academy.hackthebox.com

Password Attacks

Themesbrand

10-13 minutes

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

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:	yj9T$3QSBB6CbHEu...SNIP...f8Ms:	18955:	0:	99999:	7
<username>:	<encrypted password>:	<day of last change>:	<min age>:	<max age>:	<

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.

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

Passwd File

```
Exuurxd@htb[/htb]$ cat /etc/passwd
```

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

htb-student:	x:	1000:	1000:	,,,:	/home/htb-student:	/bin/bash
<username>:	<password>:	<uid>:	<gid>:	<comment>:	<home directory>:	<cmd executed after logging 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

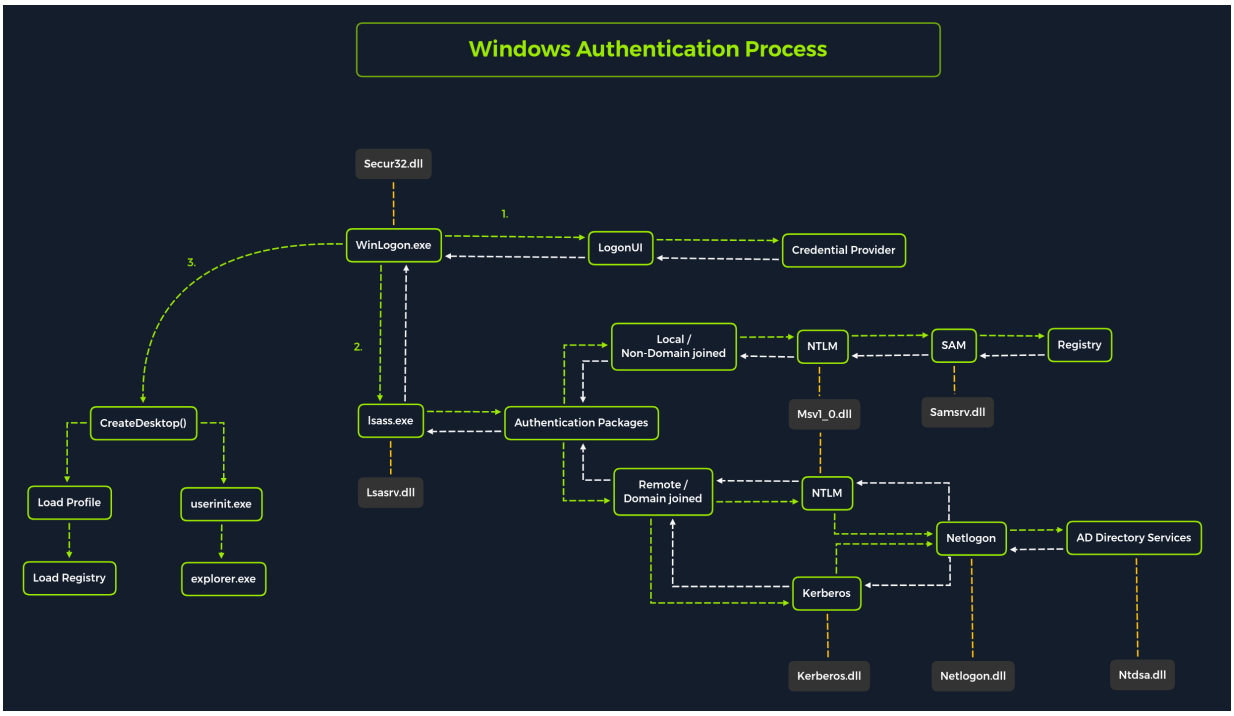
Windows Authentication Process

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.

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.

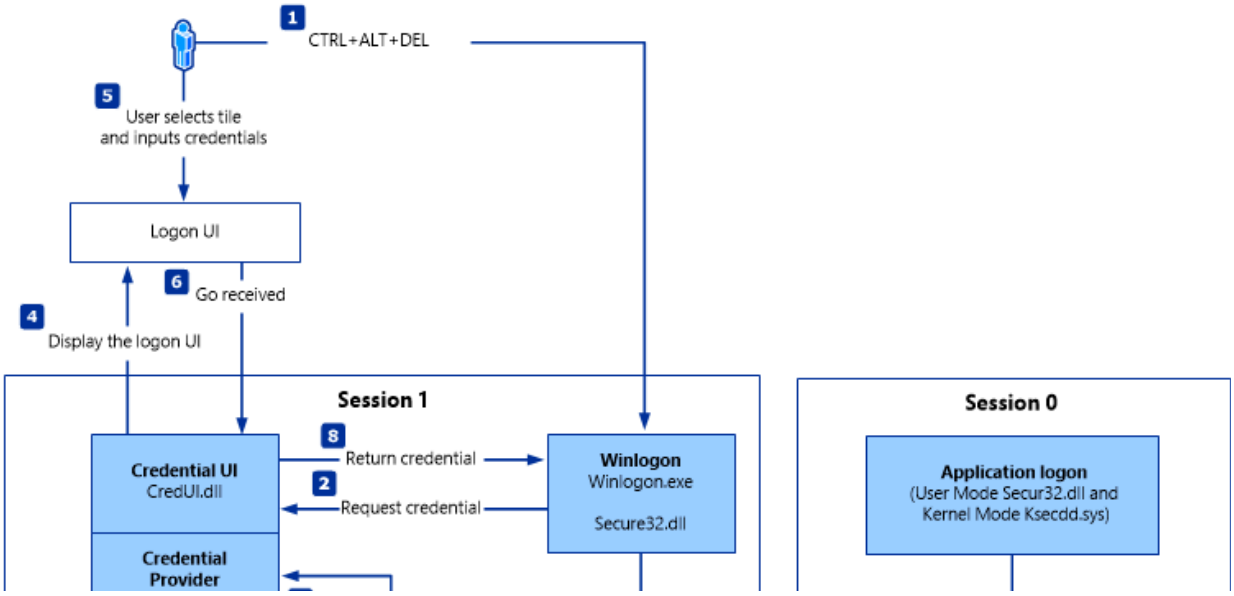
SAM Database

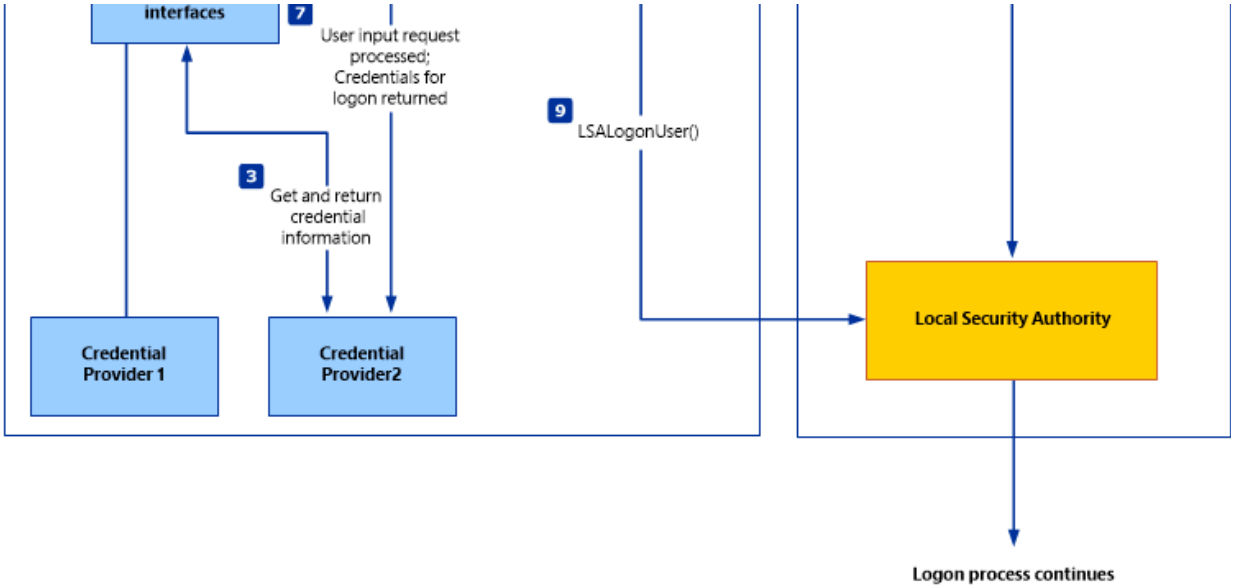
The Security Account Manager (SAM) is a database file in Windows operating systems that stores users' passwords. It can be used to authenticate local and remote users. SAM uses cryptographic measures to prevent unauthenticated users from accessing the system. User passwords are stored in a hash format 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





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:

Credential Manager

```
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.