**Password Cracking**

**The art of raking a password gash and learinign the plain text**

* Tools like John the Ripper and Hashcat
* Based on time-complexity

**Types of cracking**

* Dictionary
* Brute force
* Rainbow tables

**Dictionary**

Loads a file that contains plain text passwords on each line

* Also known as a wordlist

Hashes each word in the wordlist and compares to the original hash

* Not guaranteed to find the plain text

Can mutate a wordlist/generate your own per assessment

**Brute Force**

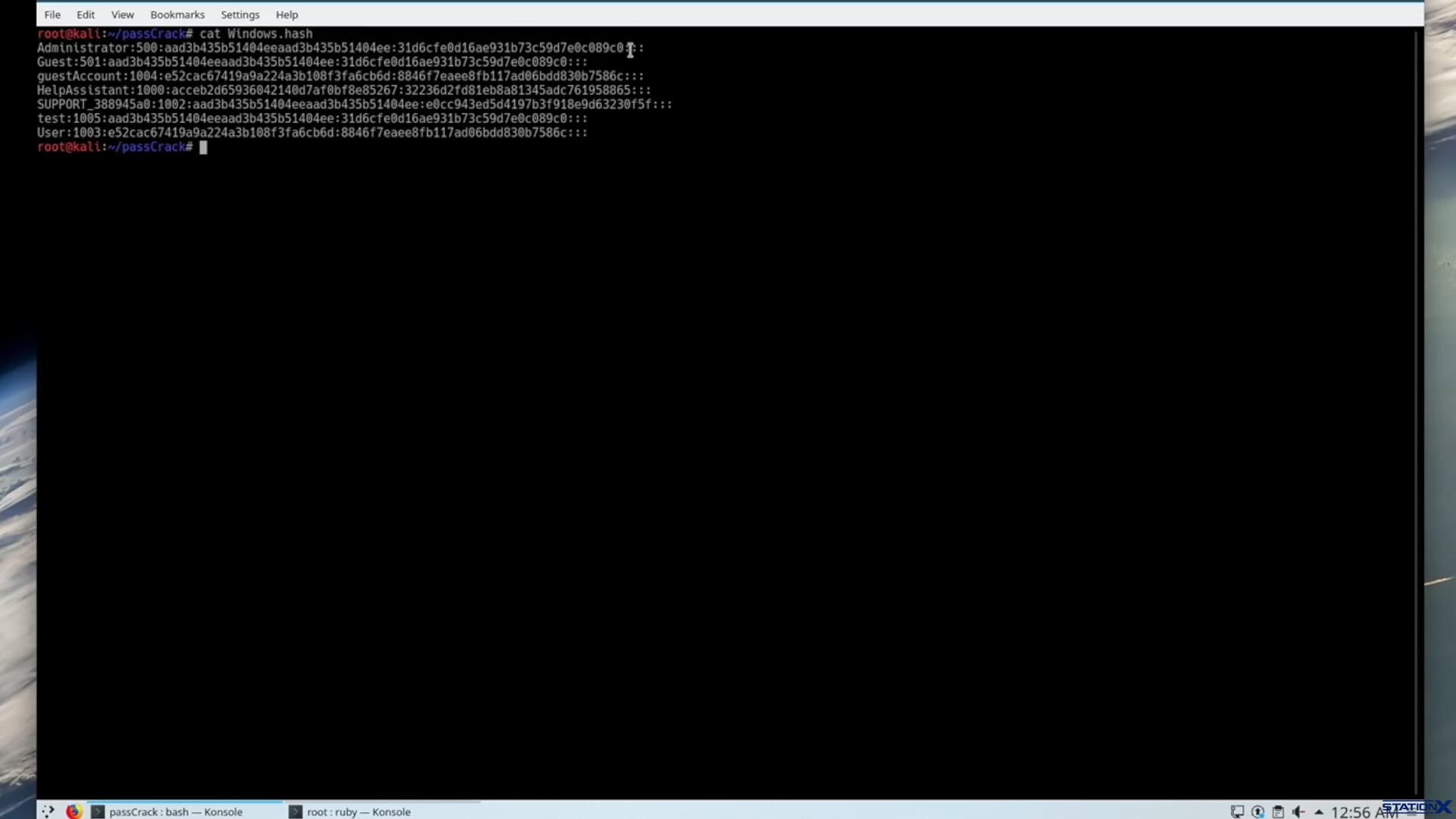
Takes every combination of every letter/digit/symbol for a specific length

* Each time the string is changed, it rehashes and checks it against the original hash
* Highly memory intensive

Guaranteed to find the plaintext of any password

* Depending on the complexity of the password, the cracking process could take months to years to decades or even thousands of years

**John the Ripper**



**Once hashes are grabbed in this format, with the NTLM:NT hashes, then they can be put into john the ripper and worked on**

John --wordlist=/usr/share/john/password.list windows.hash

* Windows.hash is our hashes file

**Linux**

Have to get the shadow file and the passwd file

* If there are x’s where the hashes should be in the passwd file, then they’ll be in the shadow file

Cat etc/passwd > passwd

* Create a copy of these files in the passcrack dir
* Then combine the two files
  + Unshadow passwd shadow > linux.hash
  + Cat linux.hash

John --wordlist=/usr/share/john/password.list linux.hash

**Hashcat**

John the ripper is good for quick passes and a little bit of brute forcing but the most speed and power comes from Hashcat

* Can use CPUs and GPUs to process everything
* The NT hashes is what we use

A computer screen capture

Description automatically generated with medium confidence

Hashcat -m 1000 -a 0 -o win\_final --remove Win-Hashcat.hash /usr/share/john/password.list --force

* -m # specifies the mode
* 1000 # specifies the hash type
* -a # specify the type of action we want i.e. dictionary
* 0 # action type – 0 is dictionary
* -o # output the file to win\_final
* --remove Win\_Hashcat.hash # remove cracked passwords into a new file so that you don’t have to work out which ones have and haven’t been cracked manually
* Path # points to wordlist path
* --force # must do on a VM, not so much on a physical machine

Text

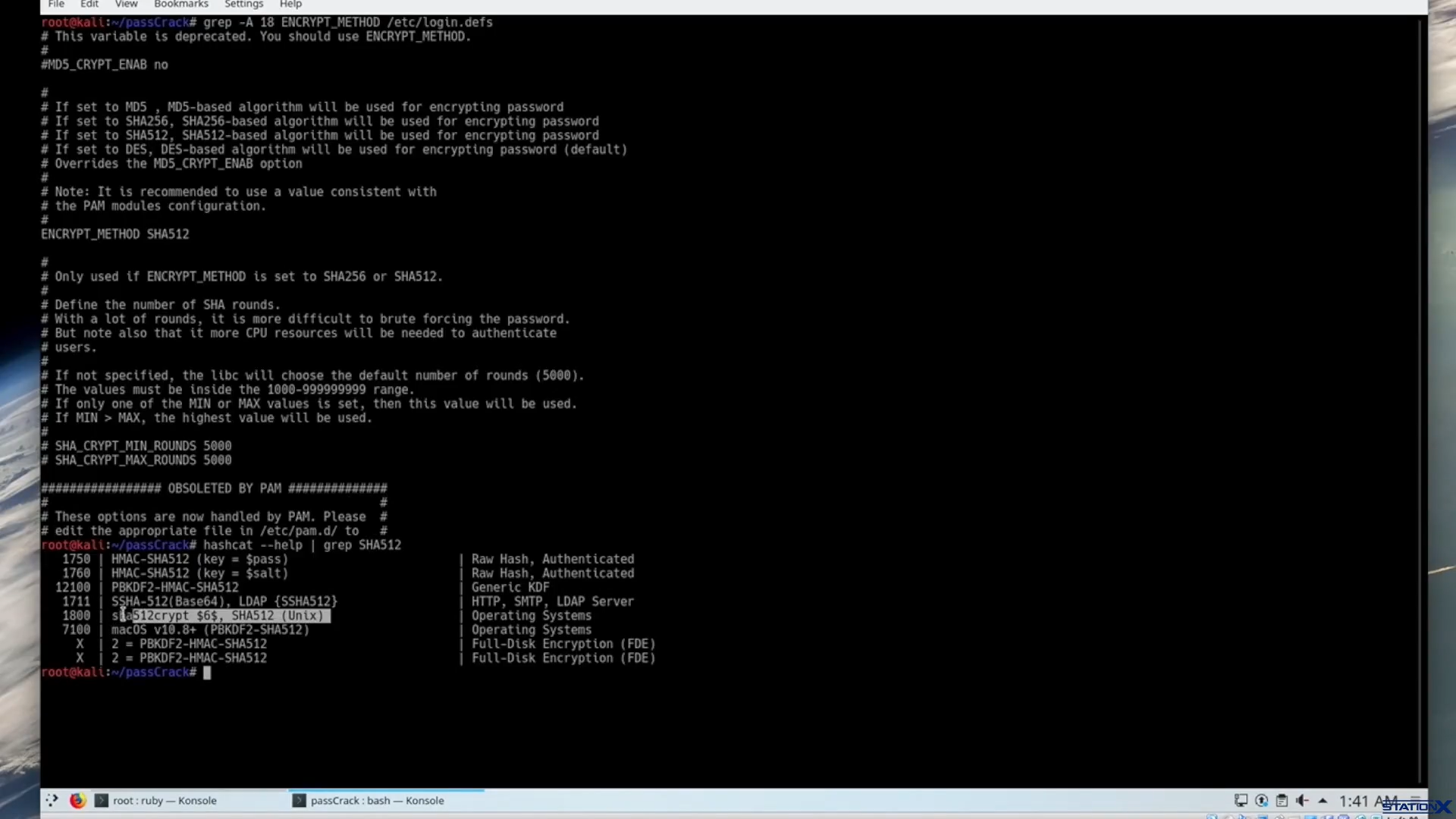
Description automatically generated

Once ran, we can see that the win\_final file has the cracked hash and its plaintext version

**Linux**

Linux is a bit different as different Linux envs use different hashing algorithms

* Grep -A 18 ENCRYPT\_METHOD /etc/login.defs
  + # to find the encryption algorithm used in this env



**‘ENCRYPT\_METHOD SHA512’ – is what were looking for (bottom of first 3rd of page)**

Then using commands, we will see what options hashcat has for us for that encryption algortithm

* Hashcat --help | grep SHA512
* 1800 is what were looking for

A computer screen capture

Description automatically generated with medium confidence

The Linux\_Hashcat.hash file simply has the admin hash pulled and saved separately into it

Run the same command just with this file

**Hashcat -m 1800 -a 0 -o lin\_final Linux\_Hashcat.hash /usr/share/john/password.list --force** Sometimes using the --remove command can cause it to fail

**Rainbow Tables**

**Large database of password hashes and their plaintext counterpart**

* Can be hundreds of gigabytes in size

Reduces memory requirements on the disk space

* Compromises runtime for memory requirement (takes longer)

Traditional brute forcing is very slow and hard to crack anything past 5-6 chars

* Hashes every time a new string is generated
* Rainbow tables makes it easier to have pre-hashed plaintext to avoid re hashing every string

**Rainbow tables use Reduction to create chains (tables) of hashes**

* Each chain will be looked at until the plaintext is found

**How reduction works**

We start with an initial MD5 HASH “25s555tgy66hwsu7782h7tg”

Then we hit the first chain and grab the first 8 chars and rehash it

* Will continue this rehashing of the first 8 chars until stopped by the user
* This is how a table/chain is built

Diagram

Description automatically generated

**HW –**

Graphical user interface, text, application, letter, email

Description automatically generated

**ALSO, repeat the Hashcat and John the Ripper process yourself**