

PWNDEFEND

HACKING 101

HASH CRACKING

Version 1.0

Welcome

Welcome to the third short course in our hacking 101 series for pwnDefend! The intention behind this is to give people new to the security testing/hacking world some basic information on common hashes and methods to break them!

This series has been created by Daniel Card (UK_Daniel_Card) and is aimed to educate and help people with some basic tools, techniques and practises.

Please only use this information for good! Ensure you use these tools where you either own the assets or have the owners permission! Check local laws and don't break them! #hack4good

Introduction

This course aims to teach the basics of hash cracking!

By the end of the course you should be familiar with common types of hashes for common file types and operating systems. You should also have a basic understanding of methods and tools that can be used to crack them.

Cracking hashes relies on a combination of access, resources (tools, wordlists, hardware) and time. Not everything will crack so it's a good idea to work on low hanging fruit and take a risk based approach before investing huge resource levels on cracking a hash.

Learning Goals

- Understand what a HASH is
- Understand common hash types
- Understand Windows and Linux hash types
- Understand different attack methods
- Be able to conduct basic attacks on common hashes

Course Requirements

Whilst you should be able to follow the content here without a huge level of experience it's advisable to have some knowledge in the following areas:

- OS fundamentals LINUX and WINDOWS
- Basic TCP/IP Network knowledge
- Experience using a hypervisor such as ORACLE Virtual Box, Hyper-V, VMware player etc.
- Be able to use a web browser and have an internet connection
- Ideally be able to access a GPU (I've made the exercises so they work with CPU cracking only)



Why we care?

- Sensitive data, such as credentials are often not stored in plain text.
- By leveraging hashes we as a defender can limit the ability of an unauthorised user from accessing sensitive data. We can also protect against sending credentials in clear text across a network.

Situation

- We have obtained access to a range of data and systems (user generated file data and Operating system credentials storage files)
- We need to crack the hashes to gain access to the sensitive data

KALI Keyboard Languages

Whilst writing this course I decided to use KALI LIVE. I'll be honest I very rarely use this mode, I mainly work from installed VM's. Sometimes I will use a USB key to demo an attack but almost everything I do is from install. One thing I noticed is the keyboard layout defaults to US.

So if you need to switch keyboard layout from the command line try this:

setxkbmap -query

setxkbmap gb

```
root@kali:/etc# setxkbmap -query
rules:      evdev
model:      pc105
layout:     us
root@kali:/etc# setxkbmap gb
root@kali:/etc# setxkbmap -query
rules:      evdev
model:      pc105
layout:     gb
root@kali:/etc#
```


PWNDEFEND

A woman with long, wavy, reddish-brown hair is sitting at a desk in a dimly lit room. She is wearing a black beanie with a silver band and a grey hoodie. She is looking directly at the camera with a serious expression. In front of her is a large computer monitor. To her left, there are other monitors displaying code and data. The room has a blueish tint, and the overall atmosphere is that of a hacker's den or a server room.

Module 1

HASHING overview

What is a HASH?

“A cryptographic hash function is a special class of hash function that has certain properties which make it suitable for use in cryptography. It is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size (a hash) and is designed to be a one-way function, that is, a function which is infeasible to invert”

https://en.wikipedia.org/wiki/Cryptographic_hash_function

Got that? it's a way of changing a string (e.g. “Password”) into a fixed length HASHED string, which can't be reversed (unlike say a ROT13 cipher or BASE64 encoding).

Where are hashes used?

Hashes are often used to protect sensitive data such as credentials.

In Windows, hashes are used to protect credentials on the SYSTEM (SAM/SYSTEM registry hives) and in transit. They are an integral part of the authentication process.

Hash Theft Methods

	Online	Offline
Windows	<ul style="list-style-type: none">• LLMNR• Compromised SMB Server• RAM Access• Process Dump• SAM/SYSTEM Hive Theft• Browser Cache• Credential Manager/Windows Logon Cache• File Theft	<ul style="list-style-type: none">• Direct Disk Access• Direct Memory Access
Linux		<ul style="list-style-type: none">• Direct Disk Access• Direct Memory Access

PWNDEFEND

Module 2

Windows OS Hashes

Windows OS Hashes

There are a range of common HASHES used in the Windows operating system. The main areas you are likely going to need to know about are as follows:

- Lan Manager (LM) FORMAT
- NTLM (NT HASH)
- NTLMv2 (NTLM)
- msCACHEDv2 HASH FORMAT

For some further reading on the subject see the following:

https://en.wikipedia.org/wiki/LAN_Manager

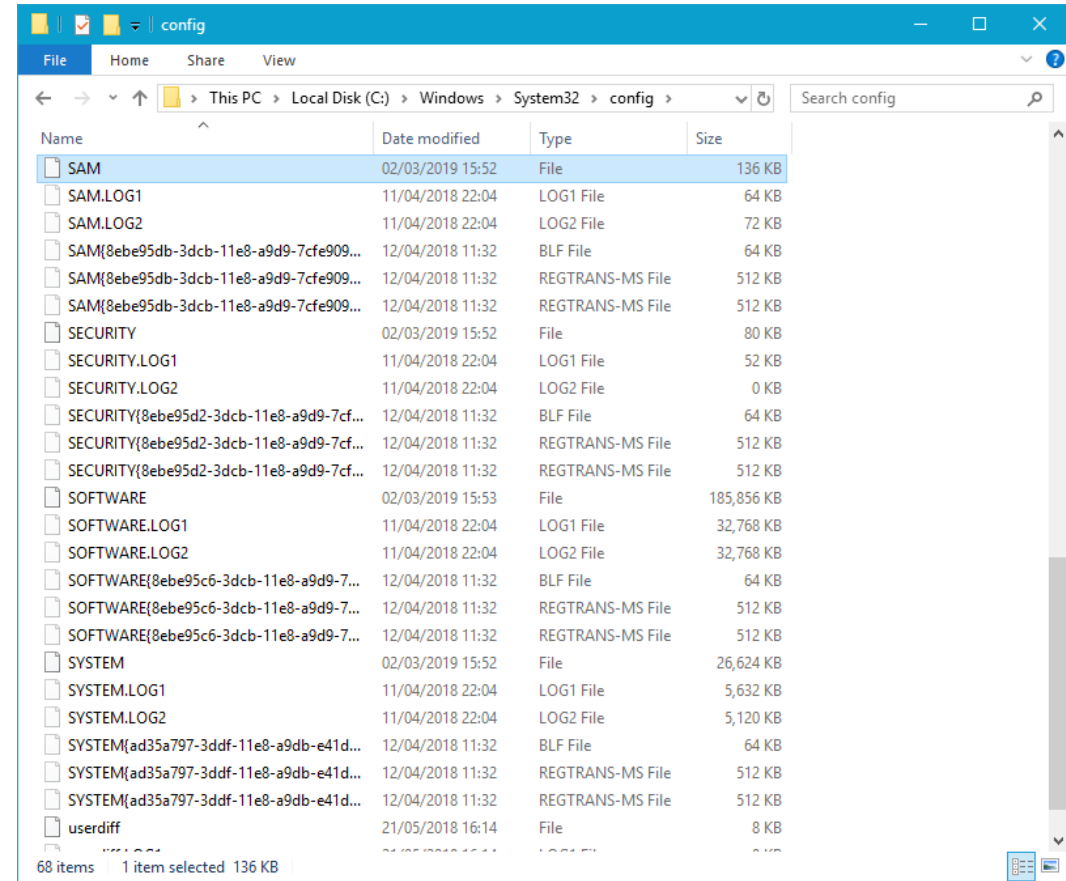
Windows System HASHES

- Windows system hashes are derived from the following two files:
 - SAM
 - SYSTEM

The default location for these files is:

c:\windows\system32\config

Please note that this is not the case for active directory passwords



LM HASH

The Lan Manager (LM) hash is a legacy hash which suffers from severe weaknesses. It came from Windows NT 3.1 (1993) and is still sometimes used for legacy system support, though normally NTLMv2 is used!

Some common weaknesses are as follows:

- They are restricted to a maximum of 14 characters (a password with longer than 14 will not have an LM hash on modern Windows systems)
- After 7 characters the clear text is split into two and each section is hashed (drastically reducing the key space)
- The HASH is NOT SALTED

From Windows Vista/Server 2008 these were disabled by default

I'll have SALT with that!

“In cryptography, a salt is random data that is used as an additional input to a one-way function that “hashes” data, a password or passphrase. Salts are used to safeguard passwords in storage. Historically a password was stored in plaintext on a system, but over time additional safeguards developed to protect a user's password against being read from the system. A salt is one of those methods.

A new salt is randomly generated for each password. In a typical setting, the salt and the password (or its version after Key stretching) are concatenated and processed with a cryptographic hash function, and the resulting output (but not the original password) is stored with the salt in a database. Hashing allows for later authentication without keeping and therefore risking the plaintext password in the event that the authentication data store is compromised.

Salts defend against dictionary attacks or against their hashed equivalent, a pre-computed rainbow table attack. Since salts do not have to be memorized by humans they can make the size of the rainbow table required for a successful attack prohibitively large without placing a burden on the users. Since salts are different in each case, they also protect commonly used passwords, or those users who use the same password on several sites, by making all salted hash instances for the same password different from each other.”

[https://en.wikipedia.org/wiki/Salt \(cryptography\)](https://en.wikipedia.org/wiki/Salt_(cryptography))

NTHASH/NTLM/NTLMv2

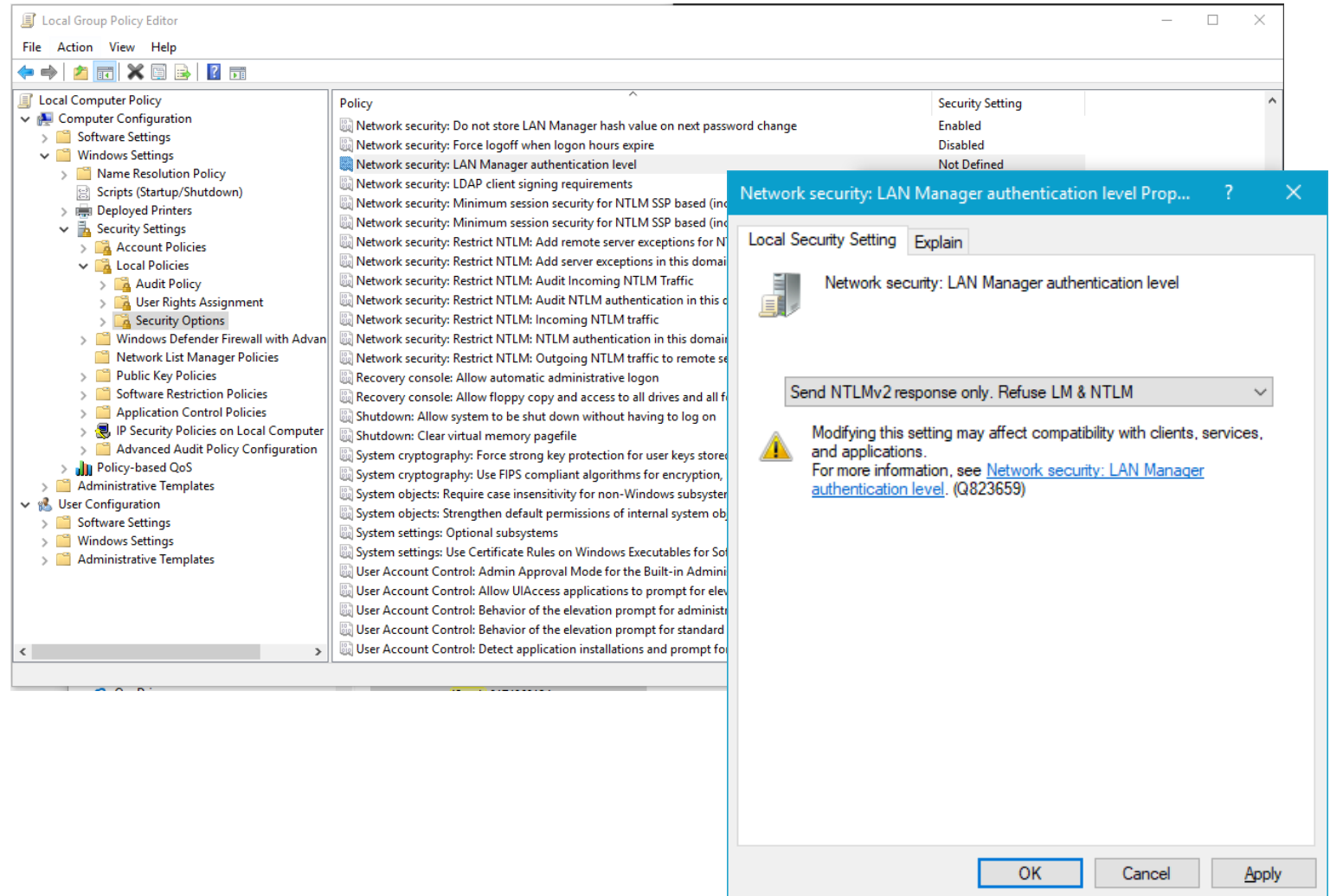
Introduced in Windows NT 4 SP4 it was designed as a Cryptographically stronger replacement for NTLMv1. However from a defender and attacker point of view its important to understand:

- NT HASHES can still be cracked
- NTLMv2 HASHES can be passed

Configuring a Windows Client

You can configure LAN Manager Authentication level using local policy of group policy configuration.

Ideally you want to disable LM and NTLMv1



Attacks

There are a number of ways a malicious actor can obtain a hash these include:

- Stealing the SAM and SYSTEM files
- DUMPING RAM Online (e.g. using Mimi Katz)
- Dumping process RAM (e.g. process explorer)
- Extract hibernation or page file data
- Capturing HASHES in transit (e.g. responder attacks)
- Cloning a hard drive

Hash Types

John is often simpler to use as it attempts to detect the hash type however if you want to GPU enable your cracking efforts then Hashcat is your best friend!

All the different types of hash types and masks can get confusing so we've got two good resources here. Firstly a tool to help identify the hash type:

<https://tools.kali.org/password-attacks/hash-identifier>

and secondly a lookup table:

https://hashcat.net/wiki/doku.php?id=example_hashes

LM/NTLM Cracking Cheat Sheet

LM Hash Crack

```
john --format=lm hash.txt
```

```
hashcat -m 3000 -a 3 hash.txt
```

NT Hash Crack

```
john --format=nt hash.txt
```

```
hashcat -m 1000 -a 3 hash.txt
```

NTLMv1

```
john --format=netntlm hash.txt
```

```
hashcat -m 5500 -a 3 hash.txt
```

NTLMv2

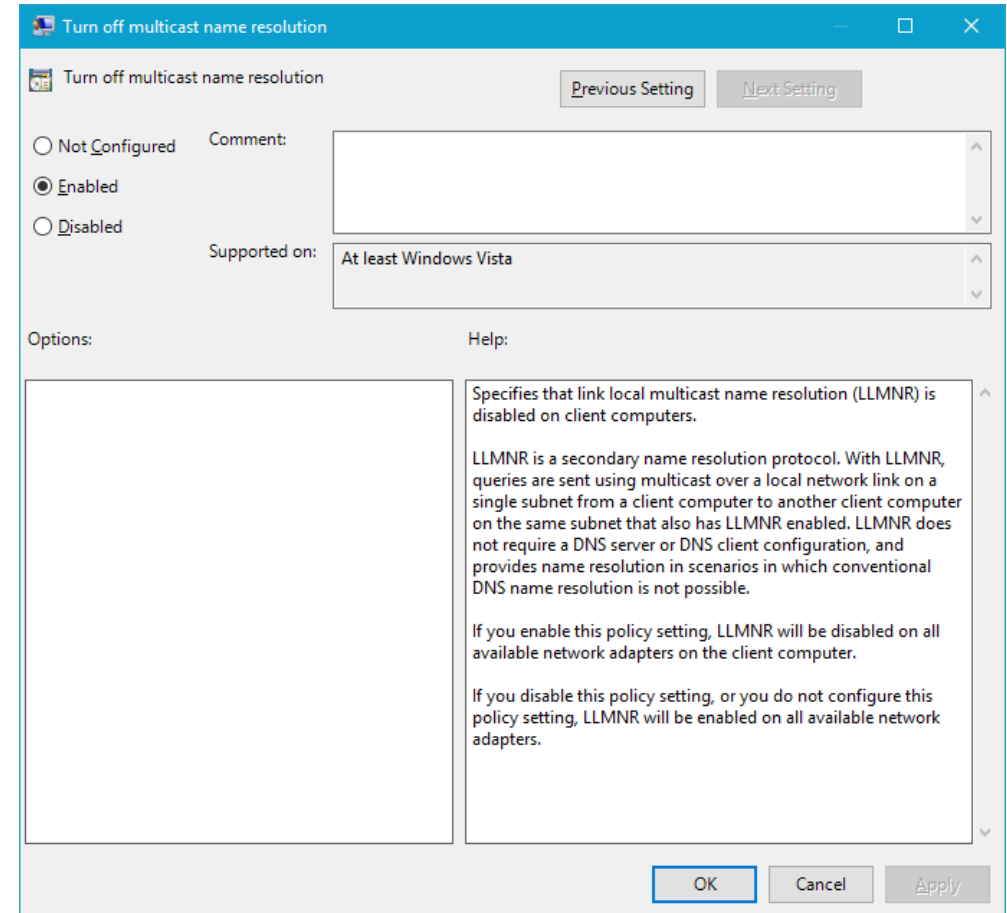
```
john --format=netntlmv2 hash.txt
```

```
hashcat -m 5600 -a 3 hash.txt
```

Defending Against Responder Attacks

To defend against responder attacks we have a number of methods, ideally you want to layer these (use all of them!)

- Disable LMNR Multi-cast name resolution
- Disable NetBIOS (either locally or via DHCP)
- Restrict SMB access (disable outbound SMB on perimeter firewall, segment internal networks and harden host based firewall configurations)
- Leverage administrator “jump boxes”
- Reduce user account access rights



Modern OS Defences

Windows 10/Server 2019 comes with a range of defences to protect it's hashes such as:

- Bit locker (though there are now techniques to bypass some configs of this)
- User Account Control
- Credential Guard
- Windows Defender

PWNDEFEND

Module 3

Offline Windows and Active Directory Attacks

Kali – List Disks

fdisk -l

cfdisk

parted -l

df -h

lsblk

blkid

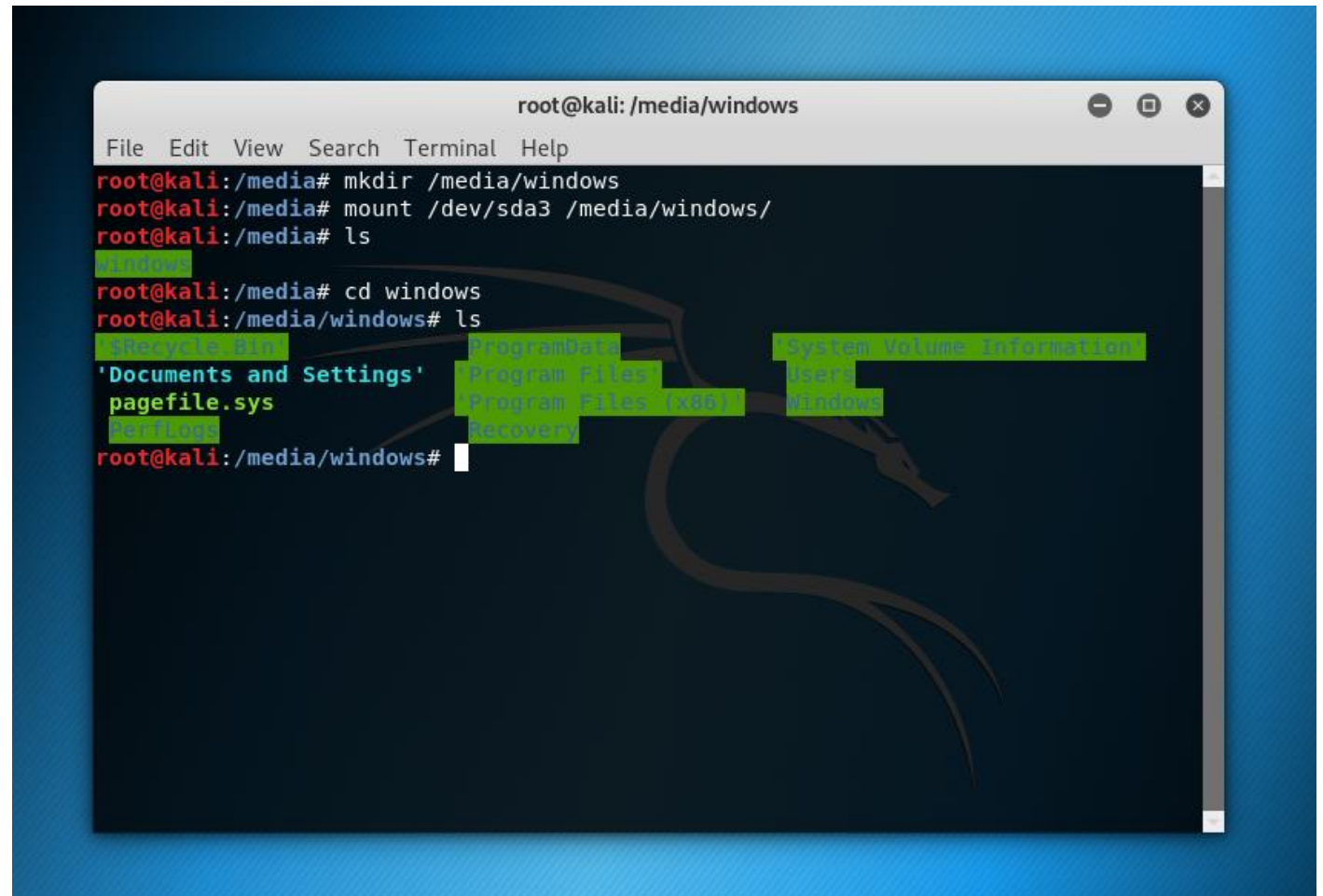
```
root@kali: /media
File Edit View Search Terminal Help
root@kali:/media# fdisk -l
Disk /dev/sda: 60 GiB, 64424509440 bytes, 125829120 sectors
Disk model: VMware Virtual S
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: D484A279-2C39-4F4A-A957-7D6A4DABC277

Device        Start      End    Sectors  Size Type
/dev/sda1     2048     411647    409600   200M EFI System
/dev/sda2    411648     673791    262144   128M Microsoft reserved
/dev/sda3    673792  125827071 125153280 59.7G Microsoft basic data

Disk /dev/loop0: 3 GiB, 3148537856 bytes, 6149488 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
root@kali:/media#
```

Mounting a Drive

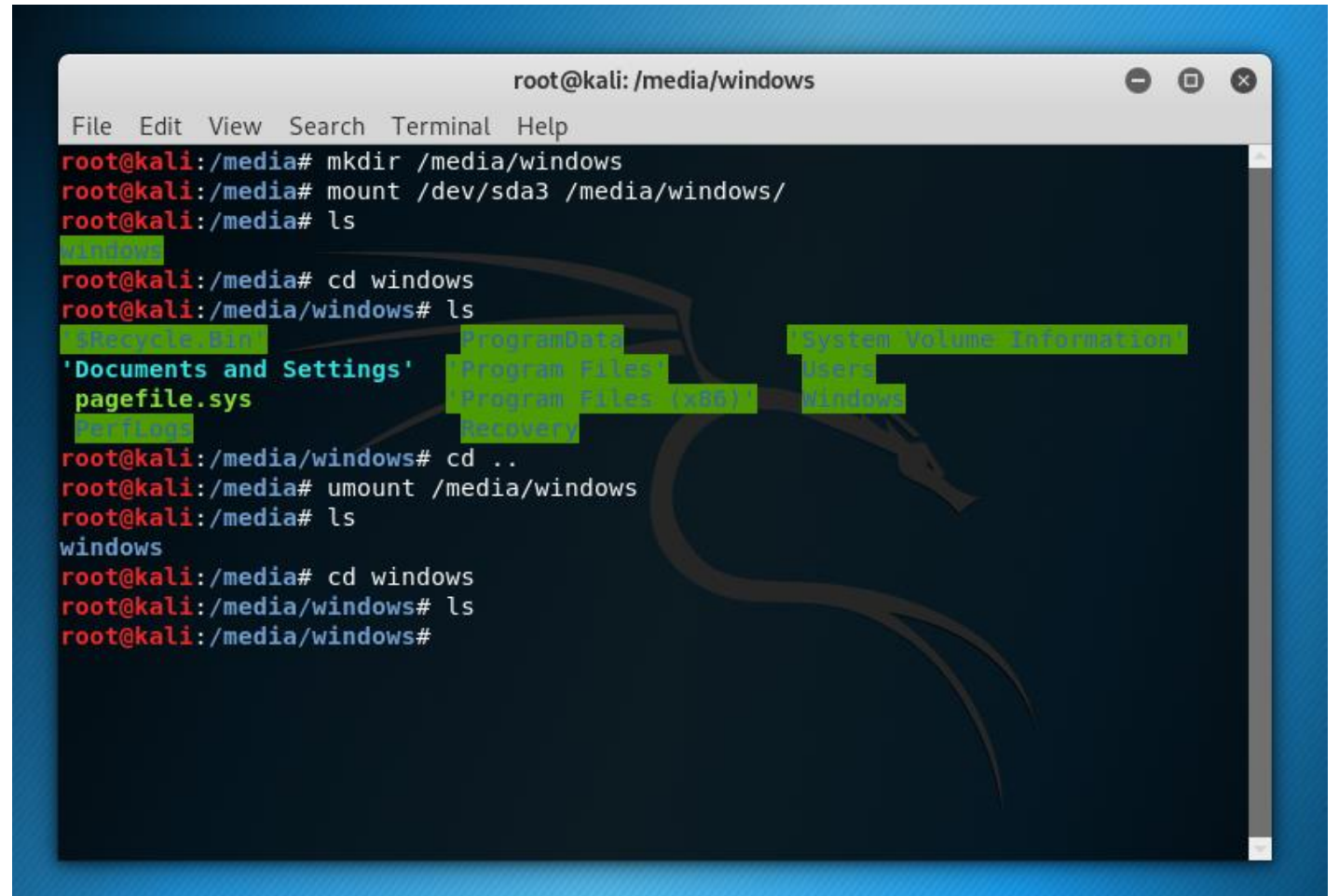
- make a new directory under media
- mount the drive
- list the contents

A terminal window titled 'root@kali: /media/windows' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
root@kali:/media# mkdir /media/windows
root@kali:/media# mount /dev/sda3 /media/windows/
root@kali:/media# ls
windows
root@kali:/media# cd windows
root@kali:/media/windows# ls
'Recycle Bin'          ProgramData             'System Volume Information'
'Documents and Settings' Program Files            Users
pagefile.sys           'Program Files (x86)'  windows
perflogs               Recovery
```


unMounting a Drive

- unmount the drive using umount followed by the mount path

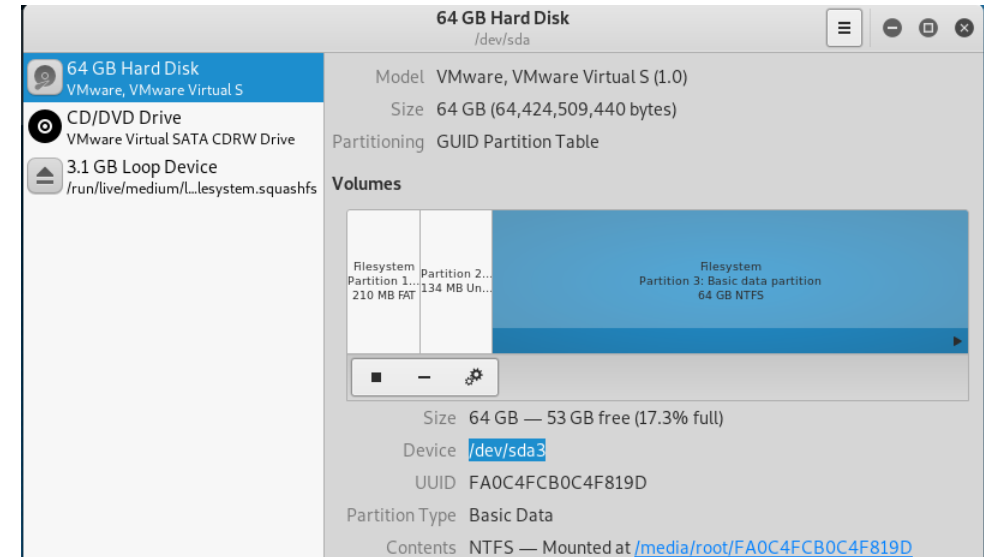


```
root@kali: /media/windows
File Edit View Search Terminal Help
root@kali:/media# mkdir /media/windows
root@kali:/media# mount /dev/sda3 /media/windows/
root@kali:/media# ls
windows
root@kali:/media# cd windows
root@kali:/media/windows# ls
'GRACycle.BIN'      ProgramData      'System Volume Information'
'Documents and Settings'  Program Files    Users
pagefile.sys        Program Files (x86)  Windows
Partitions          Recovery
root@kali:/media/windows# cd ..
root@kali:/media# umount /media/windows
root@kali:/media# ls
windows
root@kali:/media# cd windows
root@kali:/media/windows# ls
root@kali:/media/windows#
```

Hash Dump Demo (RODC)

Here we are going to perform an offline attack against a Windows Server 2019 RODC box.

- Attach bootable media with KALI 2019.1
- Boot into OS
- Mount the hard drive
- Navigate to mount point /Windows/System32/config
- Run `samdump2 SYSTEMSAM`



```
root@kali:/media/root/FA0C4FCB0C4F819D/Windows/System32/config# samdump2 SYSTEM SAM
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
*disabled* Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
*disabled* :503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
*disabled* ä:504:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
root@kali:/media/root/FA0C4FCB0C4F819D/Windows/System32/config#
```

What do you notice about these hashes?

RODC Dump the DIT

Key Paths:

/Windows/System32/config

/Windows/NTDS/

Key Files:

/Windows/System32/config/SYSTEM

/Windows/NTDS/ntds.dit

Dumping NTDS.dit from RODC

```
python secretsdump.py -ntds  
/media/windows/Windows/NTDS/ntds.dit -system  
/media/windows/Windows/System32/config/SYSTEM LOCAL
```

```
root@kali: /usr/share/doc/python-impacket/examples# python secretsdump.py -ntds /media/windows/Windows/NTDS/ntds.dit -system /media/windows/Windows/System32/config/SYSTEM LOCAL
Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

[*] Target system bootKey: 0x5fe54ac3c2c994e4f83597c5746e9058
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 899f4a7e2f2a670913daeef49c0752d
[*] Reading and decrypting hashes from /media/windows/Windows/NTDS/ntds.dit
DAL-DC-01$:1107:aad3b435b51404eeaad3b435b51404ee:99103e72fb8c13761c0351c68702ba5d:::
NY-DC-01$:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Mr-R3b00t:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt_2401:1109:aad3b435b51404eeaad3b435b51404ee:083d7891e12d3e03e4c1e257c5bef843:::
evilcorp.local\alices:1104:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
evilcorp.local\janes:1106:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
evilcorp.local\Tyrellw:1105:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Kerberos keys from /media/windows/Windows/NTDS/ntds.dit
DAL-DC-01$:aes256-cts-hmac-sha1-96:b2ca88ee09a146d451ca9793a03eb1ab6270207ed06d16e0b46947da2b199c50
DAL-DC-01$:aes128-cts-hmac-sha1-96:64b0b0c8d5f7ab4f0648f048c904b1fa
DAL-DC-01$:des-cbc-md5:9e266d1a97899231
[*] Cleaning up...
```

Rockyou on KALI

Before using rockyou on KALI you need to extract it:

`gunzip /usr/share/wordlists/rockyou.txt.gz`

```
root@kali:~/Desktop# gunzip /usr/share/wordlists/rockyou.txt.gz
root@kali:~/Desktop# ls /usr/share/wordlists/
dirb  dirbuster  dnsmap.txt  fasttrack.txt  fern-wifi  metasploit  nmap.lst  rockyou.txt  sqlmap.txt  wfuzz
root@kali:~/Desktop#
```


Cracking the hashes

- John
- Hashcat

```
root@kali:~/Desktop# john hashes.txt
Created directory: /root/.john
Warning: detected hash type "LM", but the string is also recognized as "NT"
Use the "--format=NT" option to force loading these as that type instead
Warning: detected hash type "LM", but the string is also recognized as "NT-old"
Use the "--format=NT-old" option to force loading these as that type instead
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 12 password hashes with no different salts (LM [DES 256/256 AVX2-16])
Warning: poor OpenMP scalability for this hash type, consider --fork=2
Will run 2 OpenMP threads
Proceeding with single, rules:Wordlist
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any
Warning: Only 188 candidates buffered for the current salt, minimum 512
needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
    (evilcorp.local\Tyrellw)
    (evilcorp.local\janes)
    (evilcorp.local\alices)
    (krbtgt_2401)
    (Mr-R3b00t)
    (Administrator)
    (krbtgt)
    (Guest)
    (NY-DC-01$)
    (DAL-DC-01$)
Proceeding with incremental:LM_ASCII
```

Extracting Info from NTDS.dit

NTDS uses ESE Database format

<https://github.com/libyal/libesedb/>

<https://github.com/csababarta/ntdsxtract>

```
dsusers.py ntds.dit.export/datatable.3 ntds.dit.export/link_table.5 output --  
syshive SYSTEM --passwordhashes --pwdformat ocl --ntoutfile ntout --  
lmoutfile lmout | tee all_user_info.txt
```

Kudos to : <https://blog.ropnop.com/extracting-hashes-and-domain-info-from-ntds-dit/>

PWNDEFEND

Module 4

Linux Hashes

LINUX Hashes

Linux usernames and passwords are held in the following files:

/etc/passwd

/etc/shadow

```
root@kali:/etc# ls -la /etc/shadow
-rw-r----- 1 root shadow 1503 Mar 17 07:42 /etc/shadow
root@kali:/etc#
```

To read shadow you need appropriate access rights:

Linux Cracking Demo

Because we are learning and we don't want to have to spin up loads of infrastructure let's crack the KALI LIVE CD root password

```
root@kali:/etc# unshadow /etc/passwd /etc/shadow > /root/Desktop/kali_hash.txt
root@kali:/etc# john /root/Desktop/kali_hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (descrypt, traditional crypt(3) [DES 256/256 AVX2-16])
Will run 2 OpenMP threads
Proceeding with single, rules:Wordlist
Press 'q' or Ctrl-C to abort, almost any other key for status
toor (root)
lg 0:00:00:00 DONE 1/3 (2019-03-17 09:19) 100.0g/s 51200p/s 51200c/s 51200C/s root..1root1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:/etc#
```


Unshadow

To extract the hashes from a LINUX system we need to be able to access PASSWD and SHADOW. Shadow requires SU rights to be able to read

```
Usage: unshadow PASSWORD-FILE SHADOW-FILE
root@kali:/etc# unshadow /etc/passwd /etc/shadow
root:X014elvznJq7E:0:0:root:/root:/bin/bash
daemon:*:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:*:2:2:bin:/bin:/usr/sbin/nologin
sys:*:3:3:sys:/dev:/usr/sbin/nologin
sync:*:4:65534:sync:/bin:/bin/sync
games:*:5:60:games:/usr/games:/usr/sbin/nologin
man:*:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:*:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:*:8:8:mail:/var/mail:/usr/sbin/nologin
news:*:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:*:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:*:13:13:proxy:/bin:/usr/sbin/nologin
www-data:*:33:33:www-data:/var/www:/usr/sbin/nologin
backup:*:34:34:backup:/var/backups:/usr/sbin/nologin
list:*:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:*:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:*:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:*:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:*:100:65534:./nonexistent:/usr/sbin/nologin
systemd-timesync:*:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:*:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:*:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
mysql:! :104:109:MySQL Server,,,:/nonexistent:/bin/false
Debian-exim:! :105:110:./var/spool/exim4:/usr/sbin/nologin
uuidd:*:106:112:./run/uuidd:/usr/sbin/nologin
rwhod:*:107:65534:./var/spool/rwho:/usr/sbin/nologin
redsocks:! :108:113:./var/run/redsocks:/usr/sbin/nologin
usbmux:*:109:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
miredo:*:110:65534:./var/run/miredo:/usr/sbin/nologin
ntp:*:111:114:./nonexistent:/usr/sbin/nologin
stunnel4:! :112:116:./var/run/stunnel4:/usr/sbin/nologin
postgres:*:113:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
dnsmasq:*:114:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:*:115:118:./nonexistent:/usr/sbin/nologin
iodine:*:116:65534:./var/run/iodine:/usr/sbin/nologin
arpwatch:! :117:120:ARP Watcher,,,:/var/lib/arpwatch:/bin/sh
Debian-snmpp:! :118:123:./var/lib/snmpp:/bin/false
ssllh:! :119:124:./nonexistent:/usr/sbin/nologin
rtkit:*:120:125:RealtimeKit,,,:/proc:/usr/sbin/nologin
inetsim:*:121:126:./var/lib/inetsim:/usr/sbin/nologin
avahi:*:122:130:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
geoclue:*:123:131:./var/lib/geoclue:/usr/sbin/nologin
sshd:*:124:65534:./run/sshd:/usr/sbin/nologin
colord:*:125:132:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
saned:*:126:134:./var/lib/saned:/usr/sbin/nologin
speech-dispatcher:! :127:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
pulse:*:128:135:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
king-phisher:*:129:137:./var/lib/king-phisher:/usr/sbin/nologin
Debian-gdm:*:130:138:Gnome Display Manager:/var/lib/gdm3:/bin/false
```

PWNDEFEND

Module 5

Hash Identification, Tools and Wordlists

Determine Hash Type

Kali comes with a tool pre-installed called ***hash-identifier***

if we run this tool we can paste in the hash of our target user we collected using unshadow

We can also use hashid

please bear in mind that tools are not 100% correct – check hash examples a well!

https://hashcat.net/wiki/doku.php?id=example_hashes

```
root@kali:/etc# cat /root/Desktop/kali_hash.txt
root:X014elvznJq7E:0:0:root:/root:/bin/bash
daemon*:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin*:2:2:bin:/bin:/usr/sbin/nologin
sys*:3:3:sys:/dev:/usr/sbin/nologin
sync*:4:65534:sync:/bin:/bin/sync
games*:5:60:games:/usr/games:/usr/sbin/nologin
```

```
-----
HASH: X014elvznJq7E
```

```
Possible Hashs:
[+] DES(Unix)
```

```
-----
HASH: █
```

```
root@kali:/etc# hashid X014elvznJq7E
Analyzing 'X014elvznJq7E'
[+] DES(Unix)
[+] Traditional DES
[+] DEScrypt
root@kali:/etc# █
ntdsxtract
```


Hashcat and Linux Hashes

An example of the line required for hashcat is as follows:

***hashcat -m 1500 -a 3
/root/Desktop/kali_hash.txt***

(note if you are running on a VM or certain GPUs you will need to use force)

```
Administrator: Windows PowerShell
PS C:\Users\Mr-R3b0t\Downloads\hashcat-4.2.0\hashcat-4.2.0> .\hashcat64.exe -m 1500 -a 3 .\kali-hashes.txt
hashcat (v4.2.0) starting...

* Device #1: Intel's OpenCL runtime (GPU only) is currently broken.
  We are waiting for updated OpenCL drivers from Intel.
  You can use --force to override, but do not report related errors.
OpenCL Platform #1: Intel(R) Corporation
=====
* Device #1: Intel(R) HD Graphics 630, skipped.
* Device #2: Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, skipped.

OpenCL Platform #2: NVIDIA Corporation
=====
* Device #3: GeForce GTX 1050 Ti, 1024/4096 MB allocatable, 6MCU

Hashfile '.\kali-hashes.txt' on line 2 (daemon...emon:/usr/sbin:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 3 (bin:*:2:2:bin:/bin:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 4 (sys:*:3:3:sys:/dev:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 5 (sync:*:4:65534:sync:/bin:/bin/sync): Token length exception
Hashfile '.\kali-hashes.txt' on line 6 (games:...mes:/usr/games:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 7 (man:*:.../var/cache/man:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 8 (lp:*:7.../var/spool/lpd:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 9 (mail:*...mail:/var/mail:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 10 (news:*...var/spool/news:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 11 (uucp:*...var/spool/uucp:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 12 (proxy:...:13:proxy:/bin:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 13 (www-da...-data:/var/www:/usr/sbin/nologin): Token length exception
Hashfile '.\kali-hashes.txt' on line 14 (backup...p:/var/backups:/usr/sbin/nologin): Token length exception
```

```
Session.....: hashcat
Status.....: Exhausted
Hash.Type.....: decrypt, DES (Unix), Traditional DES
Hash.Target....: X014elvznJq7E
Time.Started...: Sun Mar 17 10:08:00 2019 (0 secs)
Time.Estimated...: Sun Mar 17 10:08:00 2019 (0 secs)
Guess.Mask.....: ?1?2?2 [3]
Guess.Charset...: -1 ?1?d?u, -2 ?1?d, -3 ?1?d*!$@_, -4 Undefined
Guess.Queue.....: 3/8 (37.50%)
Speed.Dev.#3....: 11983.5 kH/s (6.23ms) @ Accel:1 Loops:1024 Thr:256 Vec:1
Recovered.....: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.....: 80352/80352 (100.00%)
Rejected.....: 0/80352 (0.00%)
Restore.Point...: 1296/1296 (100.00%)
Candidates.#3...: sar -> Xqx
HWMon.Dev.#3....: Temp: 44c Fan: 0% Util: 42% Core:1645MHz Mem:3504MHz Bus:16

X014elvznJq7E:toor
```

Hashcat & Wordlists

We can run the previous commands and specify a wordlist as the last parameter:

```
hashcat -m 1500 -a 3 /root/Desktop/kali_hash.txt  
/usr/share/wordlists/rockyou.txt
```

You can also configure masks to aid your cracking efforts!

Tools, tools and more tools!

So far we've dumped hashes, identified their types both using tools and manually and cracked these hashes using the two most popular tools:

- John The Ripper
- Hashcat

There's a whole range of tools including:

L0phtcrack (<http://www.l0phtcrack.com/>)

Cain and Abel – no longer online (<http://www.oxid.it/cain.html>) – use the WayBack machine if you want to see it!

It's not just cracking

To crack hashes we often need to extract them first, a great resource for hash extraction from files are the supplementary files included with John the Ripper.

PWNDEFEND

Hacking 101

Online Tools & Resources

Online Tools

There are a range of online hash cracking tools. Be aware though... you are sending these hashes to a 3rd party so depending upon your scope you may want to validate this is ok first!

<https://crackstation.net/>

<https://hashkiller.co.uk/>

Wordlists

Out of the box Kali includes some great wordlists, however if you are really going to get a good cracking session going on you are going to need bigger wordlists:

The Skull Security Wiki has a range of wordlists

<https://wiki.skullsecurity.org/Passwords>

Seclists is a great resource

<https://github.com/danielmiessler/SecLists/tree/master/Passwords>

Custom Wordlists

To create custom wordlists there's a great tool included in KALI called CEWL!

<https://tools.kali.org/password-attacks/cewl>

PWNDEFEND

Hacking 101

Cracking PDF Passwords

PDF A - Wordlist

- Download the latest version of John The Ripper (Jumbo version)
 - <https://github.com/magnumripper/JohnTheRipper>
- *“git clone https://github.com/magnumripper/JohnTheRipper.git”*
- Run the following command:
 - perl pdf2john.pl [targetfile] > [outputfile]
 - Cleanup the file format (it will have a prefix you need to remove for hashcat)

```
root@dragon007:/media/root/HTB/pentest/pwnDefend# perl /pentest/JohnTheRipper/run/pdf2john.pl Pwndefend\ Hacking\ 101\ Password\ protected\ PDF.pdf
Pwndefend Hacking 101 Password protected PDF.pdf:pdf$4*4*128*-1028*1*16*cd8395167d881d43a00fda45109e76d0*32*f190a2d798fbbceb18d8f9dbd980502d0000000000000000000
0000000000*32*408b37bcf12da873d7f2840f3c1b917a023961ded4c8164d38e46e9655e66775
root@dragon007:/media/root/HTB/pentest/pwnDefend# perl /pentest/JohnTheRipper/run/pdf2john.pl Pwndefend\ Hacking\ 101\ Password\ protected\ PDF.pdf > pdfhash.txt
root@dragon007:/media/root/HTB/pentest/pwnDefend# john pdfhash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
Proceeding with single, rules:Wordlist
Press 'q' or Ctrl-C to abort, almost any other key for status
password (Pwndefend Hacking 101 Password protected PDF.pdf)
1g 0:00:00:00 DONE 1/3 (2019-04-09 15:09) 8.333g/s 866.6p/s 866.6c/s 866.6C/s hackingprotected..passwordhacking
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed
root@dragon007:/media/root/HTB/pentest/pwnDefend#
```

PDF B – Brute Force (pattern)

Example command line:

john --incremental:Alpha -min-len=6 -max-len=6 pdf_bhash.txt

```
root@dragon007:/media/root/HTB/pentest/pwnDefend# john --incremental:Alpha -min-len=6 -max-len=6 pdf_bhash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:04 0.00% (ETA: 2019-04-16 06:58) 0g/s 34346p/s 34346c/s 34346C/s poldup..poppau
█
```

PDF Crack (Hashcat)

There are currently 3 hash modes for PDF hashes:

m 10400

m 10500

m 10600

m 10700

Review the hash output to identify the hash type and compare against the formats:

https://hashcat.net/wiki/doku.php?id=example_hashes

Hashcat Masks

- Crack using a bruteforce mode (-a 3) and a mask of 8 x lowercase words

- ***.\hashcat64.exe -m 10500 -a 3 pdfhash.txt ?1?1?1?1?1?1?1?1***

On a decent rig this won't take long to crack

- Crack 8 characters lower, upper, digits and specials

- ***.\hashcat64.exe -m 10500 -a 3 pdfhash.txt ?1?1?1?1?1?1?1?1 -1 ?l?u?d?s***

- (you can use the “-1 ?a” to specify all characters as well!)

Hashcat Masks Cont.

- If you want to check the range of the mask e.g. from 1 character through to the final mask count then you need to specify the – **increment** parameter
- e.g.

***hashcat64.exe -m 10500 -a 3 pdfhash.txt ?1?1?1?1?1?1?1?1 --
increment -1 ?l?u?d?s***

Cracking a ZIP File

- First step run **john2zip [filename]**

- e.g.

zip2john Zippy.zip > zippy.hash

then run john the ripper

john zippy.hash

(clearly specify parameters such as a wordlist if required e.g. rockyou.txt)

Cracking 7zip

First we need to extract the hash from the 7z file:

(note make sure you have the latest build of John (the jumbo one! git clone <https://github.com/magnumripper/JohnTheRipper.git>)

perl /usr/share/john/7z2john.pl /root/test.7z

You may also want the python version of 7z2john

<https://raw.githubusercontent.com/truongkma/ctf-tools/master/John/run/7z2john.py>

python 7z2john.py test.7z > hash.txt

Next we need to crack the 7z

john hash.txt --wordlist=/usr/share/wordlist/rockyou.txt

```
root@dragon007:~# python 7z2john.py test.7z > hashes.txt
root@dragon007:~# cat hashes.txt
test.7z:$7z$0$19$0$1122$8$b5983be4c00e01990000000000000000$3286569788$112$106$8267781929d5917ce7c5bad9660f7058df4dbedc90107e34fdd1fcb99026e8600672
508e9682a27eca469513b34c9cbd8df8140ab865c7a9fee245c12e249125846ab12f6f52092ef09ccf5f70de220c6ff80c0ef6b778be8208d60943668356c5a45a629979d25de8284c
d93d8e7247
root@dragon007:~# john hashes.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (7z, 7-Zip [SHA256 256/256 AVX2 8x AES])
No password hashes left to crack (see FAQ)
root@dragon007:~# john hashes.txt --show
zip-aes file validation failed [Error loading a zip-aes hash line. The ZIP file 'Access Control.zip' could NOT be found
] Hash is $zip2$*0*3*0*6flcd9ae3480669b2b61dbb4c0fc7ce3*fef9*299a*ZFILE*Access Control.zip*0*4d*9dcc2150285eb46bd46a*$/zip2$
test.7z:password

1 password hash cracked, 0 left
root@dragon007:~#
```


Cracking WordPress/Joomla/PHP

These tools store passwords using MD5 based hashes

Hashcat Command =

.\hashcat64.exe -m 400 .\wordpress.txt .\rockyou.txt stores passwords in MD5

John is slightly simpler with:

john [hashfile]

Cracking Times

Cracking times will vary based on technique, hash type and hardware however here's a rough guide based on brute forcing alpha, numeric and special characters:

5 characters = 7 minutes

6 characters = 10 hours

7 characters = 2 days

8 characters = 5 months

Clearly throwing wordlists at problems and using masks is likely to get you success in a better timeframe.

Also cracking 8 character NTLM hashes is now relatively easy and cheap using cloud hosted GPUs.

PWNDEFEND

Hacking 101

Course 3 Review

Exercises

Exercise A – CYBER CHEF

1. use CYBER CHEF to practise HASHING and ENCODING functions (e.g. MD5 HASH and BASE64 ENCODE/DECODE)

Exercises

Exercise B – Cracking Common Files

1. Use both Hashcat and john the ripper to crack various password protected files
 1. See GITHUB

Review

In this short course we covered:

- Understanding what a HASH is
- Understanding why we use them
- Understanding common hashes for each OS
- Tools and Techniques for cracking hashes
 - Offline cracking
 - Online cracking
- Understand how to use wordlists and make our own custom lists

PWNDEFEND

Hacking 101

Course 3 Complete

Course 3 Complete

We hope this short course content was useful and gave a decent insight into HASHING, the purpose and ways to protect and defend sensitive data such as password protected files and credentials. The subject is huge and this is just a taster. You really need to get your hands on with this and get active with the tools and techniques! Get lab time in, it will pay dividends down the road!

Go and read all the things and practise in safe environments. Remember just because you find it on the internet, even if its exposed to the public doesn't mean you should be poking into it! Remember to operate within your local laws! Use hacking as a force for good! #hack4good

I truly hope this was useful for you, please send comments/feedback/suggestions back to UK_Daniel_Card on twitter!

PWNDEFEND

HACKING 101

<https://twitter.com/pwnDefend>

<https://www.pwndefend.com>

Created by
Xservus
Cyber Security