d?d?d?d?1?d?d?d?d?d
ABCDEF,?d?d?d?d?d?1?d?d?d?d
ABCDEF,?d?d?d?d?d?d?1?d?d?d
ABCDEF,?d?d?d?d?d?d?d?1?d?d
ABCDEF,?d?d?d?d?d?d?d?d?1?d
ABCDEF,?d?d?d?d?d?d?d?d?d?1
# 2 upper next
ABCDEF,?1?1?d?d?d?d?d?d?d?d
ABCDEF,?d?1?1?d?d?d?d?d?d?d
ABCDEF,?d?d?1?1?d?d?d?d?d?d
ABCDEF,?d?d?d?1?1?d?d?d?d?d
ABCDEF,?d?d?d?d?1?1?d?d?d?d
ABCDEF,?d?d?d?d?d?1?1?d?d?d
ABCDEF,?d?d?d?d?d?d?1?1?d?d
ABCDEF,?d?d?d?d?d?d?d?1?1?d
ABCDEF,?d?d?d?d?d?d?d?d?1?1
```

And so on!

Now did you see that? l is replaced with number one. ABCDEF = ?1 and ?d is a random number from 0-9, now going through this file will take a lot of time, however you have the ability to think through what keys you want to cover. We could modify it like this as well, it doesn’t matter as long as you mark the difference with a comma in between those letters to use and the mask

```
# Random letters

QWERTY,?1?d?1?d?d?d?d?d?d?d
SPELA?1?d?d?1?d?d?d?d?d?d
QRTY,?1?d?d?d?1?d?d?d?d?d
AaBbCc,?1?d?d?d?d?1?d?d?d?d
AaBbCc,?1?d?d?d?d?d?1?d?d?d
abcdefghijklmnopqrstuvxyz,?1?d?d?d?d?d?d?1?d?d
abcDefghijKlmnoPqrstuvxyz,?1?d?d?d?d?d?d?d?1?d
```

You can also use masks

```
# With masks
?l?u?s,?d?1?d?1?d?d?d?d?d?d
```

```
?l?u?s,?d?1?d?d?1?d?d?d?d
?l?u?s,?d?1?d?d?d?1?d?d?d?d
?a,?d?1?d?d?d?d?1?d?d?d
?a,?d?1?d?d?d?d?d?1?d?d
?a,?d?1?d?d?d?d?d?d?1?d
?s,?d?1?d?d?d?d?1?d?d?d
?d,?d?1?d?d?d?d?1?d?d?d
```

This is a very powerful attack. Now to run this, simply type

```
hashcat64.exe -m 2500 test-01.hccap -a 3 my.hcmask
```

And it will go through the file one mask at a time. It won't pause between and run until its

SALT, whaaaat?

Salt, what is that? In cryptography, “a salt” is random data that is used as an additional input to a one-way function that hashes a password. The primary function of salts is to defend against dictionary attacks and against pre-computed rainbow table attacks. With this going to be a little harder to crack that specific hash, but honestly it depends on what type of hash it is. Modern graphics card with a lot of fast graphics memory will crack a few hundred thousand salted hashes at the same time, or more, without problems, depending on hash type and length of salt. However salted hashes will slow down the cracking time, and depending on the length on the salt, it can slow normal cracking speed down to half the speed. Now normally this isn’t a problem unless were trying to crack a hash that’s already hard to crack, lets say a Sha512 with a long salt, or a very huge DB with millions of rows. However at some point it going to take too long for to crack it unless you stumble on it with pure luck. And that’s what it’s all about.

If a Database with hashes + salt that takes “X” amount of days to crack, and the energy cost exceeds the expected, our mighty cracker might turn his head on other hashes that's simpler to crack

How do I know if my hash is salted?
The hash seems longer than normal and is separated by a (that's not always the case. Other punctuation marks /separators exist as these two as example and a and

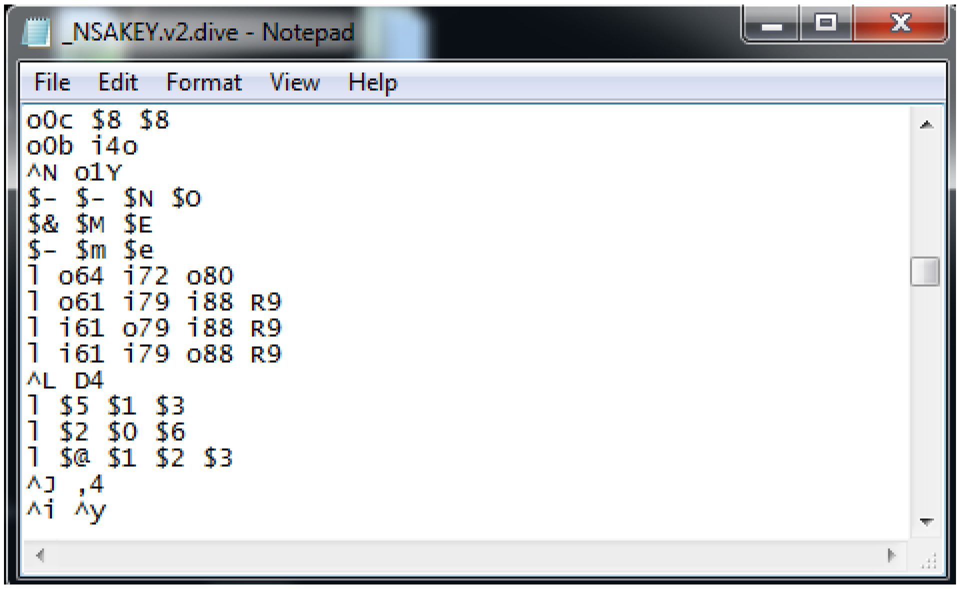
5f42c5681dba5c0d65ef0b1c26eee8f3 = a normal MD5 hash

0fda58630310a6dd91a7d8f0a4ceda2: f42253742 = a MD5 hash with a salt ($salt$pass)
= a MD5 hash with a salt ($pass$salt)

Above here you see some easy examples that I made up. The fun part is that Hashcat is made in a way that makes ($pass$salt) faster to crack then ($salt$pass) between 8-10% faster actually, depending on the hash type.

Rules

I'm surprised that more people aren't using rules nowadays. It's almost as effective as running with masks. To run with rules you need a basic wordlist of any type. Normally when running with wordlist it's pretty slow and you're only using a fraction of the capability it has. You could resemble it like you taking one ball at a time throwing it at a basket, and you wait until the ball returns to you. Now a list of rules really challenges the graphics To compare "You get a bunch of balls (pun intended) and throw a shit ton of balls at it at the same time and the balls return in a faster speed to you", and that's exactly what the rules does. It takes the word, and looks at the list with rules, and goes through the list. **"Regular expressions are too slow. Because we have to generate more than a billion fresh password candidates in less than 10ms before hashing algorithms start to become idle, and then again and again, second after second. Just take a look at your GPU speed display to get an idea of it"** as they say in the Hashcat



```
o0c $8 $8
o0b i4o
^N o1Y
$- $- $N $O
$& $M $E
$- $m $e
l o64 i72 o80
l o61 i79 i88 R9
l i61 o79 i88 R9
l i61 i79 o88 R9
^L D4
l $5 $1 $3
l $2 $0 $6
l $@ $1 $2 $3
^J ,4
^i ^y
```

So going to use that time, which in the computer world is an eternity, to run it against modified words of the original word in the wordlist. To create rules you have to "program" want you want to do with the word, before it compares it with the There is a kind of programming language to perform this. It's easy, but I won't cover it that much in this book, just the basics, more about that later. going to use already created rules that comes along with located here Just to get you an idea what it actually do. The number of rules you would use are "as many as you want", however it's limited to 14 functions. So let's see what a word can do with a rule list

A word from our wordlist

cornflakes

OUTPUT EXAMPLES

cornfl@kes

CoRnFlAkes

flakescorn

nC@ksfeLoR

CORNFLAKES

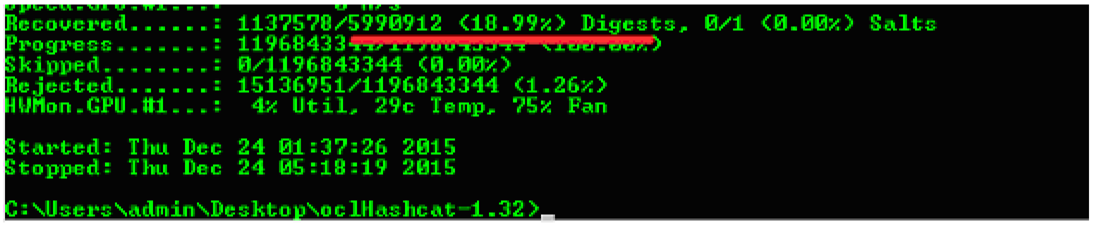
Cornflakes123

co®nfl@ke$

cornfl4k35

AND MANY MORE

Now how is this? We will try a wordlist with rules and one without rules, and then compare the result. We will run the same wordlist. The first test out there is without any rules at all and the wordlist is a 15GB file called Crackstation.txt. Number of hashes is 5,9 million, and the type is MD5. As you see below running without the rules, you find around 18,9% or 1.13 million passwords was with an earlier version of



```
Recovered......: 1137578/5990912 (18.99%) Digests, 0/1 (0.00%) Salts
Progress.......: 11968433[redacted])
Skipped........: 0/1196843344 (0.00%)
Rejected.......: 15136951/1196843344 (1.26%)
HWMon.GPU.#1...:  4% Util, 29c Temp, 75% Fan

Started: Thu Dec 24 01:37:26 2015
Stopped: Thu Dec 24 05:18:19 2015

C:\Users\admin\Desktop\oclHashcat-1.32>_
```

In the next try I'm going to use one of the rules located here **\hashcat-x.xx\rules. To run with rules you simply add this to**

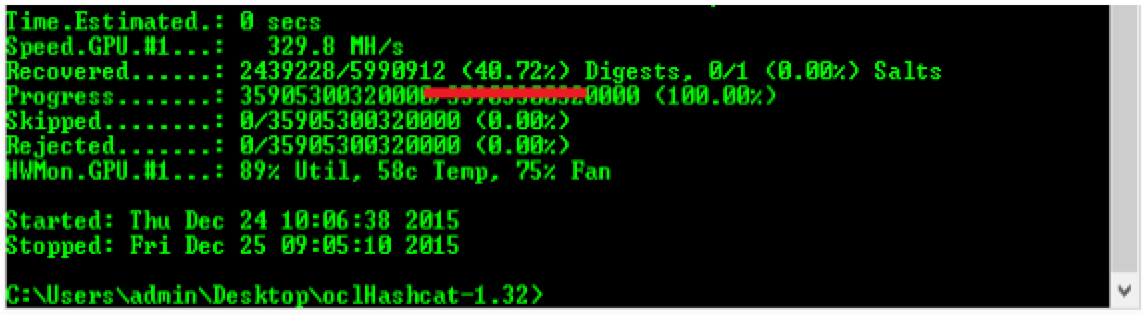
the existing command line, depending on rule

"-r c:\hashcat-3.00\rules\rockyou-3000.rule"

Also note that the rules should be added before the wordlist

Below is an example.

hashcat64.exe -m 0 hash.txt -r c:\hashcat-2.01\rules\rockyou-3000.rule e:\wordlists\pass.txt



```
Time.Estimated.: 0 secs
Speed.GPU.#1...:   329.8 MH/s
Recovered......: 2439228/5990912 (40.72%) Digests, 0/1 (0.00%) Salts
Progress.......: 35905300320000/35905300320000 (100.00%)
Skipped........: 0/35905300320000 (0.00%)
Rejected.......: 0/35905300320000 (0.00%)
HWMon.GPU.#1...: 89% Util, 58c Temp, 75% Fan

Started: Thu Dec 24 10:06:38 2015
Stopped: Fri Dec 25 09:05:10 2015

C:\Users\admin\Desktop\oclHashcat-1.32>
```

The time here is not important, what's more important is the number of passwords cracked which went from 18.9% to 40.7% or 2.43 million. So it's recommended to run with rules if you run with wordlists.

When it comes to the actual "coding" the rules, I will show you some. There are many rules and I won't go through them all here in this book, but I will show you some of the most basic ones. The only thing you need is a pure text file that we save as an unformatted text file. "Notepad" in windows is a good example. When you save the file, save it as a "[filename].rule" and done. Let's take a look at some simple rules that we can use to write our own rule list.

For this to work I will invent a word from a wordlist, and then I'm going to show you how rules can modify the actual word. So what then you may ask. First the looks at the word and compare it to the rule Then it take line by line in the rule file and change that word and compare it to the hashes, and that under a specific time when the GPU have a waiting time it will compare all "ruled words" to the hash So one word will be modified and tested a number of times under the same time

The actual word to start with, (as an example)

<u>gr3Anat</u>

: = This means leave the word unchanged = gr3Anat

l = this changes all letters in word to lowercase = gr3anat

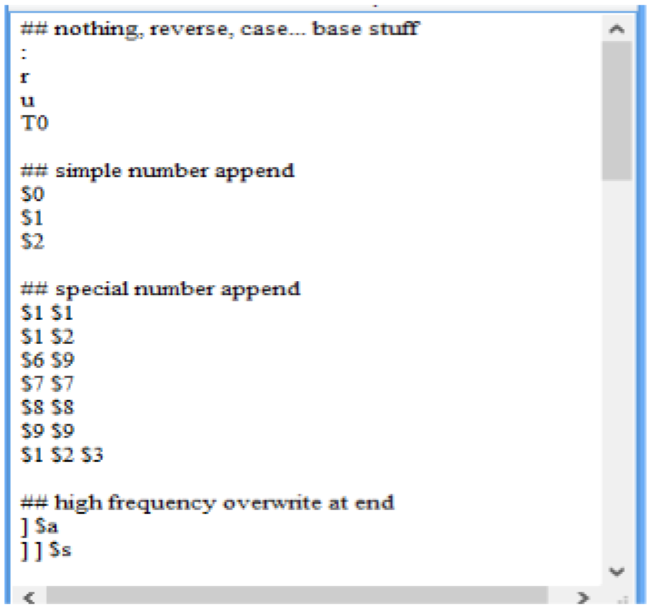
u = this changes all letters in word to uppercase = GR3ANAT

c = Capitalize the first letter in the word and use lowercase in the rest = Gr3anat
C = Lowercase first found letter, uppercase the rest = gR3ANAT
c $0 $A = capitalize the first letter and append a 0A to the end = Gr3Anat0A
t = Toggle the case of all characters in word. = GR3aNAT

r = Reverse the entire word = tanA3rg
d = Duplicate the word = gr3Anatgr3Anat
p2 = Append duplicated word number of times = gr3Anatgr3Anatgr3Anat
[= Deletes first character in word = r3Anat
([[[= Deletes the first three characters of the word = Anat)
] = Deletes last character in word = gr3Ana
(]]]] = Deletes the last four characters of the word = gr3)
D3 = Deletes character at a specific position (just change number) = grAnat
i3@ = Inserts character of choice at position 3 in word, In this case =
k = Swaps first two characters = rg3Anat
K = Swaps last two characters = gr3Anta
T2 = Toggle the case of characters at position 2 in this case = gR3Anat
$1 = Append extra character to end = gr3Anat1
($2 $0 $1 $6 for instance appends 2016 to the end of each word, like gr3Anat2016)

^1 = Prepend extra character to front = 1gr3Anat
'5 = Deletes any characters beyond the fifth character position = gr3An
sa@ = swap all occurrences of the character "a" for "@" = gr3An@t

So let's have a look inside of those existing rule file.



```
## nothing, reverse, case... base stuff
:
r
u
T0

## simple number append
$0
$1
$2

## special number append
$1 $1
$1 $2
$6 $9
$7 $7
$8 $8
$9 $9
$1 $2 $3

## high frequency overwrite at end
] $a
] ] $s
```

here you see that the creator has been nice to add comments to it with double" ##" You can follow most of the "code" written here

Maskprocessor

This is a tool that I think should be included in the download, but for some reason However it's linked from the page. Look for "TOOLS" in the side bar This little "add-on" to is very interesting. Maskprocessor is a High-Performance word generator with a per-position configurable charset packed into a single stand-alone binary. The program is available for Linux and Windows on both 32 bit and 64 bit. With this tool you can create rules and play around with Creating rules by hand works, but it could be very boring and time-consuming job, that's why I'm using this

Download and copy to Hashcat folder (windows)
https://github.com/jsteube/maskprocessor/releases

Basic usage:

mp64.exe + >

> does the same thing as –o yeah I'm lazy :)
Or just

mp64.exe
(That will display the result on screen)

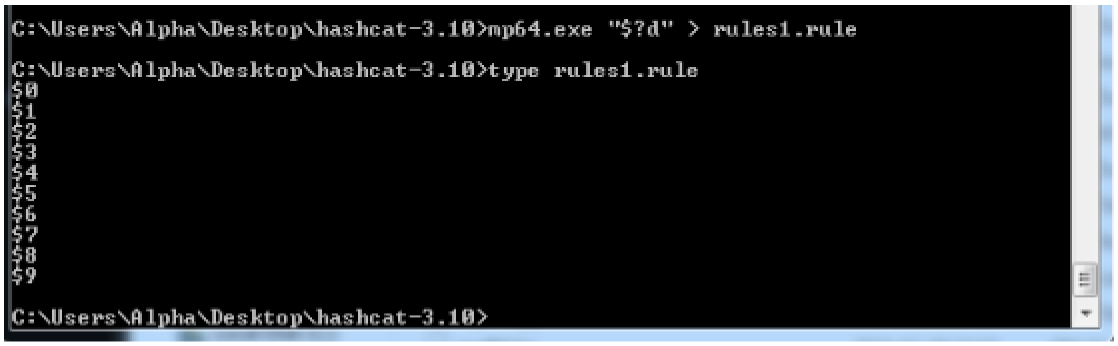
?d?d?d?d?d?d?d?d | Hashcat64.exe -m 2500 test.hccap

This is seldom seen nowadays, because can do it directly from but I wanted to show you that it's possible to pipe it with

And finally to specify and generate common rules will get a perfect rule file for just like you want it, like this for instance

mp64.exe -1 0123456789 "^?1^?1^?1" -o Prefix-0-999.rule

Below are a couple of pictures with example

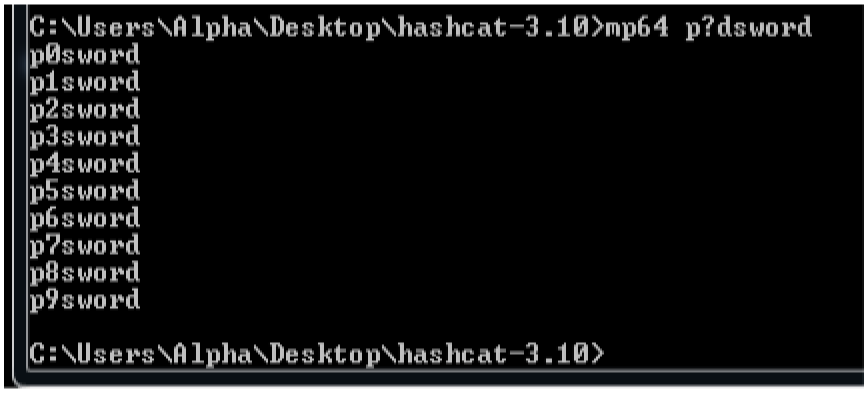


```
C:\Users\Alpha\Desktop\hashcat-3.10>mp64.exe "$?d" > rules1.rule

C:\Users\Alpha\Desktop\hashcat-3.10>type rules1.rule
$0
$1
$2
$3
$4
$5
$6
$7
$8
$9

C:\Users\Alpha\Desktop\hashcat-3.10>
```

It's also possible to use a word combined with a mask to see the result like this one for example

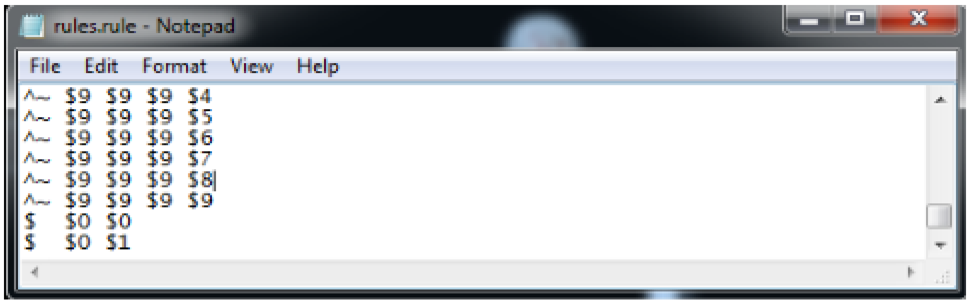
Word = **p?dssword**

And here is the result that we just display on the screen



```
C:\Users\Alpha\Desktop\hashcat-3.10>mp64 p?dsword
p0sword
p1sword
p2sword
p3sword
p4sword
p5sword
p6sword
p7sword
p8sword
p9sword

C:\Users\Alpha\Desktop\hashcat-3.10>
```

Below I have my own made rule file which isn't that advanced to be but again it's something that will make it easier for me to crack my hashes



```
^~ $9 $9 $9 $4
^~ $9 $9 $9 $5
^~ $9 $9 $9 $6
^~ $9 $9 $9 $7
^~ $9 $9 $9 $8
^~ $9 $9 $9 $9
$  $0 $0
$  $0 $1
```

Password Analysis and Cracking Kit

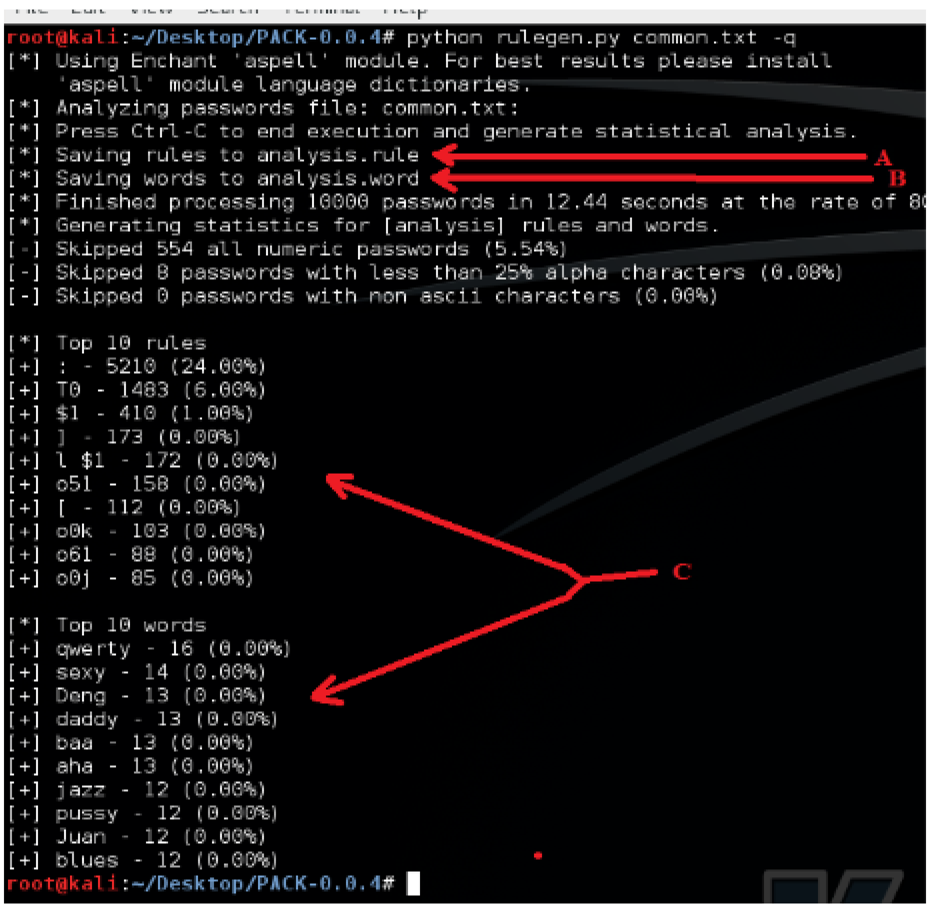
I thought to about mention this as well. or Password Analysis and Cracking Kit are a packed file containing a couple of files which contains multiple tools to create masks, rules policies and statistics. Everyone should at least know about before in to more advance This awesome kit needs Python to work so install a VM-machine with kali. You can run it in windows as well with "Python for windows" but its buggy and don't work at 100% sadly. I will let the author explain more about this kit

"PACK (Password Analysis and Cracking Toolkit) is a collection of utilities developed to aid in analysis of password lists in order to enhance password cracking through pattern detection of masks, rules, character-sets and other password characteristics. The toolkit generates valid input files for Hashcat family of password crackers.

NOTE: The toolkit itself is not able to crack passwords, but instead designed to make operation of password crackers more efficient"

I will referee you to the site or the manual and I will only show you 2 things that I think is super awesome with this. The Idea with this "pack of tool" is for example to create rules from a wordlist, or masks, policies see statistics or view statistics. I'm using this for inspiration when creating rules

Download it from here
http://thesprawl.org/projects/pack/.



```
root@kali:~/Desktop/PACK-0.0.4# python rulegen.py common.txt -q
[*] Using Enchant 'aspell' module. For best results please install
  'aspell' module language dictionaries.
[*] Analyzing passwords file: common.txt:
[*] Press Ctrl-C to end execution and generate statistical analysis.
[*] Saving rules to analysis.rule
[*] Saving words to analysis.word
[*] Finished processing 10000 passwords in 12.44 seconds at the rate of 80
[*] Generating statistics for [analysis] rules and words.
[-] Skipped 554 all numeric passwords (5.54%)
[-] Skipped 8 passwords with less than 25% alpha characters (0.08%)
[-] Skipped 0 passwords with non ascii characters (0.00%)

[*] Top 10 rules
[+] : - 5210 (24.00%)
[+] T0 - 1483 (6.00%)
[+] $1 - 410 (1.00%)
[+] ] - 173 (0.00%)
[+] l $1 - 172 (0.00%)
[+] o51 - 158 (0.00%)
[+] [ - 112 (0.00%)
[+] o0k - 103 (0.00%)
[+] o61 - 88 (0.00%)
[+] o0j - 85 (0.00%)

[*] Top 10 words
[+] qwerty - 16 (0.00%)
[+] sexy - 14 (0.00%)
[+] Deng - 13 (0.00%)
[+] daddy - 13 (0.00%)
[+] baa - 13 (0.00%)
[+] aha - 13 (0.00%)
[+] jazz - 12 (0.00%)
[+] pussy - 12 (0.00%)
[+] Juan - 12 (0.00%)
[+] blues - 12 (0.00%)
root@kali:~/Desktop/PACK-0.0.4#
```

The first is called **rulegen.py** and it goes through a normal wordlist and creates common known rules that all the words inside have in common, also it will create a word file with statistical data

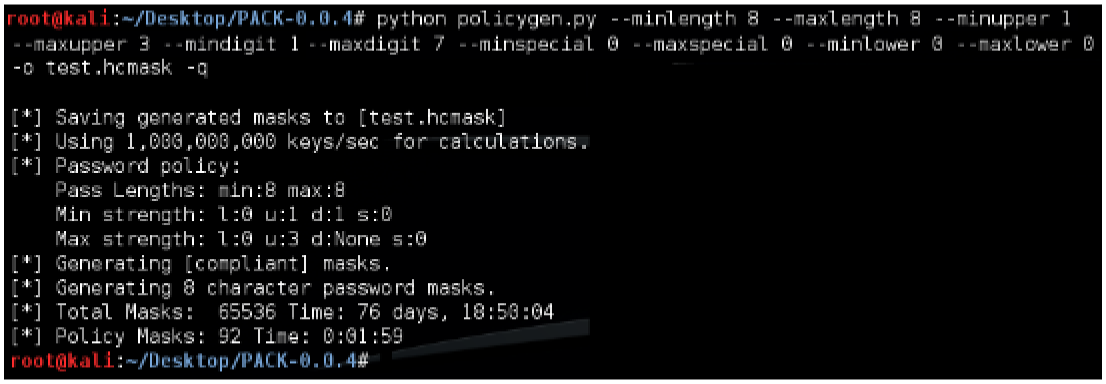
rulegen.py common.txt -q
common.txt in picture above is an ordinary wordlist

A = saving a rule file for us to go (it's a lot of duplicate rules) that we can use in if we want
B = an analyze file with from our file list
C= Statistic word containing the following words / the following rules is used

The second program is the Here you can create masks with I mean with that is that I can control the masks, if I want to save time cracking and I know for instance that the key looks at a special way, it's foolish to run the whole range of the mask, because it takes a lot of time burning energy in vain. We concentrate on specific pattern as example

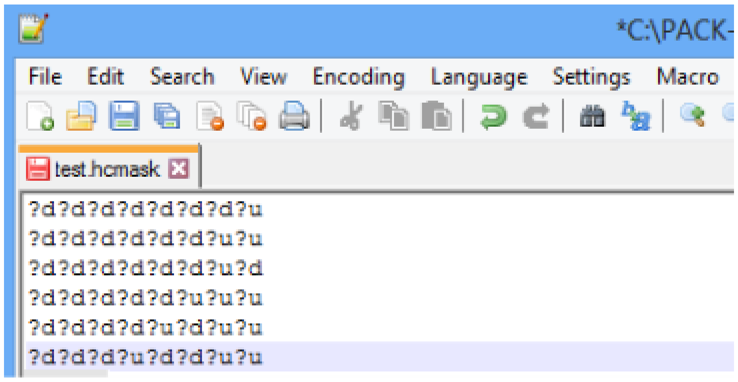
Minimum/maximum amount of letters.
Minimum/maximum amount of
Restrictions like, max 3 of the same letters in a
And of cause the pool of special characters and so

policygen.py 8 8 1 1 7 0 0 0 -- max lower 0 test.hcmask



```
root@kali:~/Desktop/PACK-0.0.4# python policygen.py --minlength 8 --maxlength 8 --minupper 1
--maxupper 3 --mindigit 1 --maxdigit 7 --minspecial 0 --maxspecial 0 --minlower 0 --maxlower 0
-o test.hcmask -q

[*] Saving generated masks to [test.hcmask]
[*] Using 1,000,000,000 keys/sec for calculations.
[*] Password policy:
    Pass Lengths: min:8 max:8
    Min strength: l:0 u:1 d:1 s:0
    Max strength: l:0 u:3 d:None s:0
[*] Generating [compliant] masks.
[*] Generating 8 character password masks.
[*] Total Masks:  65536 Time: 76 days, 18:50:04
[*] Policy Masks: 92 Time: 0:01:59
root@kali:~/Desktop/PACK-0.0.4#
```

Above command will create a file called **test.hcmask** with your choice of masks with the limitation that you choose. Now take that file and use it with no editing needed

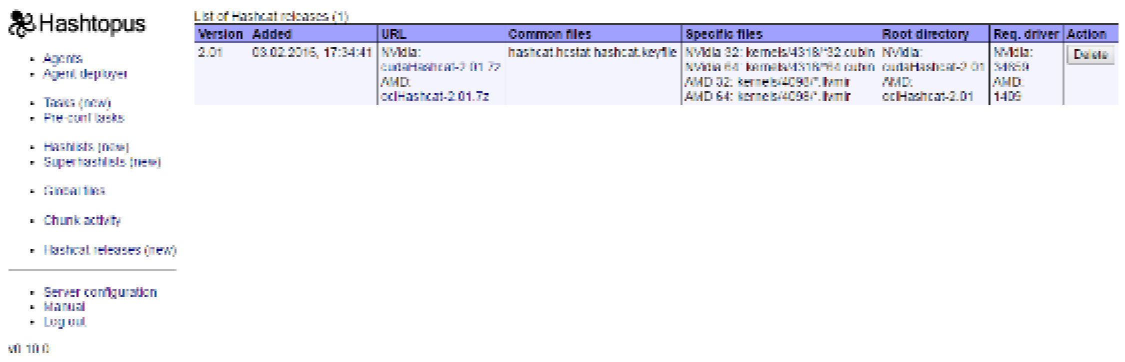


```
?d?d?d?d?d?d?d?u
?d?d?d?d?d?d?u?u
?d?d?d?d?d?d?u?d
?d?d?d?d?d?u?u?u
?d?d?d?d?u?d?u?u
?d?d?d?u?d?d?u?u
```

I will let you guys play around with this. It can be a very powerful tool to use

Smart ways to run

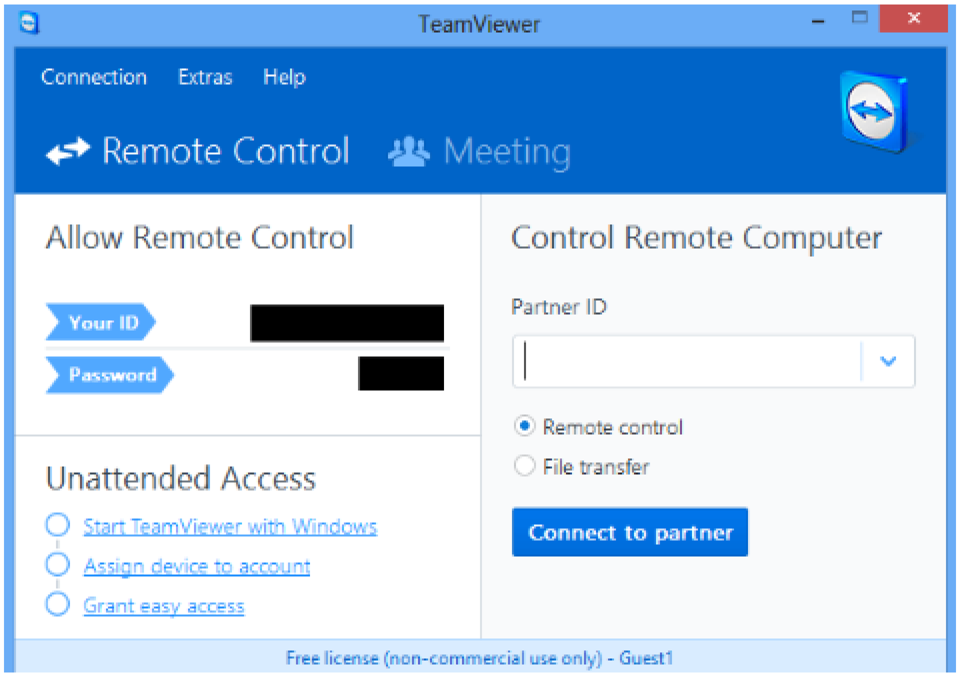
One of the smarter ways to control especially if you're running more than one rig is to run You have one controller/agent and one server part that's pretty easy to install to any computer in a LAN or WAN connection, running the main program. And you install the client on every machine with proper hardware. The idea is that it will distribute the hashes by of it down to the rigs, and will have a central role in this. You set it up as a web-server so you can log in to see the progress, as long as you have a connection it will run smoothly. way you both have full control over the rigs and hashes. Just use a web-browser to go to the controller and log in to its You can connect as many rigs or gaming computer as you have access to and that means more power and faster Below is an older version of Hashtopus



List of Hashcat releases (1)

Version	Added	URL	Common files	Specific files	Root directory	Req. driver	Action
2.01	03.02.2016, 17:34:41	NVidia: cudaHashcat-2.01.7z AMD: oclHashcat-2.01.7z	hashcat.hcstat hashcat.keyfile	NVidia 32: kernels/4318/*32.cubin NVidia 64: kernels/4318/*64.cubin AMD 32: kernels/4098/*.llvmir AMD 64: kernels/4098/*.llvmir	NVidia: cudaHashcat-2.01 AMD: oclHashcat-2.01	NVidia: 34659 AMD: 1409	Delete

Another good way to control your rig when not being at the place physically is the TeamViewer program. It’s easy to install and use. Here you can log in to the transfer hash start cracking and so on. Nowadays it’s possible to install remote desktops on our smart phones as well, and that will give us a lot of mobility, because you don’t need to baby-sit the cracking rig, you just connect through your phone to keep track, add new hashes and so



Allow Remote Control

Your ID [redacted]

Password [redacted]

Unattended Access

- Start TeamViewer with Windows
- Assign device to account
- Grant easy access

Control Remote Computer

Partner ID

[✓ Remote control]

File transfer

Connect to partner

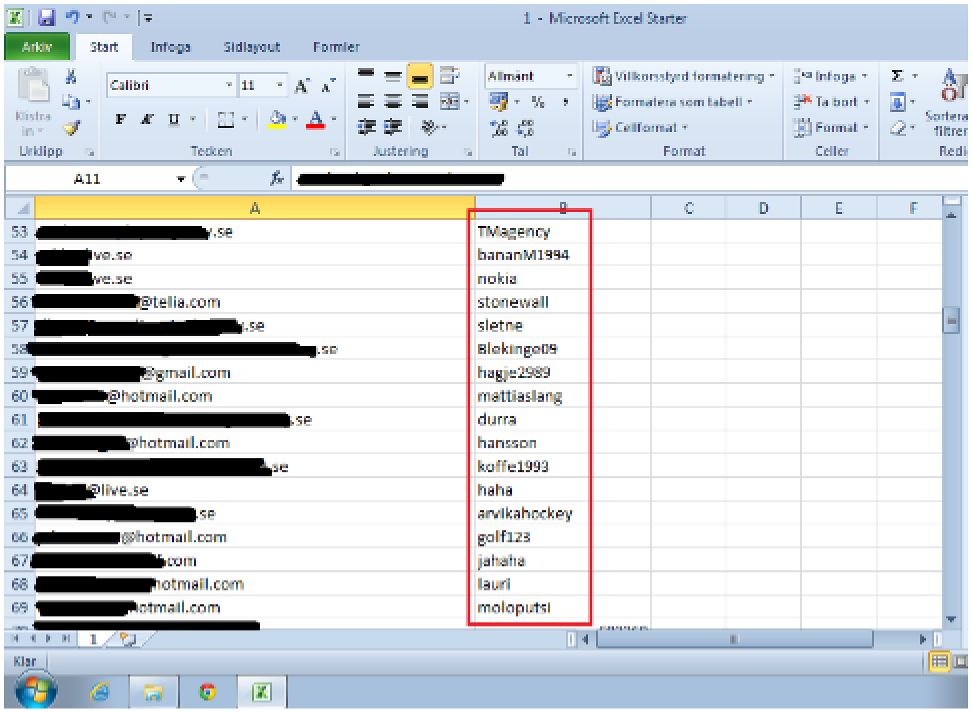
Free license (non-commercial use only) - Guest1

Wordlists, where to find those, and how to create them

There's a couple of ways to create your own password files. All of them have its advantages and disadvantages. I've seen many of you asking after some good wordlist for cracking, so I thought it would be a good idea to tell you how you do your own wordlist. I'm going to go through some ideas to find / create workable wordlists. You should try to create wordlists you can't make a mask of in It's totally unnecessary to example creating a wordlist "0-9_9 digits long"

1. Use known dumps of hacked sites!

This is people making real And as you may know people has a habit to use almost identical password everywhere preferably with dicks, porn and other less flattering vivid descriptions of himself or others. :D You just have to find an already cracked dump. Look for dumps of in your own language. The Top-level domain is a clue. If you got a dump from "site_example.cn" <-- This tells us that it's a Chinese site with (probably) Chinese passwords. Also try to look at different hacking forums, there's always people who want help to crack, or even giving away already cracked databases. Today most of us use our native language, and English so databases from both are of



	A	B
53	[redacted].se	TMagency
54	[redacted]ve.se	bananM1994
55	[redacted]ve.se	nokia
56	[redacted]@telia.com	stonewall
57	[redacted].se	sletne
58	[redacted].se	Blekinge09
59	[redacted]@gmail.com	hagje2989
60	[redacted]@hotmail.com	mattiaslang
61	[redacted].se	durra
62	[redacted]@hotmail.com	hansson
63	[redacted].se	koffe1993
64	[redacted]@live.se	haha
65	[redacted].se	arvikahockey
66	[redacted]@hotmail.com	golf123
67	[redacted].com	jahaha
68	[redacted]otmail.com	lauri
69	[redacted]otmail.com	moloputsi

Above you can see one of those databases opened in excel (Open Office might work as well in Linux). Just simply copy the password column and copy the passwords to a text editor, and save it as a pure unicode text Now there's 2 easy ways to fuse text files in to a big one in Linux.

cat *.txt >> bigfile.txt or

cat 1.txt 2.txt >> final.txt

(In windows simply "**copy file-1.txt + file-2.txt + file-3.txt**)

When you have copied your passwords into a big text it’s time to remove every password that doesn’t meet our criteria. Mine criteria for use are at least 3 characters so we can remove unnecessary stupid passwords, so we have to remove everything > 3. This is simple to do in Linux. Just open a terminal window and type

```
awk '{ if (length($0) > 3) print }' RAW_passwords.txt > Cleaned.txt
```

And it will write a new and ignore all passwords below 2 characters. However we still need to remove all And believe me, there are many duplicates in a DB. Easy enough we can use the "sort" command in Linux

```
sort -u passwordfile.lst > clean_passwordfile.lst
```

-u = Unique

This will create a file (called **clean_passwordfile.lst)** and keep one unique word from the passwordfile.lst . Depending on size this can take a while

2. Online wordlists

How about all of those wordlist often found on different sites around the net? A few of them are pretty good, but the major problems here are duplicates that will make it take longer to go through. And that's not all, in my part of the world were using the odd letters "å,ä,ö," in our alphabet, which means that many wordlists is crap. Now if I was a Russian or Greek, (Cyrillic alphabet) these wordlists would be totally useless, unless you're cracking an English database. If you are really lucky you may find wordlist in your own language, but those are small and use to contain a couple of thousand words, sadly often copies from online translation sites, which not is optimal after all. There are a few worth mention that actually work fantastic as long as the hash comes from an English or American site, (which contains English words, garbage, and such) and it's these four

Crackstation.txt

Its one wordlist 15-16GB compressed down to around 5GB. Most of our hashes tend to be found in this gigantic wordlist. There's is a possibility to donate before downloading it, but it's entire up to you

PsycOPacKv2

This "collection" of wordlist are more or less made to target WPA/WPA2 cracking Which means 8 characters and above, but it works exactly as good at other hashes. This collection is about 1,4GB compressed

Hashkiller
This is updated every week, containing the most used passwords of the week. The size is rather small compared to the others, it’s about 150mb

“100.000 most used”
Is a single text and it’s rather small compared to the others. Look in Github for more answers regarding this file

3. Compose your own
How about typing your own?? not that hard, I agree its very time consuming
Well to write your own you need to have a bit of fantasy, and I have already told you how to merge files in to one and how to clean it. The first things to hunt for are those online magazines that list top 100 common (and bad) passwords. Normally here in Sweden they lists those once a year online (because they got nothing better to write about I guess)

Pet names,

Names/Nicknames (preferably with numbers after, example Vinny1977 Lawrie81)

Porn inspired

Branches

Sports/league/players

Cars

Numbers

Movies

Flowers

Computer jargon

Star wars

Klingon

Series

Music

Hobbies

Books

Plants

And many many

4. Torrent sites

Surprisingly enough there's a lot of wordlist to download from different torrent sites around the world, mostly WPA2 ready, however there's a lot of others as

Hacked databases

The databases may look a bit different, but they all have one thing in common, Usernames, Hashes and Salt. Perhaps not in that particular order but I think you get that. In rare cases you will find databases that show the password in clear text, but most of the time its salted MD5 or

Usernames

Often an email-address or just a normal username

Hashes

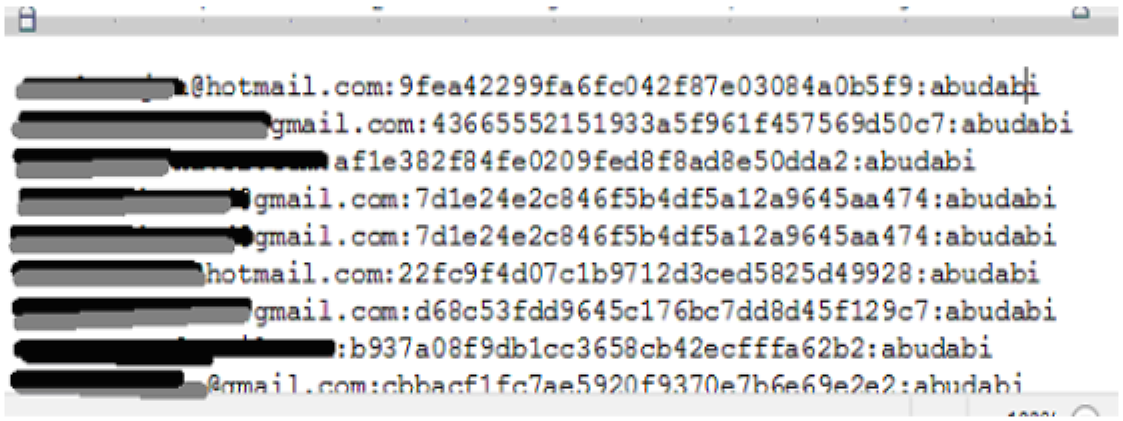
Often MD5 and SHA1, but also others depending on what type of database you got

Salt

Various length exist, I have seen from 3 characters to 15, also the placement of salt in database can differ as well, normal is before or after the hash

Other things

Things like phone numbers and more information about the user might exist, last known IP, or if the email isn't used as a username you will encounter it here

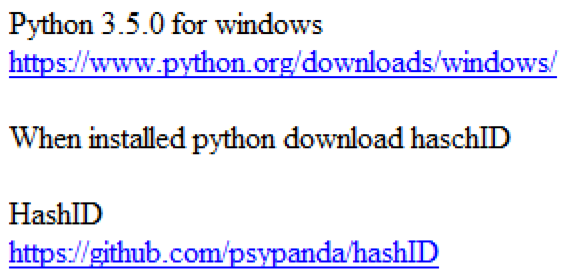


```
[redacted]@hotmail.com:9fea42299fa6fc042f87e03084a0b5f9:abudabi
[redacted]gmail.com:43665552151933a5f961f457569d50c7:abudabi
[redacted] af1e382f84fe0209fed8f8ad8e50dda2:abudabi
[redacted]gmail.com:7d1e24e2c846f5b4df5a12a9645aa474:abudabi
[redacted]gmail.com:7d1e24e2c846f5b4df5a12a9645aa474:abudabi
[redacted]hotmail.com:22fc9f4d07c1b9712d3ced5825d49928:abudabi
[redacted]gmail.com:d68c53fdd9645c176bc7dd8d45f129c7:abudabi
[redacted]:b937a08f9db1cc3658cb42ecfffa62b2:abudabi
[redacted]@gmail.com:cbbacf1fc7ae5920f9370e7b6e69e2e2:abudabi
```

As you see above usernames and hash and salt are separated by a colon ":" but that's not always the case. Other punctuation marks (separators) exist as these two as example ; and ,

Ok, so how do I know what kind of hash I got

Sometimes you get stuck with a hash and we don't know what kind of hash you have in front of you. Luckily us there's a tool called written in python, so for this to work we have to install Python 3.5.0 for windows or better, it comes with support for windows 7, 8 and 8.1 (so far). Note this program doesn't pick the right type out for you, giving you a bunch of guesses and someone of them in top 4 usually is the hash type we want


Python 3.5.0 for windows
https://www.python.org/downloads/windows/

When installed python download haschID

HashID
https://github.com/psypanda/hashID

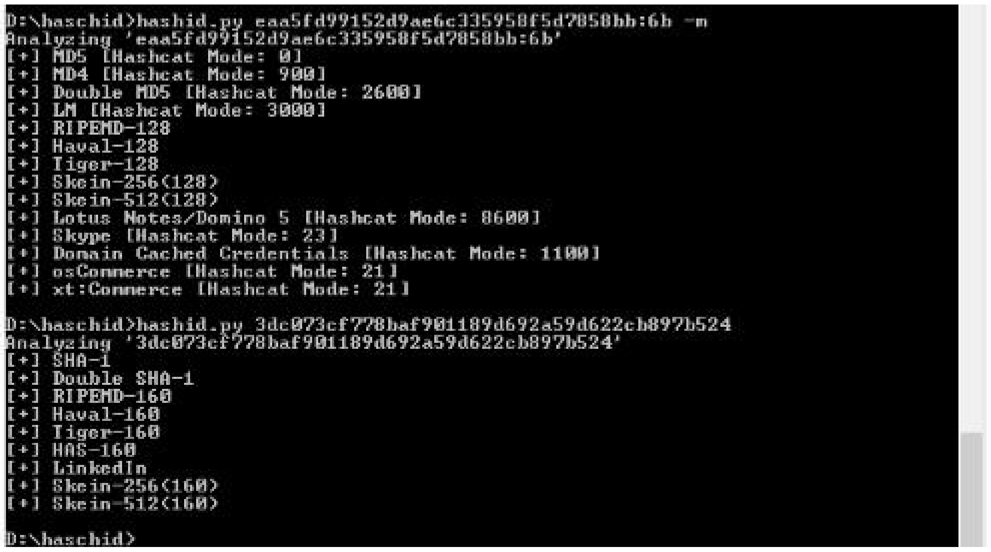
This tool is a hash type identifier for more than 220 different hash types, which is exactly what we want. So unpack it, and you're good to go! As simple as however it's required to have python It's Easy to use. Basic use is

hashid.py [hash] –m

-m = show corresponding Hashcat mode in output (example 0 for MD5)

Below you see what difference "-m" does.

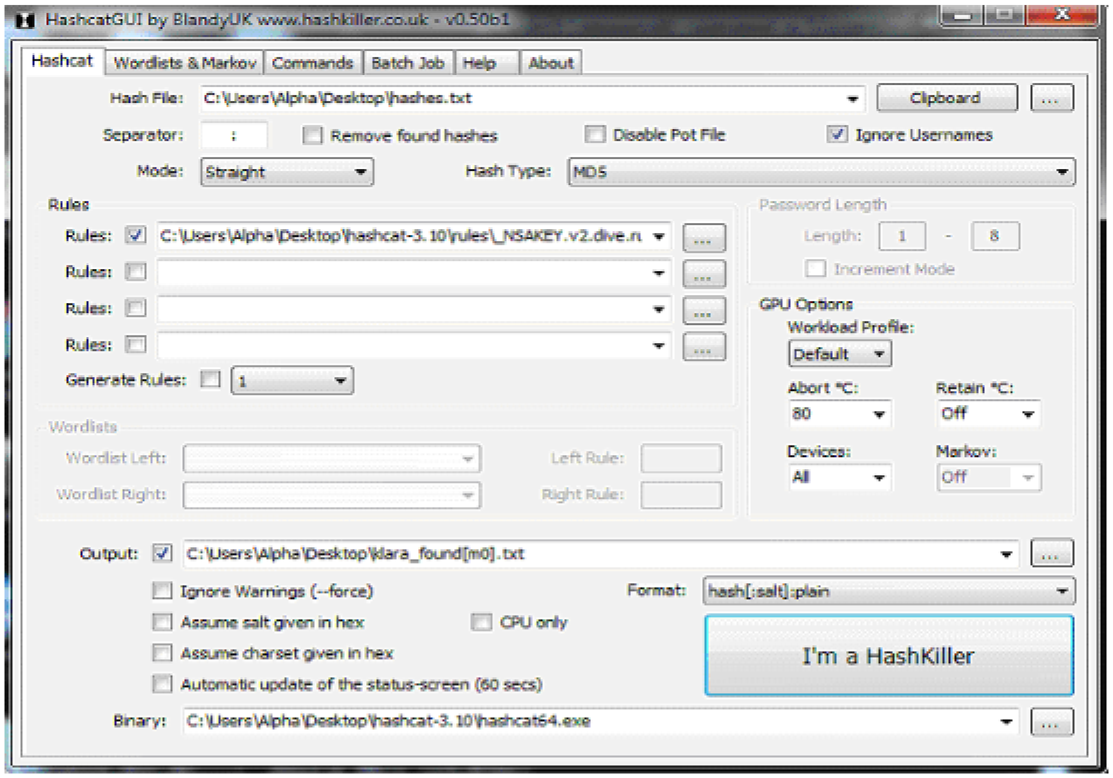
As you see in the picture below run on top with "-m" has given modes, so it's just to run with this. Often one of the suggestions in top 3



```
D:\haschid>hashid.py eaa5fd99152d9ae6c335958f5d7858bb:6b -m
Analyzing 'eaa5fd99152d9ae6c335958f5d7858bb:6b'
[+] MD5 [Hashcat Mode: 0]
[+] MD4 [Hashcat Mode: 900]
[+] Double MD5 [Hashcat Mode: 2600]
[+] LM [Hashcat Mode: 3000]
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5 [Hashcat Mode: 8600]
[+] Skype [Hashcat Mode: 23]
[+] Domain Cached Credentials [Hashcat Mode: 1100]
[+] osCommerce [Hashcat Mode: 21]
[+] xt:Commerce [Hashcat Mode: 21]

D:\haschid>hashid.py 3dc073cf778baf901189d692a59d622cb897b524
Analyzing '3dc073cf778baf901189d692a59d622cb897b524'
[+] SHA-1
[+] Double SHA-1
[+] RIPEMD-160
[+] Haval-160
[+] Tiger-160
[+] HAS-160
[+] LinkedIn
[+] Skein-256(160)
[+] Skein-512(160)

D:\haschid>
```

Hashcat GUI?

In my last book I said that the hashcat GUI was heavily outdated, but I'm proud to tell that the latest release they have support for the new hashcat This will be useful for guys that struggling with the command line. However the main idea with this book is to first learn how to use the command Note: It requires .NET Framework: v4 to work



HashcatGUI by BlandyUK www.hashkiller.co.uk - v0.50b1

Hash File: C:\Users\Alpha\Desktop\hashes.txt

Separator: :

Remove found hashes: ☐ Disable Pot File: ☐ Ignore Usernames: ✓

Mode: Straight Hash Type: MD5

Rules

Rules: ✓ C:\Users\Alpha\Desktop\hashcat-3.10\rules_NSAKEY.v2.dive.n

Rules: ☐

Rules: ☐

Rules: ☐

Generate Rules: ☐ 1

Password Length

Length: 1 - 8

Increment Mode: ☐

GPU Options

Workload Profile: Default

Abort °C: 80 Retain °C: Off

Devices: All Markov: Off

Wordlists

Wordlist Left: Left Rule:

Wordlist Right: Right Rule:

Output: ✓ C:\Users\Alpha\Desktop\klara_found[m0].txt

Ignore Warnings (--force): ☐

Format: hash[:salt]:plain

Assume salt given in hex: ☐ CPU only: ☐

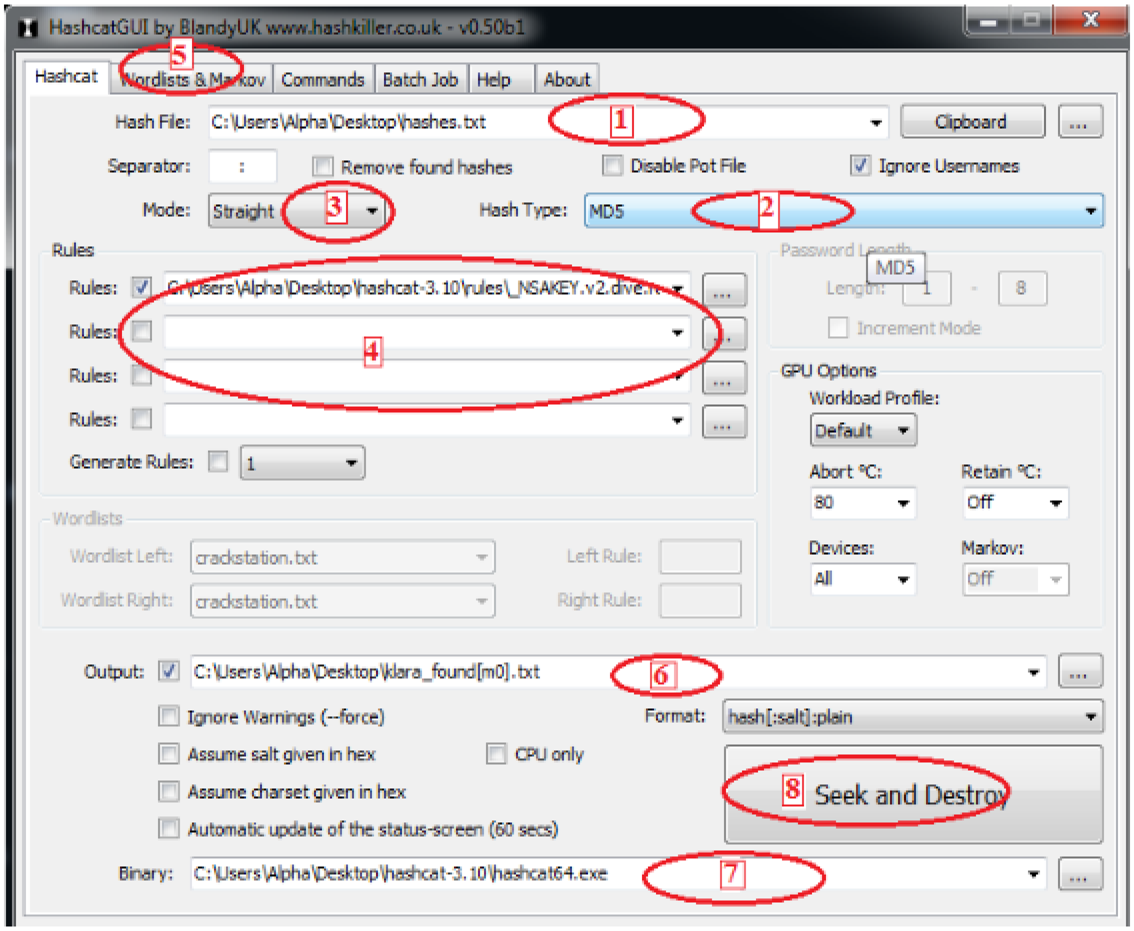
Assume charset given in hex: ☐

Automatic update of the status-screen (60 secs): ☐

Binary: C:\Users\Alpha\Desktop\hashcat-3.10\hashcat64.exe

Download link
https://hashkiller.co.uk/hashcat-gui.aspx

When starting the program, GUI will start in a command prompt window in a new Now let's go through this program together so you know how it works



Hash File: C:\Users\Alpha\Desktop\hashes.txt

Separator: :
Remove found hashes
Disable Pot File
[✓] Ignore Usernames

Mode: Straight
Hash Type: MD5

Rules
Rules: [✓] C:\Users\Alpha\Desktop\hashcat-3.10\rules_NSAKEY.v2.dive.rule
Rules:
Rules:
Rules:
Generate Rules: 1

Password Length
MD5
Length: 1 - 8
Increment Mode

GPU Options
Workload Profile: Default
Abort °C: 80
Retain °C: Off
Devices: All
Markov: Off

Wordlists
Wordlist Left: crackstation.txt
Left Rule:
Wordlist Right: crackstation.txt
Right Rule:

Output: [✓] C:\Users\Alpha\Desktop\klara_found[m0].txt
Ignore Warnings (--force)
Format: hash[:salt]:plain
Assume salt given in hex
CPU only
Assume charset given in hex
Automatic update of the status-screen (60 secs)

Binary: C:\Users\Alpha\Desktop\hashcat-3.10\hashcat64.exe

So how does it work then? I have marked a couple of fields that I'm going to cover here. I won't go very deep in this, because most of the things are self explanatory. Anyway I will explain this very short.

1.

Here you add the hash list that you want to crack, either you browse for a file or paste from the clipboard

2.

Here you tell what kind of hash it is (Similar to the "–m" function in command prompt)

3.

Attack mode. type of attack to use?

0 = Straight

1 = Combination

3 = Brute-force

6 = Hybrid + mask

7 = Hybrid mask + dictionary

4.

Which type of rules to use. Here you can use 4 rules, it will go through each rule after each other

5.

Here you add your wordlist that you're going to use. It's very simple

6.

If you want a separate output file with cracked hashes and passwords, here you add the location to the file. Hashcat will create the file if it's missing. Also if you use this and have usernames and hashes mixed in the database it's a good idea to check "ignore username" under the clipboard button

7.

Hashcat have two ways, either you run with 32 bit or 64 bit. Just mark that program you use. It's a link to that exe-file the GUI knows which of them to run

8.

Start cracking :) It's the start button. Messages on the button will vary

So talk a little about the hashes before we begin

I thought we should have a small talk about the hashes as well. First I thought about writing something about each hash, and it sounded good at the time, but after a while I saw that it would be like writing parts of it over and over and it's just a waste of time. However I will write about a couple of them, which I use daily

For each year the list grows with the hashes that are possible to crack. Nowadays it's possible to various things, all from wallets hashes to WPA hashes". The high gain graphic cards today is both powerful and low powered. As an example look at the AMDs new graphic card FX480. It draws some serious low power but performs as the much older R9 290x. NVIDIA have the 1080 that also performs real fantastic to low power. Oh well, back to the

These are the following hashes I was thinking to MD5, SHA1, SHA256, WPA2 because those are the most common ones I crack nowadays. In the end after the WPA chapter I will give you some examples on the rest but I will not go through them at all, because if you got so far as here reading my written lines you also figured out how the magic in is made, so I guess you will be fine at your own then.

The whole list below is hashes I use to crack once a while and I will give you some along the way. The list below is a small pick of the whole list that's located at site.

The number in front of the hash is the module That means that we tell hash type we want to crack with a number.

Example

Hashcat –m 0 (MD5)
Hashcat –m 100 (SHA)
Hashcat –m 1700 (SHA512)

Wait! the difference in between 10 MD5($pass.$salt) and 20 MD5($salt.$pass) then? The hash and salt looks the same! Well it's the correct order in which uses the salt in the equation to get the password

The is as follows (without HashcatXX.exe)

"-m [number][hash or hash stuff]"

Just follow this above, following this you will see that isn’t that hard to work with for a

Legally making hashes

If you don't get the proper permissions for testing your hashes there always a way to do this, by generate your own hashes. There is a lot of tools doing this, both online and different local pentesting out there, because be honest, we don't want to get in cracking some database without I know it may be tempting to do this without permission but that is just wrong and can lead to so many including jail To avoid doing stupid things like using hashes from a hacked database we can create our own hashes just to test our There are multiple sites that offer this for free, here are a few

Create MD5 hashes
http://www.miraclesalad.com/webtools/md5.php

Create SHA1 hashes
http://www.miraclesalad.com/webtools/sha1.php

Create MD4, MD5, SHA1, DES, PHPass, Blowfish hashes
http://www.finnie.org/software/randpass/demo/

Create SHA256 hashes
http://www.xorbin.com/tools/sha256-hash-calculator

Generates over 50 kinds of hashes
https://www.onlinehashcrack.com/hash-generator.php

It’s important that we do not have intent to break the law, so this is a way for you to test your skills and the wordlists, if they are good enough. There’s also a bunch of programming scripts that also comes to mind but I will not go them. To verify that the cracked hashes are correct you can visit the and verify every one of them

http://verifier.insidepro.com/

Which makes a little dumb when we already created them, but it’s a tool that you can use IF you get permission to crack some hashes, and you want to verify that you got the right hash /password.

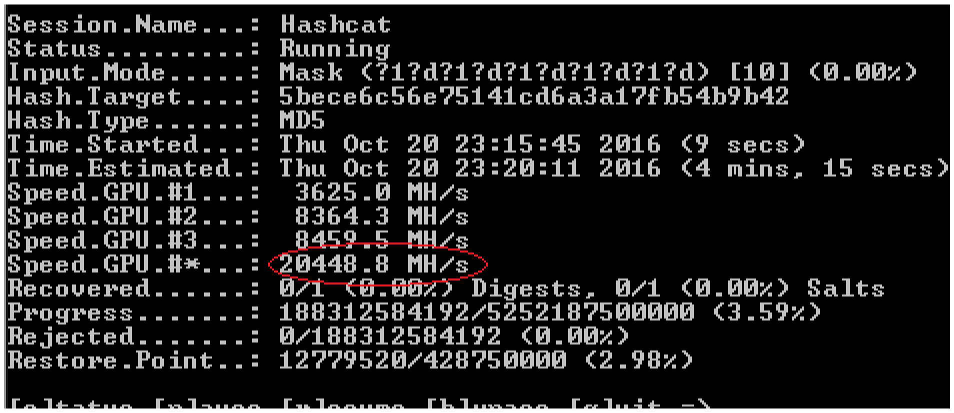
Okay, so its time to take a look t the different hashes and how to use Hashcat to crack them

MD5

So lets talk about MD5.The MD5 or “Message-Digest algorithm 5” is a widely used cryptographic hash function, which produce a 128-bit hash value (16-byte), typically expressed as a 32 digit hexadecimal number. MD5 has been utilized in a wide variety of cryptographic applications; MD5 has 24+ years under its belt, so of course there are some weaknesses in this algorithm. MD5 was designed by Ronald Rivest in 1991 to replace an earlier version called MD4, to enhance the security a salt in various length is added to those hashes to make it more harder to crack, however this hash is normally quite easy to crack so a salt added to this won’t slow the cracking process down much I’m afraid.

Cracking MD5 is one of the easiest you ever can crack and the speed doing this is enormous even for a mid grade graphics card as mine. The difference is enormous between running a dictionary attack and a mask-attack, however there’s a reason for that, (see rules chapter) it’s the GPU waiting time between each chunks of word, remember the rules section? Read that again if you want to know why it’s slow :)

I have a special “MD5.hcmask” file for this, and I can afford a little waiting, because the speed running Hashcat with MD5 is so ridicules fast.


```
Session.Name...: Hashcat
Status.........: Running
Input.Mode.....: Mask (?1?d?1?d?1?d?1?d?1?d) [10] (0.00%)
Hash.Target....: 5bece6c56e75141cd6a3a17fb54b9b42
Hash.Type......: MD5
Time.Started...: Thu Oct 20 23:15:45 2016 (9 secs)
Time.Estimated.: Thu Oct 20 23:20:11 2016 (4 mins, 15 secs)
Speed.GPU.#1...:  3625.0 MH/s
Speed.GPU.#2...:  8364.3 MH/s
Speed.GPU.#3...:  8459.5 MH/s
Speed.GPU.#*...: 20448.8 MH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 188312584192/5252187500000 (3.59%)
Rejected.......: 0/188312584192 (0.00%)
Restore.Point..: 12779520/428750000 (2.98%)
```

<u>So how to run a single MD5 hash</u>

This is easy but can be a little trickier due to the salt- we must know how to read the hash and salt. That's why we need to change mode from "-m 0" to "-m 10" or similar. See in list below

0 md5 9fea42299fa6fc042f87e03084a0b5f9

10

md5($pass.$salt)01dfae6e5d4d90d892622325959afbe:7050461

20 md5($salt.$pass)0fda5863031Oa6dd91a7d8f0a4ceda2:

f4225637426

30

md5(unicode($pass).$salt)b31d032cfdcf47a399990a71e43c5d2a:1448

16

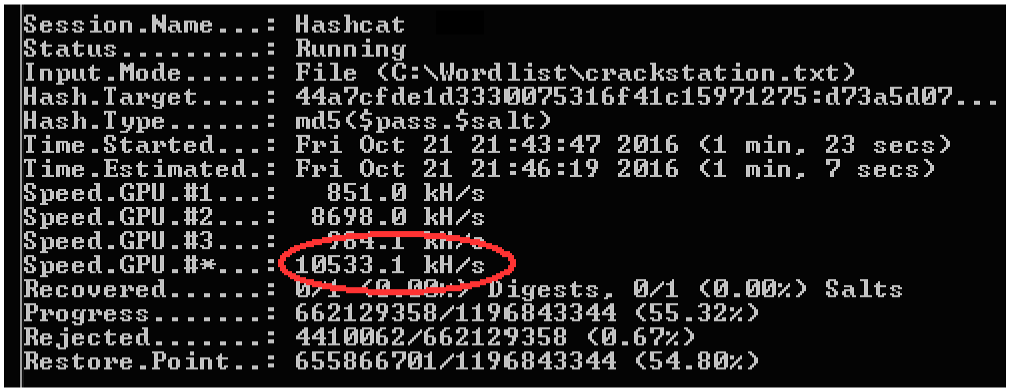
40

md5($salt.unicode($pass))d63d0e21fdc05f618d55ef306c54af82:13288442151473

These above are five usual can handle MD5, however the list is actually A single salted MD5 doesn't provide any challenge, at least when running with a single hash. In picture below running achieving ridicules speeds despite the salt that is exactly as long as the hash itself, however the speed is somewhat reduced compared to non-salted MD5 hash. The fun part is that is made in a way that makes ($pass$salt) faster to crack then ($salt$pass) between 8-10% faster actually, depending in hash

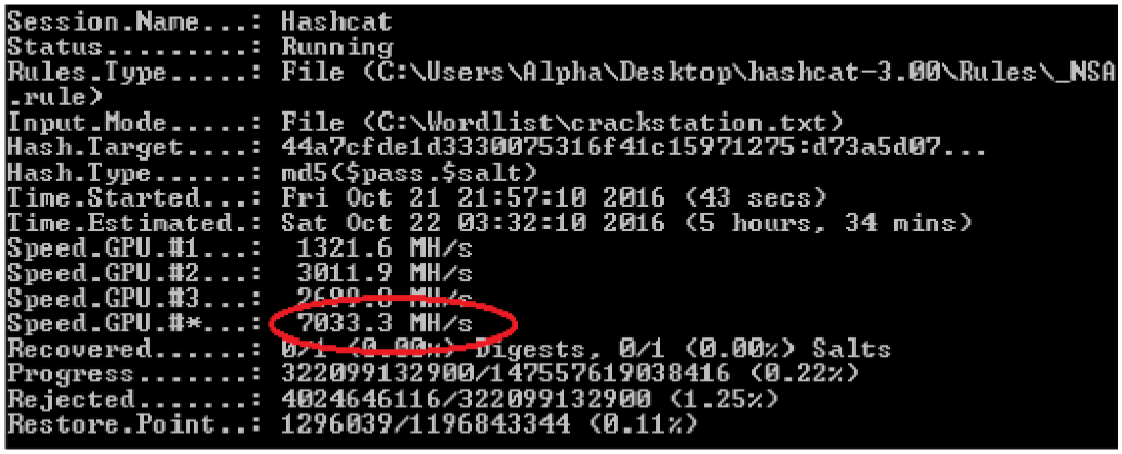
44a7cfde1d3330075316f41c15971275:d73a5d07ed321d3fcb5f430bddeb7e72

Here I'm testing running the hash above, both with a wordlist and with a mask.



```
Session.Name...: Hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: 44a7cfde1d3330075316f41c15971275:d73a5d07...
Hash.Type......: md5($pass.$salt)
Time.Started...: Fri Oct 21 21:43:47 2016 (1 min, 23 secs)
Time.Estimated.: Fri Oct 21 21:46:19 2016 (1 min, 7 secs)
Speed.GPU.#1...:   851.0 kH/s
Speed.GPU.#2...:  8698.0 kH/s
Speed.GPU.#3...:   984.1 kH/s
Speed.GPU.#*...: 10533.1 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 662129358/1196843344 (55.32%)
Rejected.......: 4410062/662129358 (0.67%)
Restore.Point..: 655866701/1196843344 (54.80%)
```

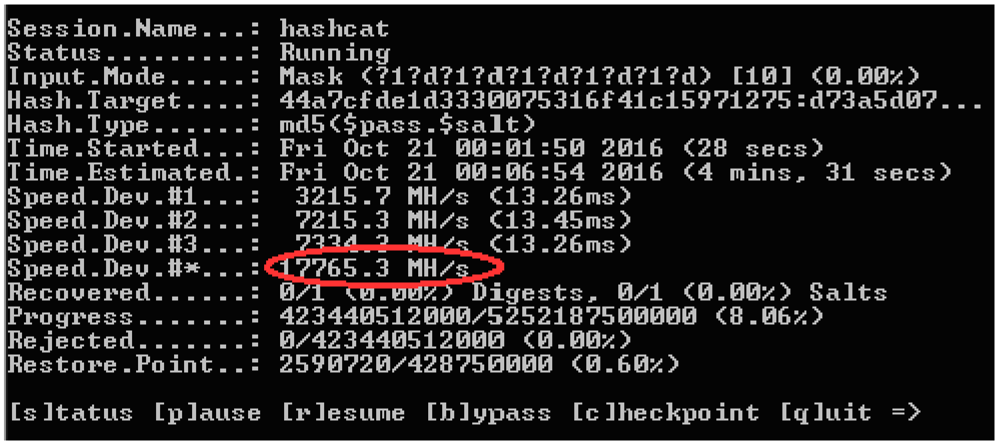
Sure it's uneven in speed and load in the picture above I know due to obvious reasons. We can always this with rules see below. However estimated time will be longer



```
Session.Name...: Hashcat
Status.........: Running
Rules.Type.....: File (C:\Users\Alpha\Desktop\hashcat-3.00\Rules\_NSA
.rule)
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: 44a7cfde1d3330075316f41c15971275:d73a5d07...
Hash.Type......: md5($pass.$salt)
Time.Started...: Fri Oct 21 21:57:10 2016 (43 secs)
Time.Estimated.: Sat Oct 22 03:32:10 2016 (5 hours, 34 mins)
Speed.GPU.#1...:  1321.6 MH/s
Speed.GPU.#2...:  3011.9 MH/s
Speed.GPU.#3...:  2699.8 MH/s
Speed.GPU.#*...:  7033.3 MH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 322099132900/147557619038416 (0.22%)
Rejected.......: 4024646116/322099132900 (1.25%)
Restore.Point..: 1296039/1196843344 (0.11%)
```

Mission accomplished!

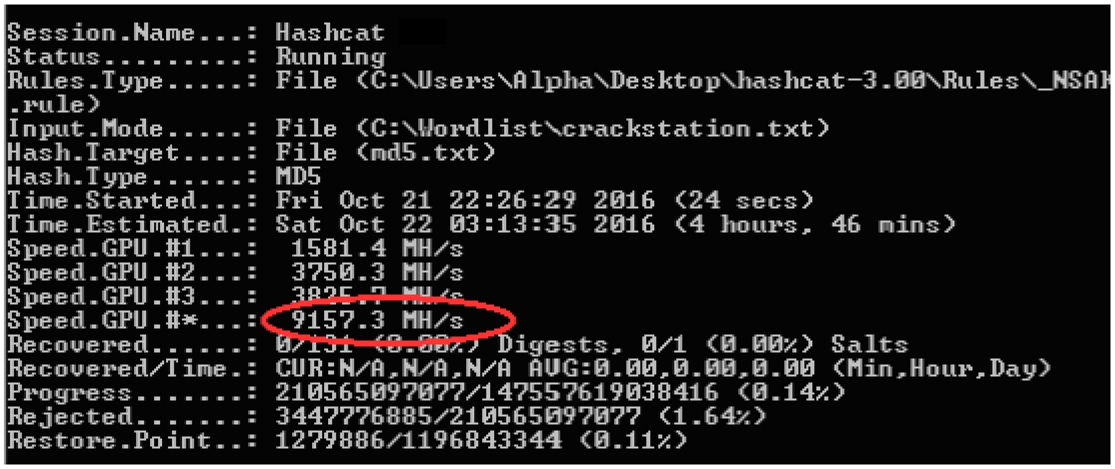
In the picture below running a mask attack, still against only one hash.



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?d?1?d?1?d?1?d?1?d) [10] (0.00%)
Hash.Target....: 44a7cfde1d3330075316f41c15971275:d73a5d07...
Hash.Type......: md5($pass.$salt)
Time.Started...: Fri Oct 21 00:01:50 2016 (28 secs)
Time.Estimated.: Fri Oct 21 00:06:54 2016 (4 mins, 31 secs)
Speed.Dev.#1...:  3215.7 MH/s (13.26ms)
Speed.Dev.#2...:  7215.3 MH/s (13.45ms)
Speed.Dev.#3...:  7334.3 MH/s (13.26ms)
Speed.Dev.#*...: 17765.3 MH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 423440512000/5252187500000 (8.06%)
Rejected.......: 0/423440512000 (0.00%)
Restore.Point..: 2590720/428750000 (0.60%)

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

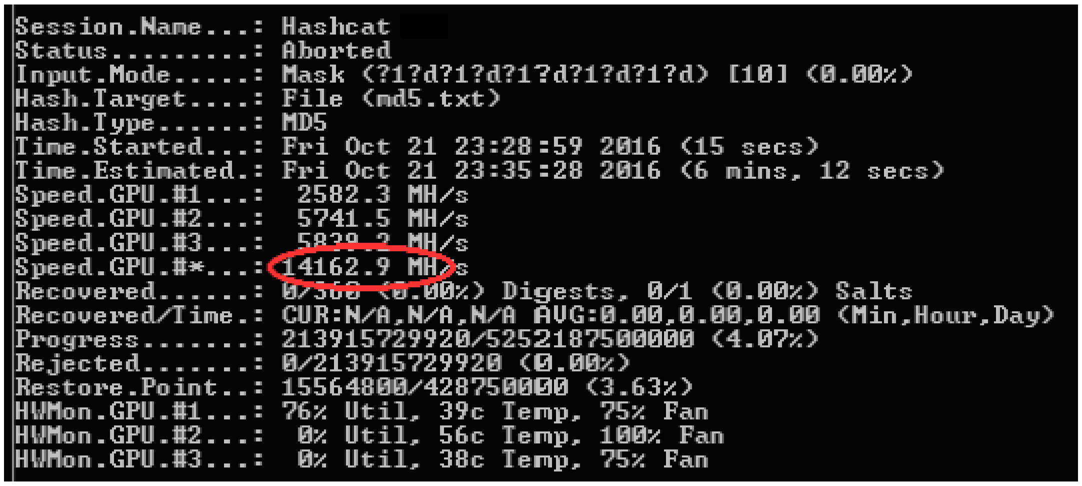
So how to run a file of hashes

Now testing with a bunch of hashes in a text as we can see here the speeds dropped a major bit, but still



```
Session.Name...: Hashcat
Status.........: Running
Rules.Type.....: File (C:\Users\Alpha\Desktop\hashcat-3.00\Rules\_NSAK
.rule)
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: File (md5.txt)
Hash.Type......: MD5
Time.Started...: Fri Oct 21 22:26:29 2016 (24 secs)
Time.Estimated.: Sat Oct 22 03:13:35 2016 (4 hours, 46 mins)
Speed.GPU.#1...:  1581.4 MH/s
Speed.GPU.#2...:  3750.3 MH/s
Speed.GPU.#3...:  3825.7 MH/s
Speed.GPU.#*...:  9157.3 MH/s
Recovered......: 0/131 (0.00%) Digests, 0/1 (0.00%) Salts
Recovered/Time.: CUR:N/A,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 210565097077/147557619038416 (0.14%)
Rejected.......: 3447776885/210565097077 (1.64%)
Restore.Point..: 1279886/1196843344 (0.11%)
```

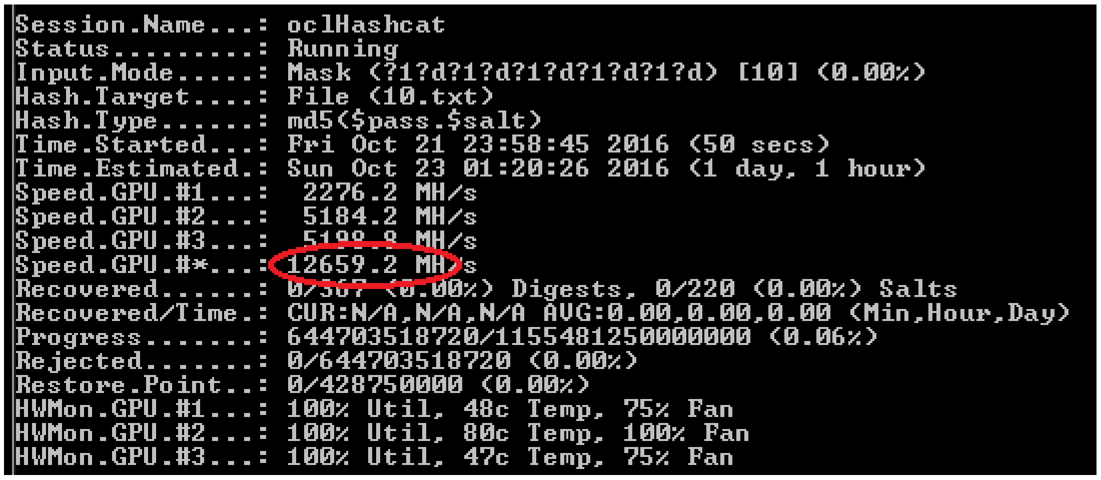
Below I'm running with plain MD5, compare that to the above, and there you can see what happens when you release the full raw power of a mask on my medium range Graphics cards.



```
Session.Name...: Hashcat [redacted]
Status.........: Aborted
Input.Mode.....: Mask (?1?d?1?d?1?d?1?d?1?d) [10] (0.00%)
Hash.Target....: File (md5.txt)
Hash.Type......: MD5
Time.Started...: Fri Oct 21 23:28:59 2016 (15 secs)
Time.Estimated.: Fri Oct 21 23:35:28 2016 (6 mins, 12 secs)
Speed.GPU.#1...:  2582.3 MH/s
Speed.GPU.#2...:  5741.5 MH/s
Speed.GPU.#3...:  5839.2 MH/s
Speed.GPU.#*...: 14162.9 MH/s
Recovered......: 0/360 (0.00%) Digests, 0/1 (0.00%) Salts
Recovered/Time.: CUR:N/A,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 213915729920/5252187500000 (4.07%)
Rejected.......: 0/213915729920 (0.00%)
Restore.Point..: 15564800/428750000 (3.63%)
HWMon.GPU.#1...: 76% Util, 39c Temp, 75% Fan
HWMon.GPU.#2...:  0% Util, 56c Temp, 100% Fan
HWMon.GPU.#3...:  0% Util, 38c Temp, 75% Fan
```

As you see running a text file with salted hashes does really make the graphics cards work, the speed is more reduced than before and this is the major point with salt slow the cracking down a bit. It's the time that is of the essence when a cracker crack hashes. If it takes to long he might just abort the cracking.

going to see some big difference further down when we come to more slower and advanced hashes. I think we will a major difference between with normal SHA1 and salted in the next



```
Session.Name...: oclHashcat
Status.........: Running
Input.Mode.....: Mask (?1?d?1?d?1?d?1?d?1?d) [10] (0.00%)
Hash.Target....: File (10.txt)
Hash.Type......: md5($pass.$salt)
Time.Started...: Fri Oct 21 23:58:45 2016 (50 secs)
Time.Estimated.: Sun Oct 23 01:20:26 2016 (1 day, 1 hour)
Speed.GPU.#1...:  2276.2 MH/s
Speed.GPU.#2...:  5184.2 MH/s
Speed.GPU.#3...:  5198.8 MH/s
Speed.GPU.#*...: 12659.2 MH/s
Recovered......: 0/367 (0.00%) Digests, 0/220 (0.00%) Salts
Recovered/Time.: CUR:N/A,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 644703518720/1155481250000000 (0.06%)
Rejected.......: 0/644703518720 (0.00%)
Restore.Point..: 0/428750000 (0.00%)
HWMon.GPU.#1...: 100% Util, 48c Temp, 75% Fan
HWMon.GPU.#2...: 100% Util, 80c Temp, 100% Fan
HWMon.GPU.#3...: 100% Util, 47c Temp, 75% Fan
```

So back to the point

So how to run a single hash

hashcat64.exe -m 0 68df4f1af360fe583ad9d73576674647 -a 3 md5.hcmask

hashcat64.exe -m 0 68df4f1af360fe583ad9d73576674647 -a 3 -1 ?l?u?s 1?1?1?d?d?1

hashcat64.exe -m 10 68df4f1af360fe583ad9d73576674647:123 d:\wordlists

hashcat64.exe -m 20 68df4f1af360fe583ad9d73576674647 d:\wordlists\pass.txt

-m 0 = choosing MD5

-m 10 = choosing MD5 ($pass.$salt) with salt

-m 20 = choosing MD5 ($salt.$pass) with salt

My recommendations for cracking a MD5 "Single Hash"

```
hashcat64.exe -m [number] [hash and optional salt] -a 3
md5.hcmask
```

Or simply

```
hashcat64.exe -m [number] [hash and optional salt]
d:\wordlists\wordlist1 -w 3
```

Why I recommend a mask file is because the speeds are so great it be a good idea to do this in a file, because we can search more effectively. In example below it goes through 4, 5, 6, 7, 8 and 9 characters both small and capital letters and numbers, and because the speed is that fast it's possible to do that. The number of salted hashes may affect the number of masks used; perhaps it's best to run with only 8 masks when running with 2500 salted hashes, depending on your system.

```
?l?u?d,?1?1?1?1


?l?u?d,?1?1?1?1?1
?l?u?d,?1?1?1?1?1?1
?l?u?d,?1?1?1?1?1?1?1
?l?u?d,?1?1?1?1?1?1?1?1
?l?u?d,?1?1?1?1?1?1?1?1?1
```

My recommendations for cracking MD5 “file with hashes”

hashcat64.exe -m [nbr] [hash file] -o [cracked hash file] [rules] [password]
hashcat64.exe -m [nbr] [hash file] -o [cracked hash file] -a 3 my.hcmask -w 3

As example

hashcat64.exe -m 0 68df4f1af360fe583ad9d735766746447 -a 3 md5.hcmask -w 3
hashcat64.exe -m 0 68df4f1af360fe583ad9d735766746447 d:\wordlists -w 3
hashcat64.exe -m 0 68df4f1af360fe583ad9d735766746447 d:\wordlists\pass.txt -w 3

-m 0 = choosing MD5
68df4f1af360fe583ad9d735766746447 = a single hash
-a 3 md5.hcmask = Attack mode, custom-character set (mask attack) with those masks I provided in the file (file called md5.hcmask)

-w 3 = the workload putting in cracking. is normal. Running with will give you a couple of thousand more hashes more per second, but keep an eye at the heat, it will arise. You could set it on **-w 4** as well but keep an eye at the temperature on your GPUs!!!

D:\wordlist = a folder or disk where the actual files are located And there's nothing different running with a salt.
\wordlists\pass.txt = a specific wordlist / wordlists

Also running with a hash list, there's a couple of alternatives, this time I use salted hash and the only thing as changed are the numbers after "-m" There's a workaround if you want the result replicated in to a specific file, we have to specify an with -o and doing this the hashes will NOT be saved in Hashcat pot file

hashcat64.exe -m 20 hash.txt [mask or wordlist]-o c:\cracked.txt
-o = output file

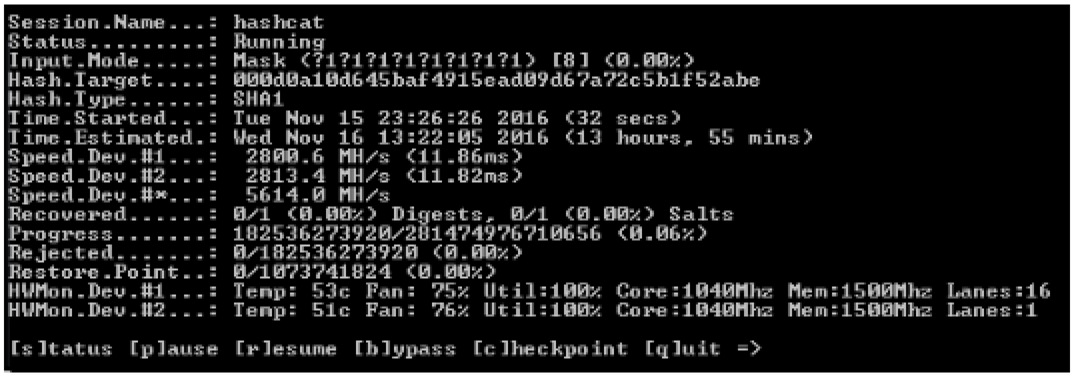
Most common MD5 Hash types (number)

0= md5
10 = md5($pass.$salt)
20 = md5($salt.$pass)
30 = md5(unicode($pass).$salt)
40 = md5($salt.unicode($pass)
3800 = md5($salt.$pass.$salt)

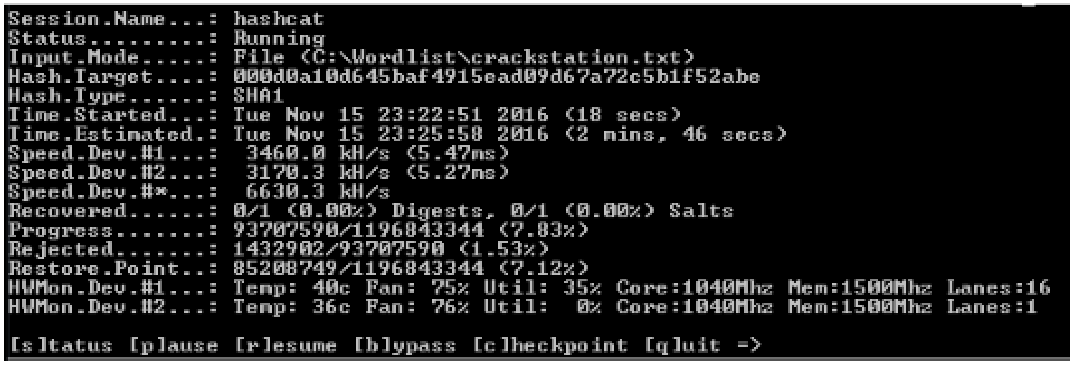
There's a lot more md5 hash types, but as a beginner you can start with those

SHA1

So let's talk about SHA1. SHA1 (Secure Hash Algorithm 1) or SHA-1 is a cryptographic hash function, and its designed by the United States "National Security Agency" and a U.S. Federal Information Processing Standard published by the United States, published 1995. It has used 15years by now so naturally it has proven to have some SHA-1 produces a 160-bit (20-byte) hash value known as a message digest. A SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long. just as popular as MD5. When it comes to cracking, still fast speeds we're talking about, but true that this is tougher to crack than MD5. Here below I'm just cracking one hash both with mask and with a wordlist and can see it's a descent speed achieved



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1?1?1) [8] (0.00%)
Hash.Target....: 000d0a10d645baf4915ead09d67a72c5b1f52abe
Hash.Type......: SHA1
Time.Started...: Tue Nov 15 23:26:26 2016 (32 secs)
Time.Estimated.: Wed Nov 16 13:22:05 2016 (13 hours, 55 mins)
Speed.Dev.#1...:  2800.6 MH/s (11.86ms)
Speed.Dev.#2...:  2813.4 MH/s (11.82ms)
Speed.Dev.#*...:  5614.0 MH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 182536273920/281474976710656 (0.06%)
Rejected.......: 0/182536273920 (0.00%)
Restore.Point..: 0/1073741824 (0.00%)
HWMon.Dev.#1...: Temp: 53c Fan: 75% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 51c Fan: 76% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

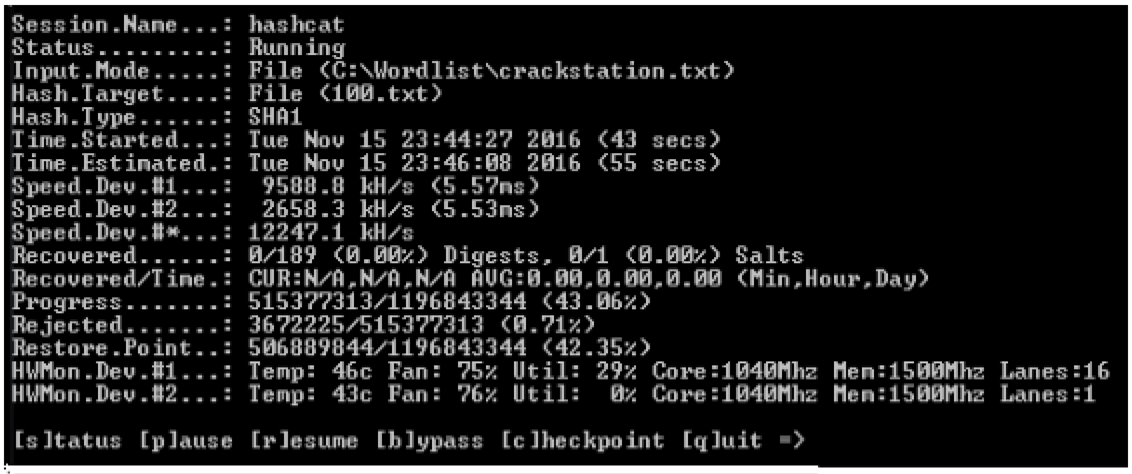


```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: 000d0a10d645baf4915ead09d67a72c5b1f52abe
Hash.Type......: SHA1
Time.Started...: Tue Nov 15 23:22:51 2016 (18 secs)
Time.Estimated.: Tue Nov 15 23:25:58 2016 (2 mins, 46 secs)
Speed.Dev.#1...:  3460.0 kH/s (5.47ms)
Speed.Dev.#2...:  3170.3 kH/s (5.27ms)
Speed.Dev.#*...:  6630.3 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 93707590/1196843344 (7.83%)
Rejected.......: 1432902/93707590 (1.53%)
Restore.Point..: 85208749/1196843344 (7.12%)
HWMon.Dev.#1...: Temp: 40c Fan: 75% Util: 35% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 36c Fan: 76% Util:  0% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

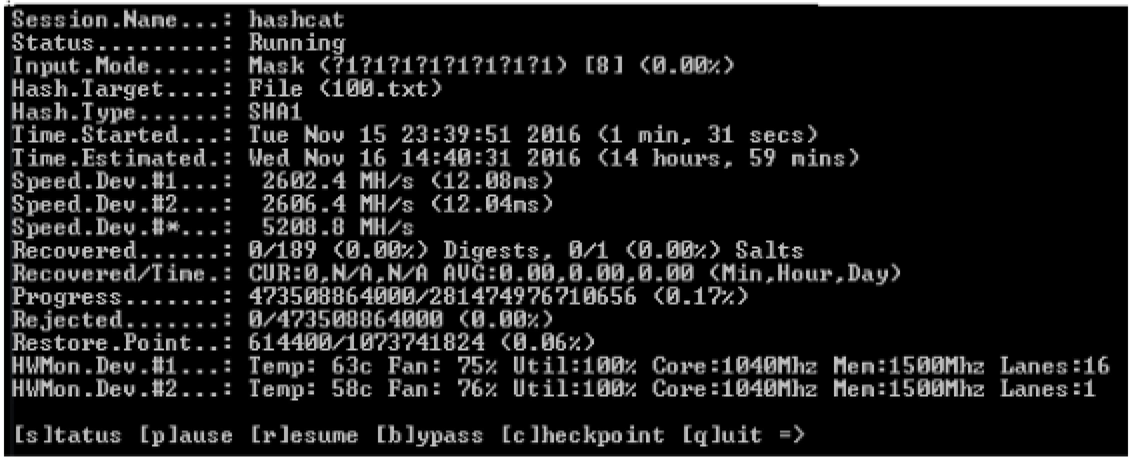
As stated in previous chapter; running with a mask attack isn't always the right yeah is still many times faster but it also have to get through a lot of crappy weird keys before you find the sometimes a wordlist can find password faster when it provides a man written passwords, preferable with

In picture below, I'm running a file with 189 hashes in a file, here you can see that it got some however running with 8 masks is the limit here (digits, uppercase and lowercase As you see we will go through 8 mask list in 14 with the speed about at each GPU which still is an enormous speed. A with first running an 8 mask brute force. Then running a wordlist on top of that is a just to get a greater coverage unless you're running with a better GPU than I have of cause.

Running 40 days (with a 9 mask) is not worth it unless for finding really really important forgotten password, normally you probably will find it before 40 days but nothing is for granted, so plan for the

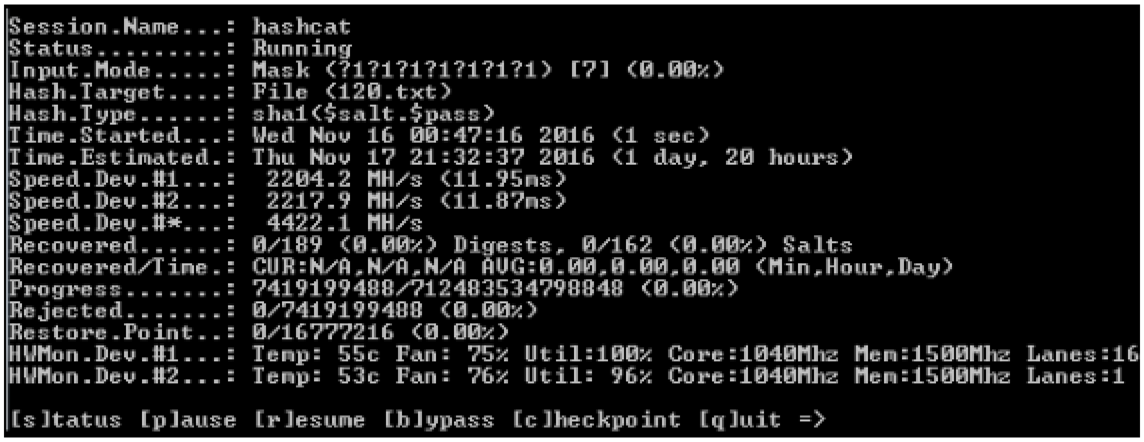


```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: File (100.txt)
Hash.Type......: SHA1
Time.Started...: Tue Nov 15 23:44:27 2016 (43 secs)
Time.Estimated.: Tue Nov 15 23:46:08 2016 (55 secs)
Speed.Dev.#1...:  9588.8 kH/s (5.57ms)
Speed.Dev.#2...:  2658.3 kH/s (5.53ms)
Speed.Dev.#*...: 12247.1 kH/s
Recovered......: 0/189 (0.00%) Digests, 0/1 (0.00%) Salts
Recovered/Time.: CUR:N/A,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 515377313/1196843344 (43.06%)
Rejected.......: 3672225/515377313 (0.71%)
Restore.Point..: 506889844/1196843344 (42.35%)
HWMon.Dev.#1...: Temp: 46c Fan: 75% Util: 29% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 43c Fan: 76% Util:  0% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1?1?1) [8] (0.00%)
Hash.Target....: File (100.txt)
Hash.Type......: SHA1
Time.Started...: Tue Nov 15 23:39:51 2016 (1 min, 31 secs)
Time.Estimated.: Wed Nov 16 14:40:31 2016 (14 hours, 59 mins)
Speed.Dev.#1...:  2602.4 MH/s (12.08ms)
Speed.Dev.#2...:  2606.4 MH/s (12.04ms)
Speed.Dev.#*...:  5208.8 MH/s
Recovered......: 0/189 (0.00%) Digests, 0/1 (0.00%) Salts
Recovered/Time.: CUR:0,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 473508864000/281474976710656 (0.17%)
Rejected.......: 0/473508864000 (0.00%)
Restore.Point..: 614400/1073741824 (0.06%)
HWMon.Dev.#1...: Temp: 63c Fan: 75% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 58c Fan: 76% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

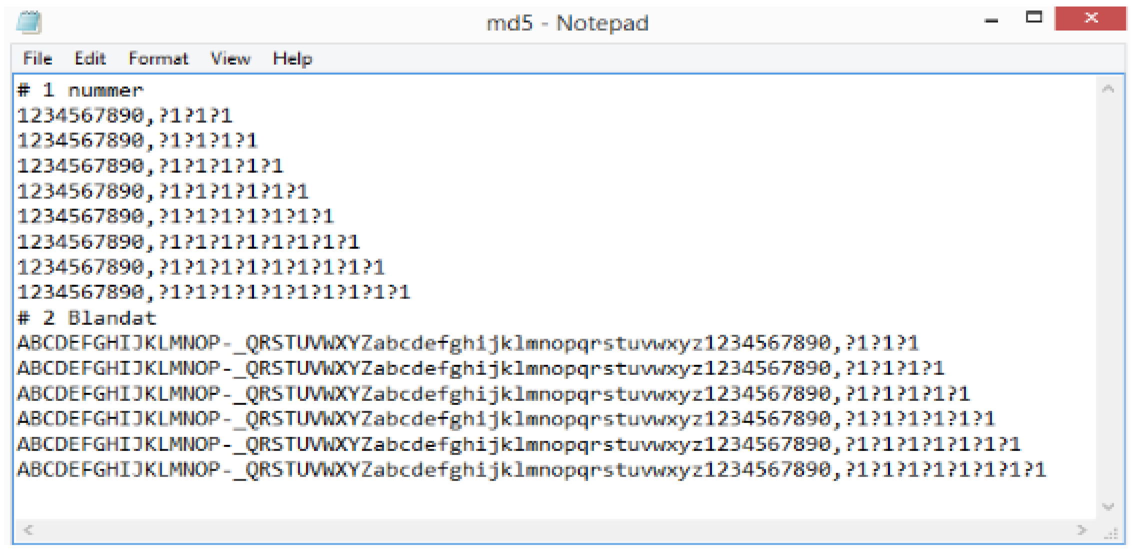
In the next picture we can see what a salt is doing to how it affects cracking speed We have to step down from 8 masks to only 7 masks, because else were spending days doing this. using the same amount of only difference is that this one is salted



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1?1) [7] (0.00%)
Hash.Target....: File (120.txt)
Hash.Type......: sha1($salt.$pass)
Time.Started...: Wed Nov 16 00:47:16 2016 (1 sec)
Time.Estimated.: Thu Nov 17 21:32:37 2016 (1 day, 20 hours)
Speed.Dev.#1...:  2204.2 MH/s (11.95ms)
Speed.Dev.#2...:  2217.9 MH/s (11.87ms)
Speed.Dev.#*...:  4422.1 MH/s
Recovered......: 0/189 (0.00%) Digests, 0/162 (0.00%) Salts
Recovered/Time.: CUR:N/A,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 7419199488/712483534798848 (0.00%)
Rejected.......: 0/7419199488 (0.00%)
Restore.Point..: 0/16777216 (0.00%)
HWMon.Dev.#1...: Temp: 55c Fan: 75% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 53c Fan: 76% Util: 96% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

The more salted hashes you have in the hash the lesser speed you will

Let’s have a second look at that md5.hcmask file, going to use it again; however we need to modify it a bit



```
# 1 nummer
1234567890,?1?1?1
1234567890,?1?1?1?1
1234567890,?1?1?1?1?1
1234567890,?1?1?1?1?1?1
1234567890,?1?1?1?1?1?1?1
1234567890,?1?1?1?1?1?1?1?1
1234567890,?1?1?1?1?1?1?1?1?1
1234567890,?1?1?1?1?1?1?1?1?1?1
# 2 Blandat
ABCDEFGHIJKLMNOP-_QRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890,?1?1?1
ABCDEFGHIJKLMNOP-_QRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890,?1?1?1?1
ABCDEFGHIJKLMNOP-_QRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890,?1?1?1?1?1
ABCDEFGHIJKLMNOP-_QRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890,?1?1?1?1?1?1
ABCDEFGHIJKLMNOP-_QRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890,?1?1?1?1?1?1?1
ABCDEFGHIJKLMNOP-_QRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890,?1?1?1?1?1?1?1?1
```

Well this is going to be an almost identical copy of MD5 chapter from here on

So running a single SHA1 there's 3 ways

hashcat64.exe -m 100 ef4a44d581846e1707c9642a5f27c9e0b78030bb -a 3 md5.hcmask -w 3
hashcat64.exe -m 100 ef4a44d581846e1707c9642a5f27c9e0b78030bb d:\word -w 3
hashcat64.exe -m 100 ef4a44d581846e1707c9642a5f27c9e0b78030bb d:\word\pass.txt -w 3

-m 100 = choosing SHA1
ef4a44d581846e1707c9642a5f27c9e0b78030bb = a single hash

-a 3 = Attack mode, custom-character set (mask attack) with those masks I provided in the file (file called md5.hcmask)

-w 3 = the workload putting in cracking. is normal. Running with will give you a couple of thousand more hashes more every second, but keep an eye at the heat, it will arise. You could set it on -w 4 as well but keep an eye at the temperature on your GPUs!!!
D:\word =a folder, disk or a specific file

Also running with a hash list, there's a couple of alternatives, this time I use salted hash and the only thing changed are the numbers after "-m"

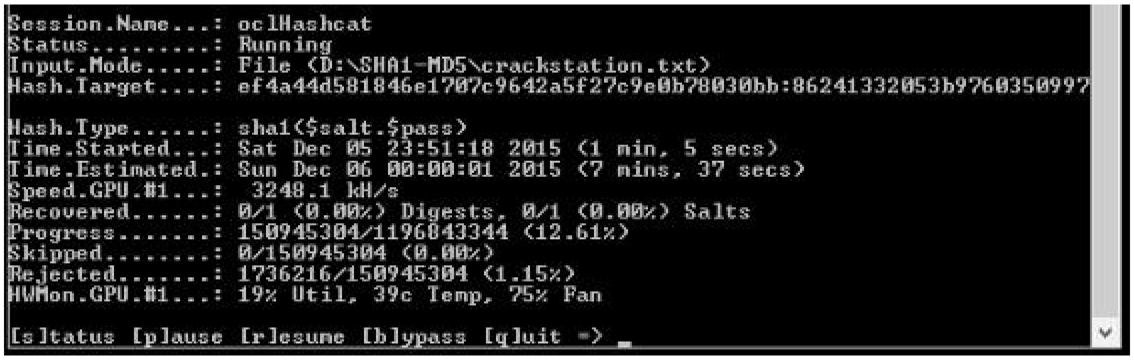
<u>SHA1 + Salt</u>
ef4a44d581846e1707c9642a5f27c9e0b78030bb:86241332053b976035 0997

hashcat64.exe -m 120 hash.txt D:\SHA1-MD5
hashcat64.exe -m 120 hash.txt -a 3 my.hcmask -w 3
Or
hashcat64.exe -m 120 [hash & salt] D:\SHA1-MD5
hashcat64.exe -m 120 [hash & salt] -a 3 my.hcmask -w 3
-m 120 = choosing salted SHA1 ($salt.$pass)
There's a workaround if you want the result in to a specific file, and we don't want the result in the pot file, then we have

to specify an out file with "-o"

hashcat64.exe -m 120 hash.txt d:\password -o c:\cracked.txt
hashcat64.exe -m 120 hash.txt -a 3 my.hcmask -o c:\cracked.txt w 3
hashcat64.exe -m 120 [hash & salt] d:\password -o c:\cracked.txt
hashcat64.exe -m 120 [hash & salt] -a 3 my.hcmask -w 3
-o = output file

Ok to take a look at different ways the salt may appear



```
Session.Name...: oclHashcat
Status.........: Running
Input.Mode.....: File (D:\SHA1-MD5\crackstation.txt)
Hash.Target....: ef4a44d581846e1707c9642a5f27c9e0b78030bb:86241332053b9760350997

Hash.Type......: sha1($salt.$pass)
Time.Started...: Sat Dec 05 23:51:18 2015 (1 min, 5 secs)
Time.Estimated.: Sun Dec 06 00:00:01 2015 (7 mins, 37 secs)
Speed.GPU.#1...:  3248.1 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 150945304/1196843344 (12.61%)
Skipped........: 0/150945304 (0.00%)
Rejected.......: 1736216/150945304 (1.15%)
HWMon.GPU.#1...: 19% Util, 39c Temp, 75% Fan

[s]tatus [p]ause [r]esume [b]ypass [q]uit => _
```

<u>Most common SHA1 Hash types (number)</u>

100 = sha1
110 = sha1($pass.$salt)
120 = sha1($salt.$pass)
130 = sha1(unicode($pass).$salt)
140 = sha1($salt.unicode($pass))

4900 = sha1($salt.$pass.$salt)

<u>**My recommendations for cracking SHA1 “Single Hash”**</u>

hashcat64.exe -m [number] [hash and maybe a salt] -a 3 md5.hcmask -w 3
hashcat64.exe -m [number] [hash and maybe a salt] d:\wordlists -w 3
hashcat64.exe -m [number] [hash and maybe a salt] [rules]d:\wordlists -w 3

<u>**My recommendations for cracking SHA1 “file with hashes”**</u>

hashcat64.exe -m [nbr] [hash file] -o [cracked hash file] [password]
hashcat64.exe -m [nbr] [hash file] -o [cracked hash file] -a 3 my.hcmask -w 3

Note! To run a clean I see no problems running with a brute force 8 mask depending on the amount of hashes you’re trying to however running a big salted SHA1 file with more than 7 masks is nothing I recommend, if you want to run with a 8 mask (specified as md5.hcmask shown before) keep the number of hashes inside hash file low unless you already have a high end graphics card or multiple graphics cards.

PHPass

This is one though hash to crack. stands for types of hashes. Wordpress, phpBB3, Joomla > 2.5.18, all of these uses this identical hash type. So what to say about this? It has some similarities with bcrypt. Its 34 character long, the whole alphabet, uppercase and lowercase letters include special characters and contains an eight-character random salt for each password, so it don't contain any external salt that you have to separate.

Unless you have a monster of a graphics card I don't recommend you running this in a list at all with an hcmask file without maybe modify it slightly. As you will see it's a hard hash to crack compared against easier MD5. In this example I'm running with only 100 hashes and my 2 current graphics cards almost walks on its knees.

So running a single there are 2 ways

```
hashcat64.exe -m 400 $P$CGxGibBllkeRP5UsSJ.en1EjhSqB/B0 -a 3 md5.hcmask -w 3
```

hashcat64.exe -m 400 $P$CGxGibBllkeRP5UsSJ.en1EjhSqB/Bo d:\word -w 3

-m 400 = choosing Phpass
a single hash
-a 3 md5.hcmask = Attack mode, custom-character set (mask attack) with those masks I provided in the file (file called md5.hcmask)

-w the workload putting in cracking. –"w 2" is normal. Running with "-w 3" will give you a couple of thousand more hashes more at every but keep an eye at the heat, it will (-w 4 also possible)
D:\word = folder, disk or a specific file

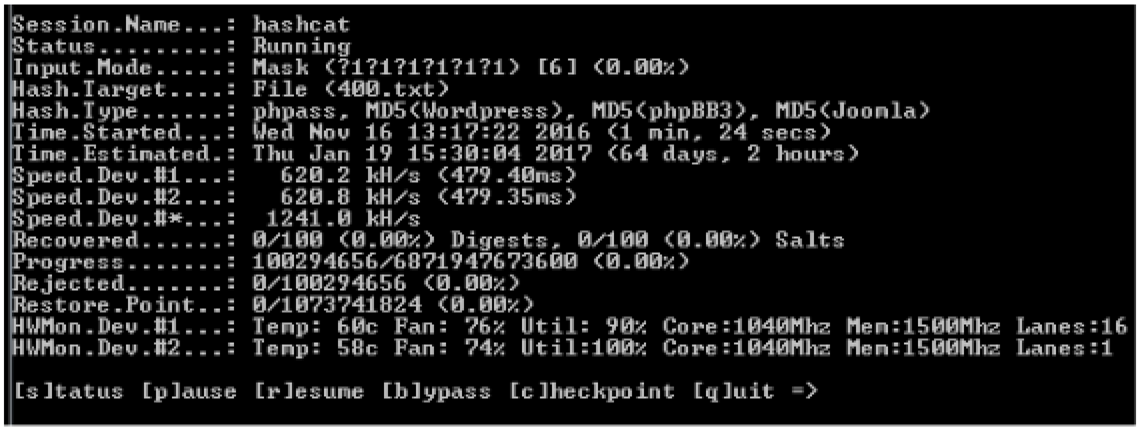
So running a text file containing hashes there are 2 ways

hashcat64.exe -m 400 hash.txt D:\SHA1-MD5
hashcat64.exe -m 400 hash.txt -a 3 my.hcmask

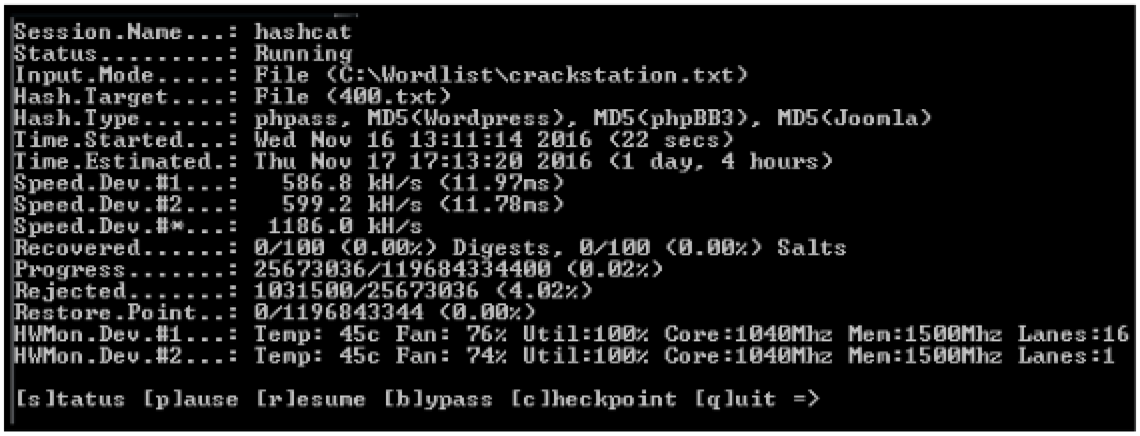
So let's start looking at what will happen running with many hashes a in this case 100 of

Now this is tough o my system. As you see in picture below running with 6 masks with 2 mid gain graphics cards it brings my system down on its knees with 64 days to pull all the masks off. I have gone down to 5 masks for it to be able to pull it off in a day. Now a better graphics card might be able to pull around 1200 kH/s on each but that's still a low speed, and you would need more than one graphics card to pull something nice out of

However you noticed that the speed doesn't change much between running with a mask or a wordlist. I had to change load to highest to be able to make some difference.

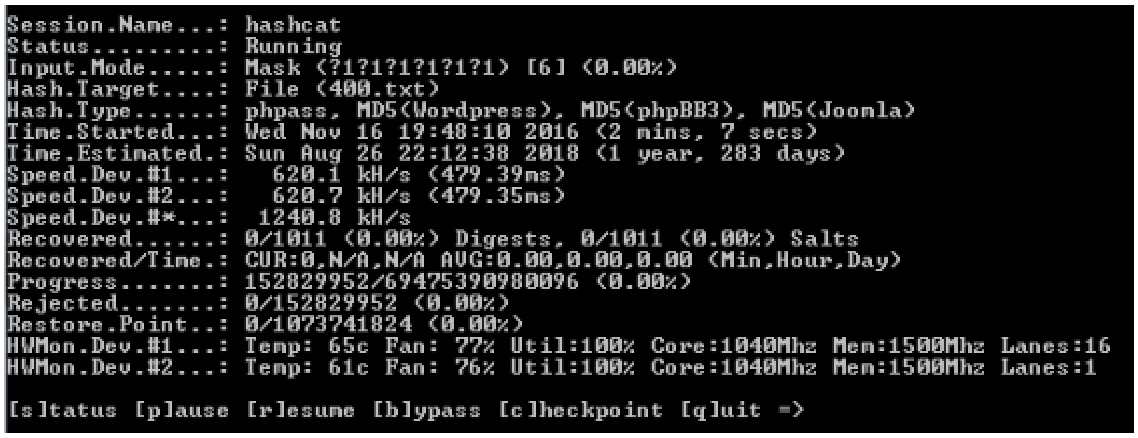


```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1) [6] (0.00%)
Hash.Target....: File (400.txt)
Hash.Type......: phpass, MD5(Wordpress), MD5(phpBB3), MD5(Joomla)
Time.Started...: Wed Nov 16 13:17:22 2016 (1 min, 24 secs)
Time.Estimated.: Thu Jan 19 15:30:04 2017 (64 days, 2 hours)
Speed.Dev.#1...:   620.2 kH/s (479.40ms)
Speed.Dev.#2...:   620.8 kH/s (479.35ms)
Speed.Dev.#*...:  1241.0 kH/s
Recovered......: 0/100 (0.00%) Digests, 0/100 (0.00%) Salts
Progress.......: 100294656/6871947673600 (0.00%)
Rejected.......: 0/100294656 (0.00%)
Restore.Point..: 0/1073741824 (0.00%)
HWMon.Dev.#1...: Temp: 60c Fan: 76% Util: 90% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 58c Fan: 74% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```



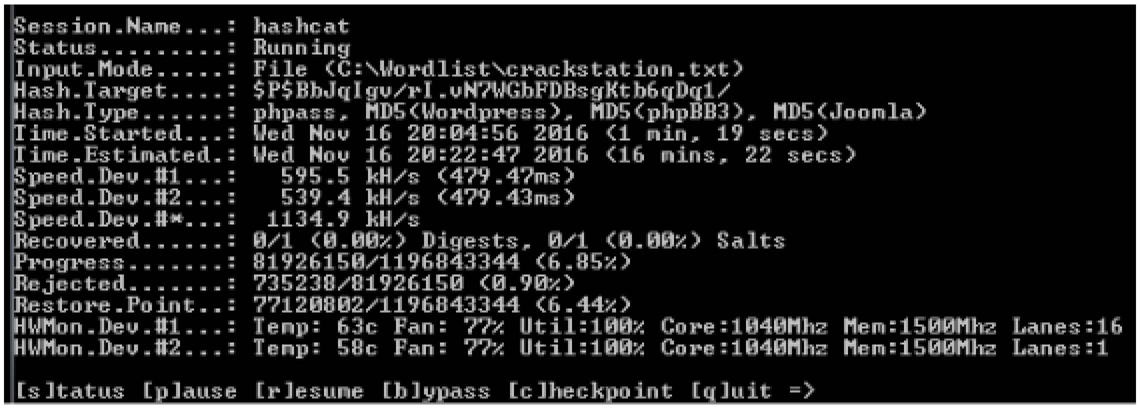
```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: File (400.txt)
Hash.Type......: phpass, MD5(Wordpress), MD5(phpBB3), MD5(Joomla)
Time.Started...: Wed Nov 16 13:11:14 2016 (22 secs)
Time.Estimated.: Thu Nov 17 17:13:20 2016 (1 day, 4 hours)
Speed.Dev.#1...:   586.8 kH/s (11.97ms)
Speed.Dev.#2...:   599.2 kH/s (11.78ms)
Speed.Dev.#*...:  1186.0 kH/s
Recovered......: 0/100 (0.00%) Digests, 0/100 (0.00%) Salts
Progress.......: 25673036/119684334400 (0.02%)
Rejected.......: 1031500/25673036 (4.02%)
Restore.Point..: 0/1196843344 (0.00%)
HWMon.Dev.#1...: Temp: 45c Fan: 76% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 45c Fan: 74% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

You see that's almost no difference between running with 100 hashes and running with hashes as in the picture below, which means that the bottleneck isn't the amount of hashes we use

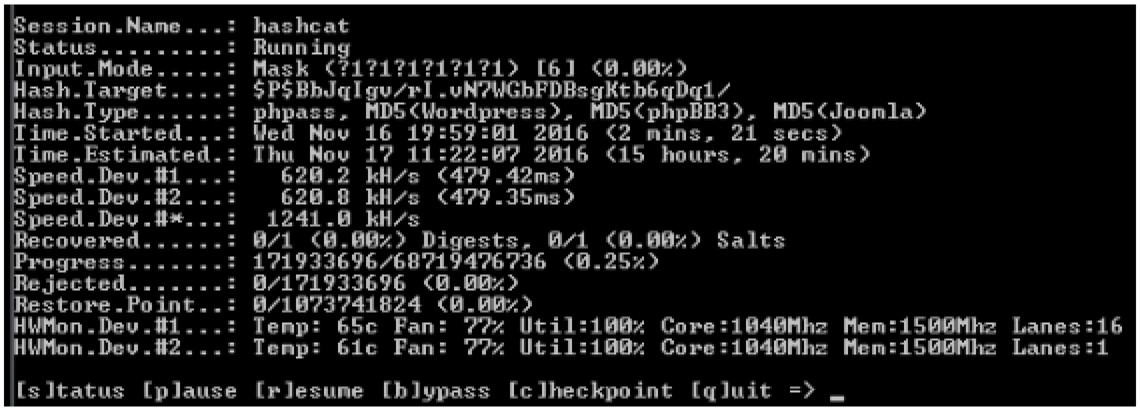


```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1) [6] (0.00%)
Hash.Target....: File (400.txt)
Hash.Type......: phpass, MD5(Wordpress), MD5(phpBB3), MD5(Joomla)
Time.Started...: Wed Nov 16 19:48:10 2016 (2 mins, 7 secs)
Time.Estimated.: Sun Aug 26 22:12:38 2018 (1 year, 283 days)
Speed.Dev.#1...:   620.1 kH/s (479.39ms)
Speed.Dev.#2...:   620.7 kH/s (479.35ms)
Speed.Dev.#*...:  1240.8 kH/s
Recovered......: 0/1011 (0.00%) Digests, 0/1011 (0.00%) Salts
Recovered/Time.: CUR:0,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 152829952/69475390980096 (0.00%)
Rejected.......: 0/152829952 (0.00%)
Restore.Point..: 0/1073741824 (0.00%)
HWMon.Dev.#1...: Temp: 65c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 61c Fan: 76% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

Running with a single hash doesn't change much as you can see in picture below, and I think it's because the hash itself is a slow So in my case I need more powerful GPUs to get more speed. Running with a rule here is maybe not the smartest thing to do because it will slow it down even further



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: $P$BbJqIgv/rI.vN7WGbFDBsgKtb6qDq1/
Hash.Type......: phpass, MD5(Wordpress), MD5(phpBB3), MD5(Joomla)
Time.Started...: Wed Nov 16 20:04:56 2016 (1 min, 19 secs)
Time.Estimated.: Wed Nov 16 20:22:47 2016 (16 mins, 22 secs)
Speed.Dev.#1...:   595.5 kH/s (479.47ms)
Speed.Dev.#2...:   539.4 kH/s (479.43ms)
Speed.Dev.#*...:  1134.9 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 81926150/1196843344 (6.85%)
Rejected.......: 735238/81926150 (0.90%)
Restore.Point..: 77120802/1196843344 (6.44%)
HWMon.Dev.#1...: Temp: 63c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 58c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1) [6] (0.00%)
Hash.Target....: $P$BbJqIgv/rI.vN7WGbFDBsgKtb6qDq1/
Hash.Type......: phpass, MD5(Wordpress), MD5(phpBB3), MD5(Joomla)
Time.Started...: Wed Nov 16 19:59:01 2016 (2 mins, 21 secs)
Time.Estimated.: Thu Nov 17 11:22:07 2016 (15 hours, 20 mins)
Speed.Dev.#1...:   620.2 kH/s (479.42ms)
Speed.Dev.#2...:   620.8 kH/s (479.35ms)
Speed.Dev.#*...:  1241.0 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 171933696/68719476736 (0.25%)
Rejected.......: 0/171933696 (0.00%)
Restore.Point..: 0/1073741824 (0.00%)
HWMon.Dev.#1...: Temp: 65c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 61c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => _
```

Most common Hash types (number)

400 =

My recommendations for cracking PHPass "Single

hashcat64.exe -m 400 d:\wordlists -w 3

This is actually worth running first

hashcat64.exe -m 400 -a 3 md5.hcmask -w 3

The only time I would be fine running a brute force mask attack is if we heavily change the md5.hcmask file 122kH/s are an ok speed but not that great.

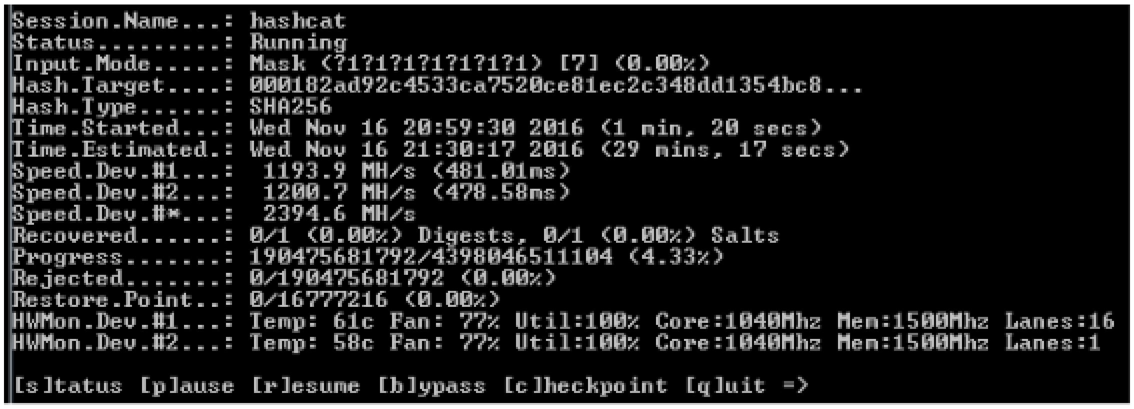
My recommendations for cracking PHPass “file with

```
hashcat64.exe -m 400 -o [output file] [password location] –w 3
hashcat64.exe -m 400 -o [output file] -a 3 my.hcmask -w 3
```

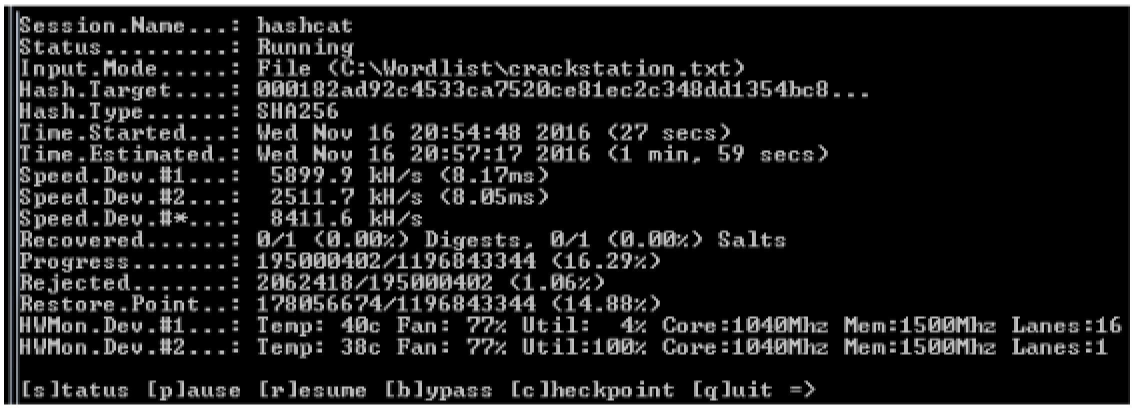
Note! The only time I would be fine running a brute force mask attack is if we heavily change the md5.hcmask and if get some more powerful graphics reduce the number of hashes in file for higher speed

SHA256

So let's talk about SHA256. SHA256 (Secure Hash Algorithm 256) as belong to the SHA-2 family, is a cryptographic hash function, and its designed by the United States "National Security Agency" and a U.S. Federal Information Processing Standard published by the United States SHA256 produces a 256-bit hash and it provides a 128bits of security against collisions. A SHA256 hash value is typically rendered as a number, 64 digits long. Despite what many believe this is a fast hash to crack. The speed is getting down a due to the difficulty. In the picture below we run with only one hash and still getting with a mask attack and with a

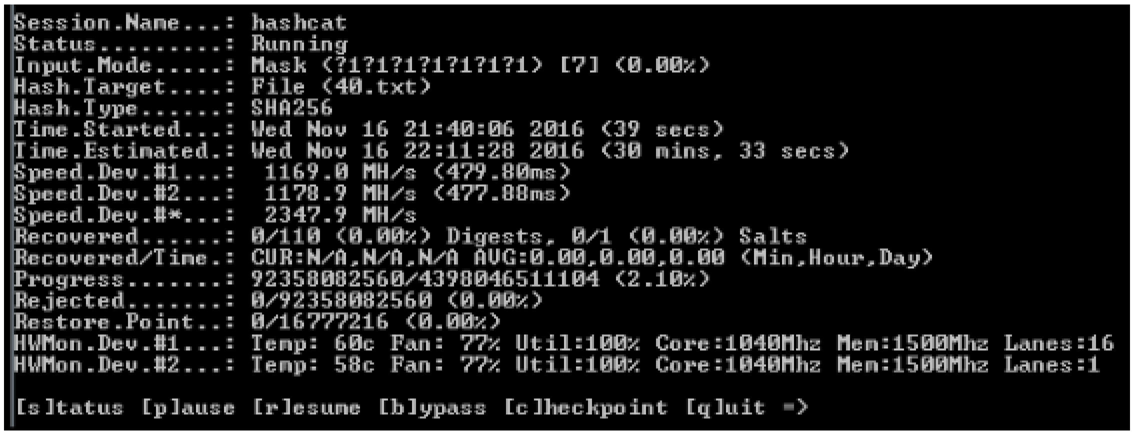


```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1?1) [7] (0.00%)
Hash.Target....: 000182ad92c4533ca7520ce81ec2c348dd1354bc8...
Hash.Type......: SHA256
Time.Started...: Wed Nov 16 20:59:30 2016 (1 min, 20 secs)
Time.Estimated.: Wed Nov 16 21:30:17 2016 (29 mins, 17 secs)
Speed.Dev.#1...:  1193.9 MH/s (481.01ms)
Speed.Dev.#2...:  1200.7 MH/s (478.58ms)
Speed.Dev.#*...:  2394.6 MH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 190475681792/4398046511104 (4.33%)
Rejected.......: 0/190475681792 (0.00%)
Restore.Point..: 0/16777216 (0.00%)
HWMon.Dev.#1...: Temp: 61c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 58c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```



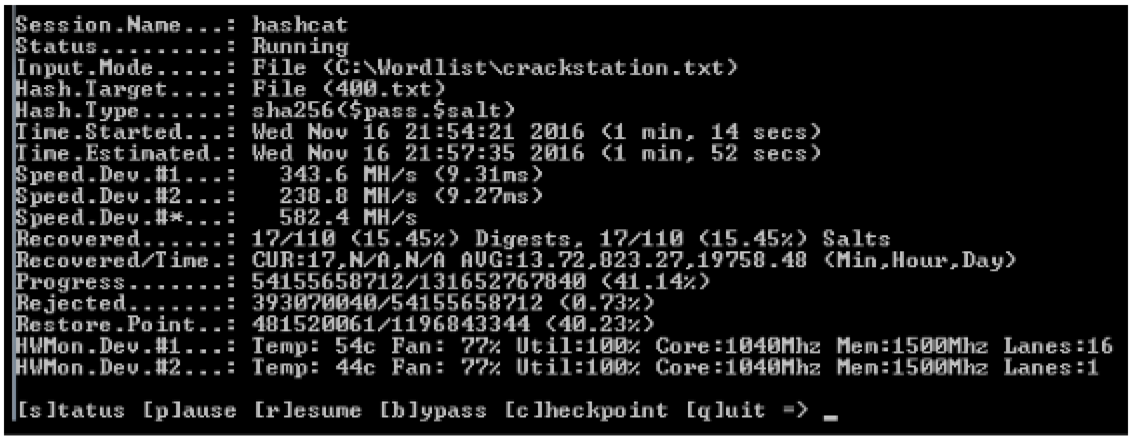
```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: 000182ad92c4533ca7520ce81ec2c348dd1354bc8...
Hash.Type......: SHA256
Time.Started...: Wed Nov 16 20:54:48 2016 (27 secs)
Time.Estimated.: Wed Nov 16 20:57:17 2016 (1 min, 59 secs)
Speed.Dev.#1...:  5899.9 kH/s (8.17ms)
Speed.Dev.#2...:  2511.7 kH/s (8.05ms)
Speed.Dev.#*...:  8411.6 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 195000402/1196843344 (16.29%)
Rejected.......: 2062418/195000402 (1.06%)
Restore.Point..: 178056674/1196843344 (14.88%)
HWMon.Dev.#1...: Temp: 40c Fan: 77% Util:  4% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 38c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

And as you can see it does not do any major difference with a small database with 110 hashes either so that's good

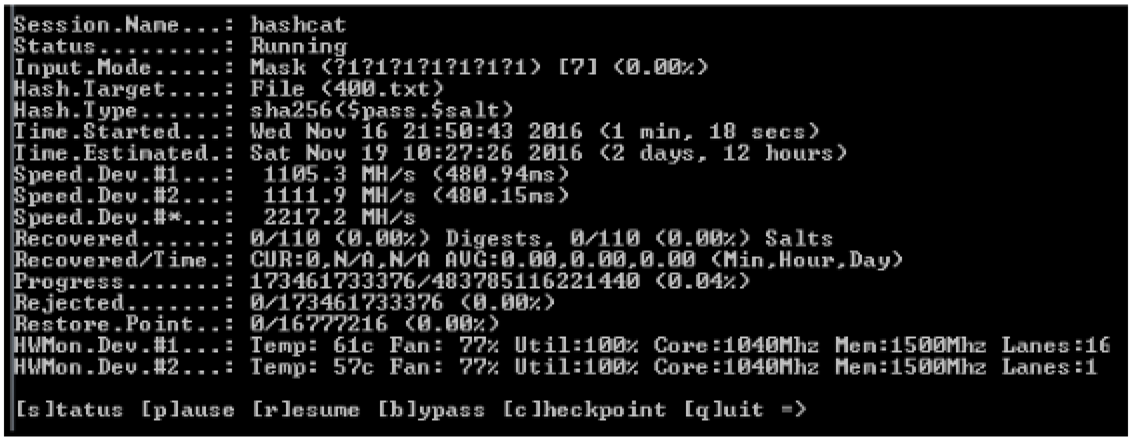


```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1?1) [7] (0.00%)
Hash.Target....: File (40.txt)
Hash.Type......: SHA256
Time.Started...: Wed Nov 16 21:40:06 2016 (39 secs)
Time.Estimated.: Wed Nov 16 22:11:28 2016 (30 mins, 33 secs)
Speed.Dev.#1...:  1169.0 MH/s (479.80ms)
Speed.Dev.#2...:  1178.9 MH/s (477.88ms)
Speed.Dev.#*...:  2347.9 MH/s
Recovered......: 0/110 (0.00%) Digests, 0/1 (0.00%) Salts
Recovered/Time.: CUR:N/A,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 92358082560/4398046511104 (2.10%)
Rejected.......: 0/92358082560 (0.00%)
Restore.Point..: 0/16777216 (0.00%)
HWMon.Dev.#1...: Temp: 60c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 58c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

Now when it comes to salted sha256, it's another the speed still rocks, but it has dropped some, but the speed is quite good.



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: File (400.txt)
Hash.Type......: sha256($pass.$salt)
Time.Started...: Wed Nov 16 21:54:21 2016 (1 min, 14 secs)
Time.Estimated.: Wed Nov 16 21:57:35 2016 (1 min, 52 secs)
Speed.Dev.#1...:   343.6 MH/s (9.31ms)
Speed.Dev.#2...:   238.8 MH/s (9.27ms)
Speed.Dev.#*...:   582.4 MH/s
Recovered......: 17/110 (15.45%) Digests, 17/110 (15.45%) Salts
Recovered/Time.: CUR:17,N/A,N/A AVG:13.72,823.27,19758.48 (Min,Hour,Day)
Progress.......: 54155658712/131652767840 (41.14%)
Rejected.......: 393070040/54155658712 (0.73%)
Restore.Point..: 481520061/1196843344 (40.23%)
HWMon.Dev.#1...: Temp: 54c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 44c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => _
```



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1?1) [7] (0.00%)
Hash.Target....: File (400.txt)
Hash.Type......: sha256($pass.$salt)
Time.Started...: Wed Nov 16 21:50:43 2016 (1 min, 18 secs)
Time.Estimated.: Sat Nov 19 10:27:26 2016 (2 days, 12 hours)
Speed.Dev.#1...:  1105.3 MH/s (480.94ms)
Speed.Dev.#2...:  1111.9 MH/s (480.15ms)
Speed.Dev.#*...:  2217.2 MH/s
Recovered......: 0/110 (0.00%) Digests, 0/110 (0.00%) Salts
Recovered/Time.: CUR:0,N/A,N/A AVG:0.00,0.00,0.00 (Min,Hour,Day)
Progress.......: 173461733376/483785116221440 (0.04%)
Rejected.......: 0/173461733376 (0.00%)
Restore.Point..: 0/16777216 (0.00%)
HWMon.Dev.#1...: Temp: 61c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 57c Fan: 77% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

So running a single SHA256 there are 3 ways

(Sorry the hash is simply too long to write in a row. with the whole command, here an

<u>0047f7e51487c2526e21931e380dd89fce95156cb0e7b7801039d68d89d54715</u>

hashcat64.exe -m 1400 [hash] -a 3 md5.hcmask -w 3
hashcat64.exe -m 1400 [hash] d:\wordlists -w 3
hashcat64.exe -m 1400 [hash] d:\wordlists\pass.txt -w 3

Also don’t forget to add rules, if needed

Or if it’s salted
(This is so long with that I had to shrink it to be able to fit in one

<u>d136c71a26088e7d9afc1d2d1481ba7cc01efaafebff24cefbac450786f65752:4a6e592fa16a426fdb9fc9d0bc7a3500</u>

hashcat64.exe -m -a 3 md5.hcmask -w 3
hashcat64.exe -m d:\wordlists -w 3
hashcat64.exe -m d:\wordlists\pass.txt -w 3

-m 1400 = choosing SHA256
-a 3 md5.hcmask = Attack mode, custom-character set (mask attack) with those masks I provided in the file (file called md5.hcmask)

-w the workload. Running with "-w 3" will give you a couple of thousand more hashes more / but keep an eye at the heat, it will **(-w 4** also possible)
D:\wordlist = a folder or disk where the actual files are located and there's nothing different running with a
\wordlists\pass.txt = a specific wordlist / wordlists

Also running with a list, there's a couple of alternatives, this time I use salted hashes and the only thing as changed are the numbers after "-m"

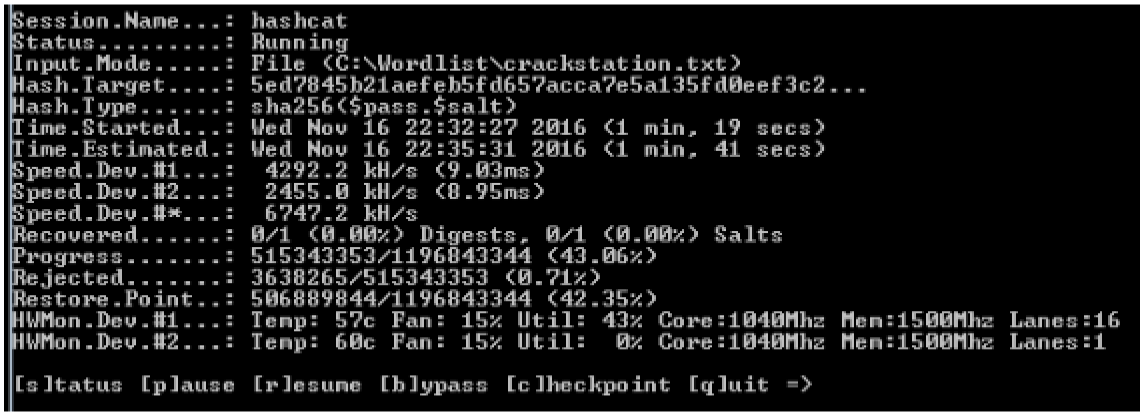
hashcat64.exe -m hash.txt D:\SHA1-MD5
hashcat64.exe -m 1410 hash.txt -a 3 my.hcmask -w 3
-m = choosing salted ($salt.$pass)

Most common Hash types (number)

1400	SHA256
1410	
1420	
1430	
1440	

My recommendations for cracking "Single

hashcat64.exe -m [number] and maybe a salt] d:\wordlists -w 4
Note! I suggest you start with wordlist first, because the speed is okay, wait with masks until you have done all your wordlist



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: 5ed7845b21aefeb5fd657acca7e5a135fd0eef3c2...
Hash.Type......: sha256($pass.$salt)
Time.Started...: Wed Nov 16 22:32:27 2016 (1 min, 19 secs)
Time.Estimated.: Wed Nov 16 22:35:31 2016 (1 min, 41 secs)
Speed.Dev.#1...:  4292.2 kH/s (9.03ms)
Speed.Dev.#2...:  2455.0 kH/s (8.95ms)
Speed.Dev.#*...:  6747.2 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 515343353/1196843344 (43.06%)
Rejected.......: 3638265/515343353 (0.71%)
Restore.Point..: 506889844/1196843344 (42.35%)
HWMon.Dev.#1...: Temp: 57c Fan: 15% Util: 43% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 60c Fan: 15% Util:  0% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

<u>**My recommendations for cracking “file with**</u>

hashcat64.exe -m [nbr] -o [cracked hash [password]
hashcat64.exe -m [nbr] -o [cracked hash -a 3 my.hcmask -w 3

Note! As suggested above start with wordlist before running a mask to save time, because a mask can take a while to go through. Also don’t forget to add some small rules to your wordlist.

WPA/WPA2

What is a handshake or to be more exact the “4-way Handshake”? The 4-way Handshake is a way to calculate the valid WPA2 key for both the client and access point without sending the key itself on the LAN. It would be a major vulnerability if we sent it directly. So how does it work then? Here is an example.

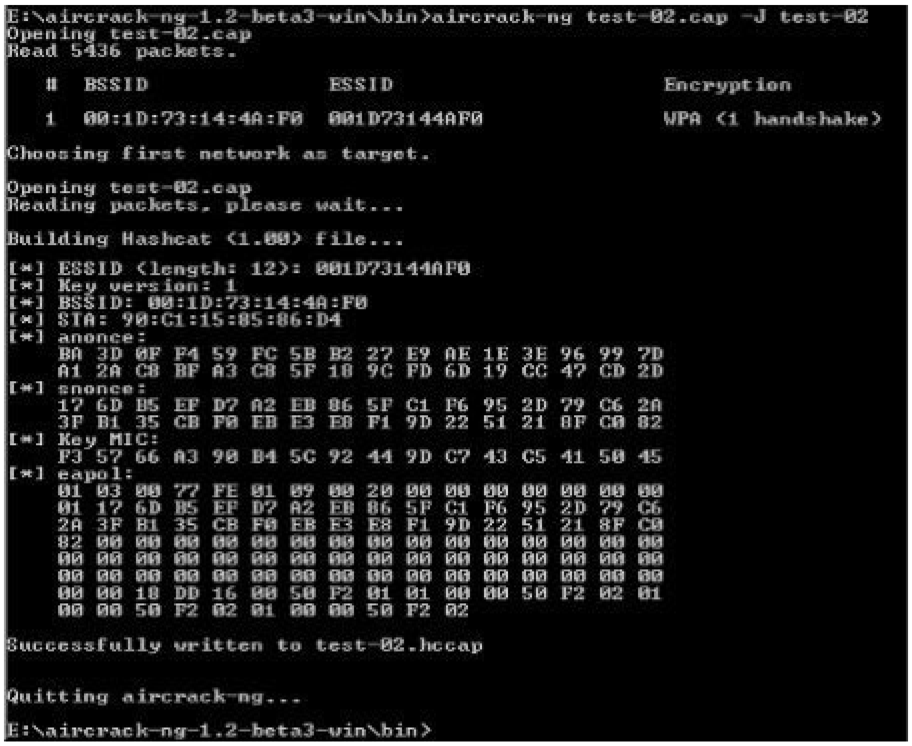
1. The AP sends a value to the Client (EAPOL 1-4) (needed by client to make PTK)
2. The client generates a key and responds back to AP its own random value and as code to verify that value using the value that the AP sent. (EAPOL 2-4)
3. The AP generates a key and if needed it sends back a group key and another verification code (EAPOL 3-4)
4. The Client sends back a message to confirm everything is okay. (EAPOL 4-4)

To crack WPA and WPA2 we need to the handshake in a *.cap file convert it to a *.hccap file Hashcat can read To do this we need at least first and second EAPOL messages or all 4. To convert it from a *.cap, we either can convert it in Linux with the suite or in windows with the same. I will not go through which commands to use to get the handshake for this, but I will tell you how make it possible to run with To do it's easy and we can do it while we are in Linux (Penetration OS like Kali etc), but it's also possible in windows. You just have to

Download it from their site,

http://www.aircrack-ng.org/

We're going to use to convert it to a *.hccap file.



```
E:\aircrack-ng-1.2-beta3-win\bin>aircrack-ng test-02.cap -J test-02
Opening test-02.cap
Read 5436 packets.

   #  BSSID              ESSID                         Encryption

   1  00:1D:73:14:4A:F0  001D73144AF0                  WPA (1 handshake)

Choosing first network as target.

Opening test-02.cap
Reading packets, please wait...

Building Hashcat (1.00) file...

[*] ESSID (length: 12): 001D73144AF0
[*] Key version: 1
[*] BSSID: 00:1D:73:14:4A:F0
[*] STA: 90:C1:15:85:86:D4
[*] anonce:
    BA 3D 0F F4 59 FC 5B B2 27 E9 AE 1E 3E 96 99 7D
    A1 2A C8 BF A3 C8 5F 18 9C FD 6D 19 CC 47 CD 2D
[*] snonce:
    17 6D B5 EF D7 A2 EB 86 5F C1 F6 95 2D 79 C6 2A
    3F B1 35 CB F0 EB E3 E8 F1 9D 22 51 21 8F C0 82
[*] Key MIC:
    F3 57 66 A3 90 B4 5C 92 44 9D C7 43 C5 41 50 45
[*] eapol:
    01 03 00 77 FE 01 09 00 20 00 00 00 00 00 00 00
    01 17 6D B5 EF D7 A2 EB 86 5F C1 F6 95 2D 79 C6
    2A 3F B1 35 CB F0 EB E3 E8 F1 9D 22 51 21 8F C0
    82 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 18 DD 16 00 50 F2 01 01 00 00 50 F2 02 01
    00 00 50 F2 02 01 00 00 50 F2 02

Successfully written to test-02.hccap


Quitting aircrack-ng...

E:\aircrack-ng-1.2-beta3-win\bin>
```

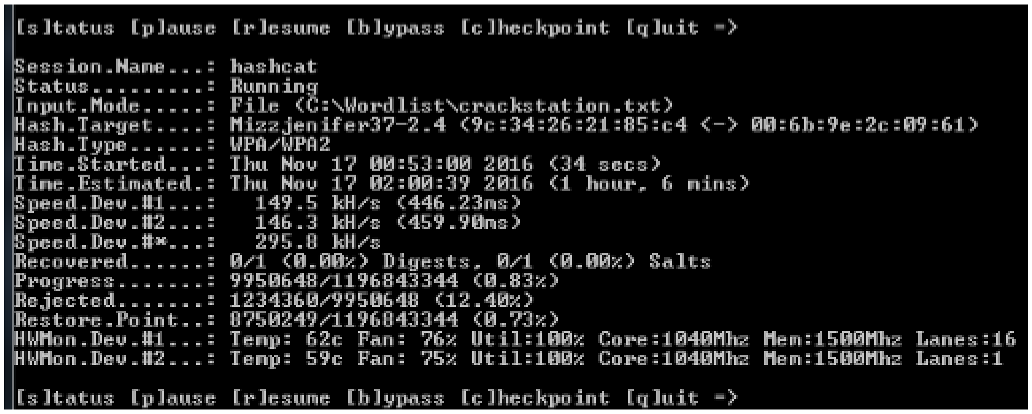
To create an *.hccap file from a *.cap file as the picture above shows, browse the exe files that's located in folder. Here you write

```
aircrack-ng capfile.cap –J your-hccap-file
```

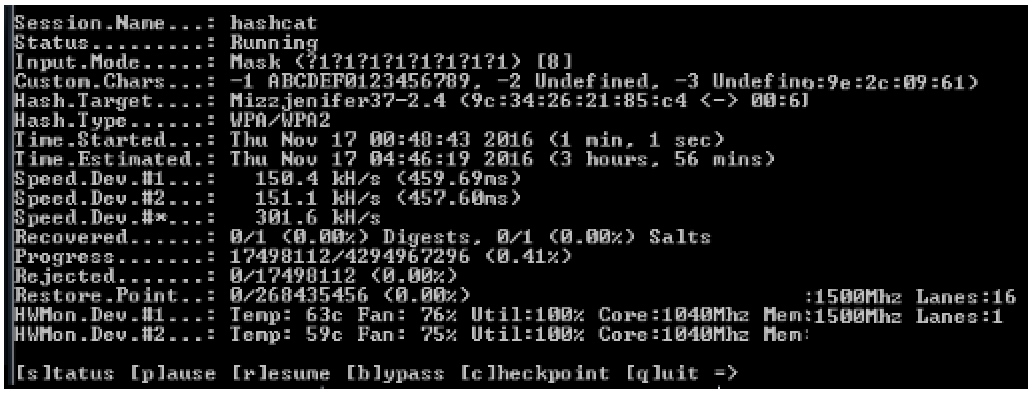
As you see it's not that hard to convert, and I did it in Windows as well. Now were going to start and run our handshake. Here we can't use any types of wordlist. All words below 8 characters will simply be except if you using rules that together will make each word greater than 8 characters of course. we have to use specialized masks or wordlist. Or at least I do for I already have a bunch of those because I'm a WiFi hacker as well. it's time to the dumps 8 in Hashcat will ignore all words below 8 characters anyway, but I want my I don't know if performance are affected or not with a file containing lesser words than 8 in but I'm not taking any chances. This is one of few times both mask and wordlist are equally fast. And as you see the speed is horrible. It's still 15-20 times faster compared to a normal CPU at least! Now a faster Graphics card or a third equivalent might double that or triple that speed depending on which graphics card we choose. Also, we can run more than 1 handshake at the same time; however it will affect "estimated cracking time" badly. if you got the GPU power to make up for or you want to try it out, how you do it in Windows

copy *.hccap

All *.hccap files will copied in to one file, after that you run normal cracking



```
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>

Session.Name...: hashcat
Status.........: Running
Input.Mode.....: File (C:\Wordlist\crackstation.txt)
Hash.Target....: Mizzjenifer37-2.4 (9c:34:26:21:85:c4 <-> 00:6b:9e:2c:09:61)
Hash.Type......: WPA/WPA2
Time.Started...: Thu Nov 17 00:53:00 2016 (34 secs)
Time.Estimated.: Thu Nov 17 02:00:39 2016 (1 hour, 6 mins)
Speed.Dev.#1...:   149.5 kH/s (446.23ms)
Speed.Dev.#2...:   146.3 kH/s (459.90ms)
Speed.Dev.#*...:   295.8 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 9950648/1196843344 (0.83%)
Rejected.......: 1234360/9950648 (12.40%)
Restore.Point..: 8750249/1196843344 (0.73%)
HWMon.Dev.#1...: Temp: 62c Fan: 76% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:16
HWMon.Dev.#2...: Temp: 59c Fan: 75% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```



```
Session.Name...: hashcat
Status.........: Running
Input.Mode.....: Mask (?1?1?1?1?1?1?1?1) [8]
Custom.Chars...: -1 ABCDEF0123456789, -2 Undefined, -3 Undefino:9e:2c:09:61)
Hash.Target....: Mizzjenifer37-2.4 (9c:34:26:21:85:c4 <-> 00:6]
Hash.Type......: WPA/WPA2
Time.Started...: Thu Nov 17 00:48:43 2016 (1 min, 1 sec)
Time.Estimated.: Thu Nov 17 04:46:19 2016 (3 hours, 56 mins)
Speed.Dev.#1...:   150.4 kH/s (459.69ms)
Speed.Dev.#2...:   151.1 kH/s (457.60ms)
Speed.Dev.#*...:   301.6 kH/s
Recovered......: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.......: 17498112/4294967296 (0.41%)
Rejected.......: 0/17498112 (0.00%)
Restore.Point..: 0/268435456 (0.00%)                    :1500Mhz Lanes:16
HWMon.Dev.#1...: Temp: 63c Fan: 76% Util:100% Core:1040Mhz Mem:1500Mhz Lanes:1
HWMon.Dev.#2...: Temp: 59c Fan: 75% Util:100% Core:1040Mhz Mem:

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

(Above picture is running with <u>only ONE handshake </u>in *. hccap file)

So running <u>WPA/WPA2 </u>there's 2 ways

hashcat64.exe -m 2500 test-01.hccap -a 3 md5.hcmask -w 3
hashcat64.exe -m 2500 test-01.hccap d:\word -w 3

-m 2500 = choosing WPA/WPA2
test-01.hccap = a single hash
-a 3 md5.hcmask = Attack mode, custom-character set (mask attack) with those masks I provided in the file (file called md5.hcmask)
-w 3 = the workload putting in cracking. –“w 2” is normal. Running with “-w 3” will give you a couple of thousand more hashes every but keep an eye at the heat, it will arise (as said before you could use **-w 4** as well)

D:\word =a folder, disk or a specific file

Most common WPA/WPA2 Hash types (number)

2500 = WPA/WPA2

My recommendations for cracking WPA/WPA2 “Single

hashcat64.exe -m 2500 [HCCAP-file] d:\wordlists -w 3
This is actually the only thing worth running, unless you have multiple graphics Or a speed over 100.000 k/s

-m 2500 [HCCP-file] -a 3 md5.hcmask -w 3
The only time I would be fine running a brute force mask attack is if we got a real powerful graphic card that can

produce 100.000k/s or better (or more graphics cards)

A small list of hashes

Here I'm leaving a short list on various attack modules. To use this just change the module number number, nothing can be more So **hashcat64.exe -m [number]** However a few of the hashes needs some kind of before working with and sorry I go through them, but there's a lot of tutorials out there how you get the special hash out of things as wallets etc.

Short List

0 MD5
10 MD5($pass.$salt)
20 MD5($salt.$pass)
23 Skype
30 MD5(unicode($pass).$salt)
40 MD5($salt.unicode($pass))
100 SHA1
110 SHA1($pass.$salt)
120 SHA1($salt.$pass)
122 OS X v10.4, v10.5, v10.6
130 SHA1(unicode($pass).$salt)
140 SHA1($salt.unicode($pass))
300 MySQL4.1 / SQL5

400
1000 NTLM
1400 SHA256
1410 SHA256($pass.$salt)
1420 SHA256($salt.$pass)
1700 SHA512
1710 SHA512($pass.$salt)

1720 SHA512($salt.$pass)
2500 WPA/ WPA2
2611 vBulletin < v3.8.5
2711 vBulletin >= v3.8.5
2811 IPB2+, MyBB1.2+
9400 Office 2007
9500 Office 2010
9600 Office 2013
11300 Bitcoin wallet
116007-Zip
12500RAR3-hp
13000RAR5
13600WinZip
13800Windows 8+ phone PIN/Password

As usual at the end of this book I got a couple of persons that need to be recognized that have influenced me in different ways through the journey it took to write this This book has been a long way from straight forward, and half way in the book the programmers' changed the layout and the name of the program, which meant that I had to start all over again, but that's okay, the book is finally done, and I can look for a new project to write about. So let's go and give a number of people some credit shall we.

To wonderful wife.

Thank you so much for you believes in me, and let me sit and play with my computers

Chick3nman

Very close to a mentor as possible. Had patience with all questions I at him

"Martin" a.k.a that donated a new motherboard to me when my power supply exploded, in case my motherboard was That's what a real friend Thanks for all help and all support man!

Armada
Yeah that guy again, :) Because without his help once a time, this book wouldn't be possible

mObile
That helped me getting cheaper hardware in a He knows what I mean Thank you

CyberGuard
A guy with knowledge

And special greetings to all members in **Hash Cracking Slashes** the guys

This was all I had thought to write about in this book, I MIGHT write a second book about this subject in the future, I haven't decided yet. It depends on the demand for the book of course