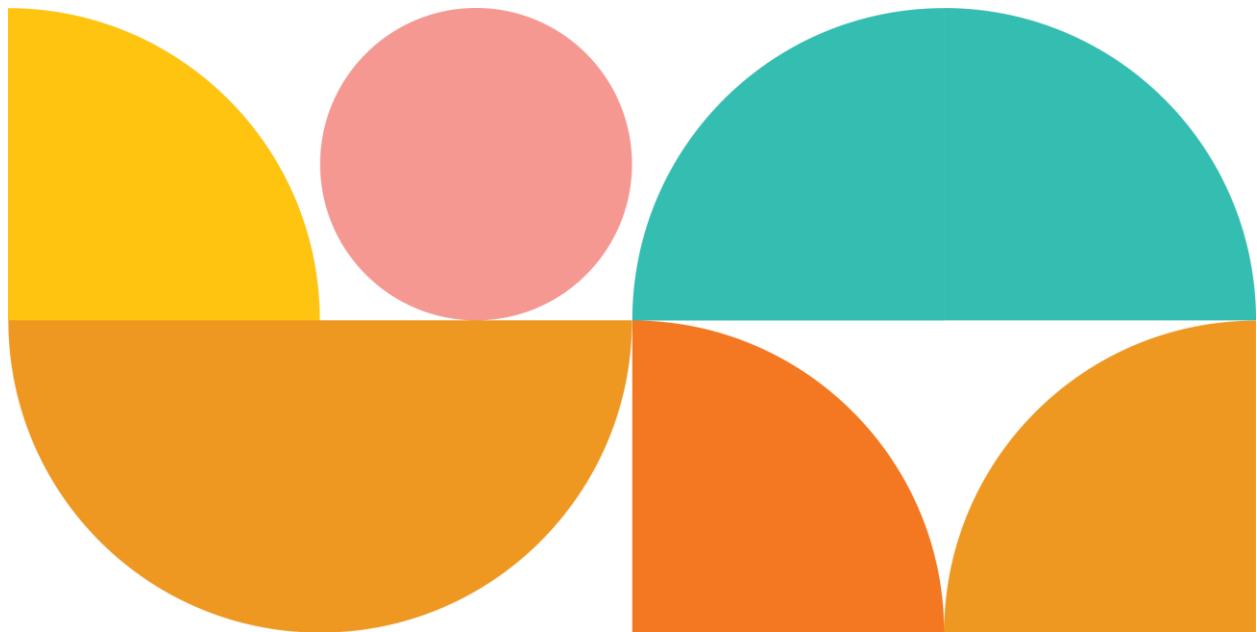


28/10/2024

# Mimikatz



Jocelin Rey

## Qu'est-ce que Mimikatz ?

Mimikatz est un outil open-source puissant utilisé pour extraire les informations d'authentification de la mémoire d'un système Windows. Cet outil est fréquemment employé dans les tests de sécurité pour vérifier si les mots de passe ou jetons d'authentification sont vulnérables aux attaques.

## Installation :

- **Étape 1 : Télécharger Mimikatz**

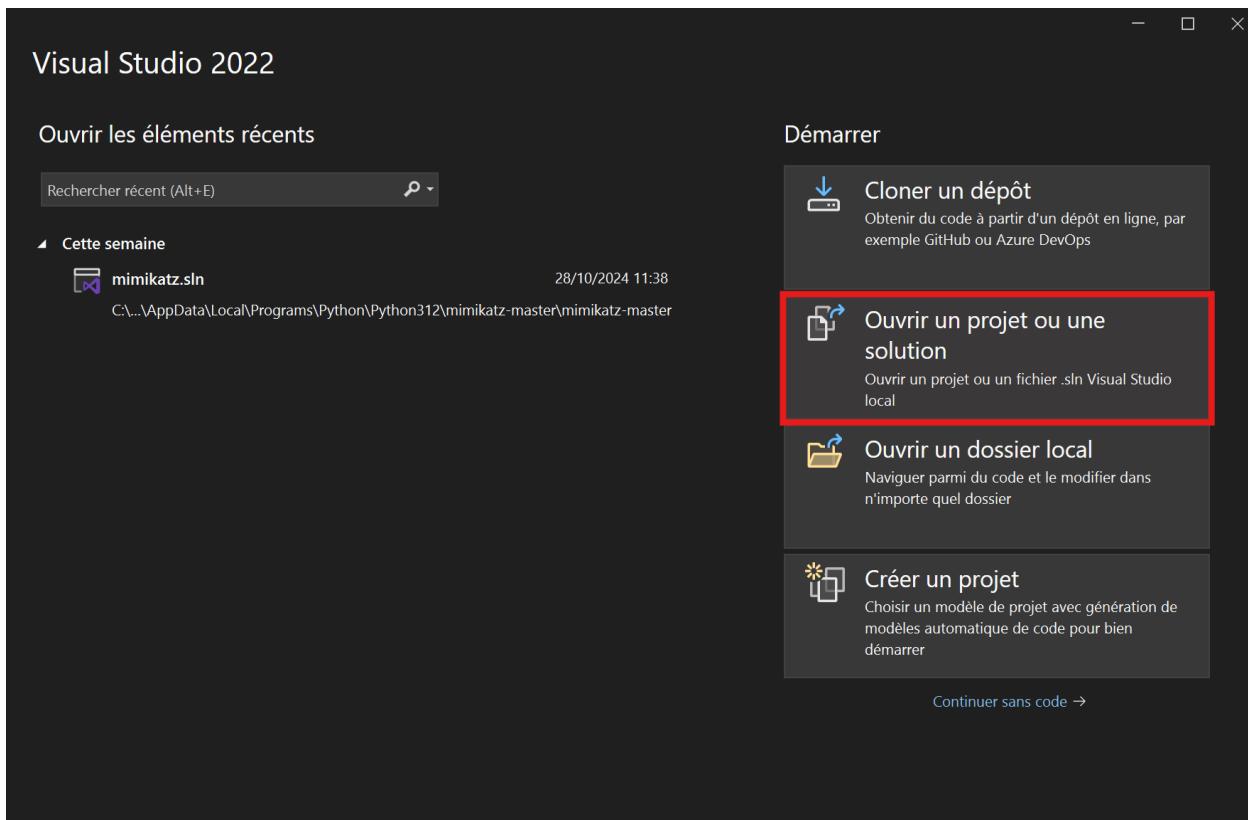
Ouvrez votre navigateur et rendez-vous sur la page officielle de *Mimikatz* sur GitHub : [Mimikatz GitHub](#).

- **Étape 2 : Extraire les fichiers**

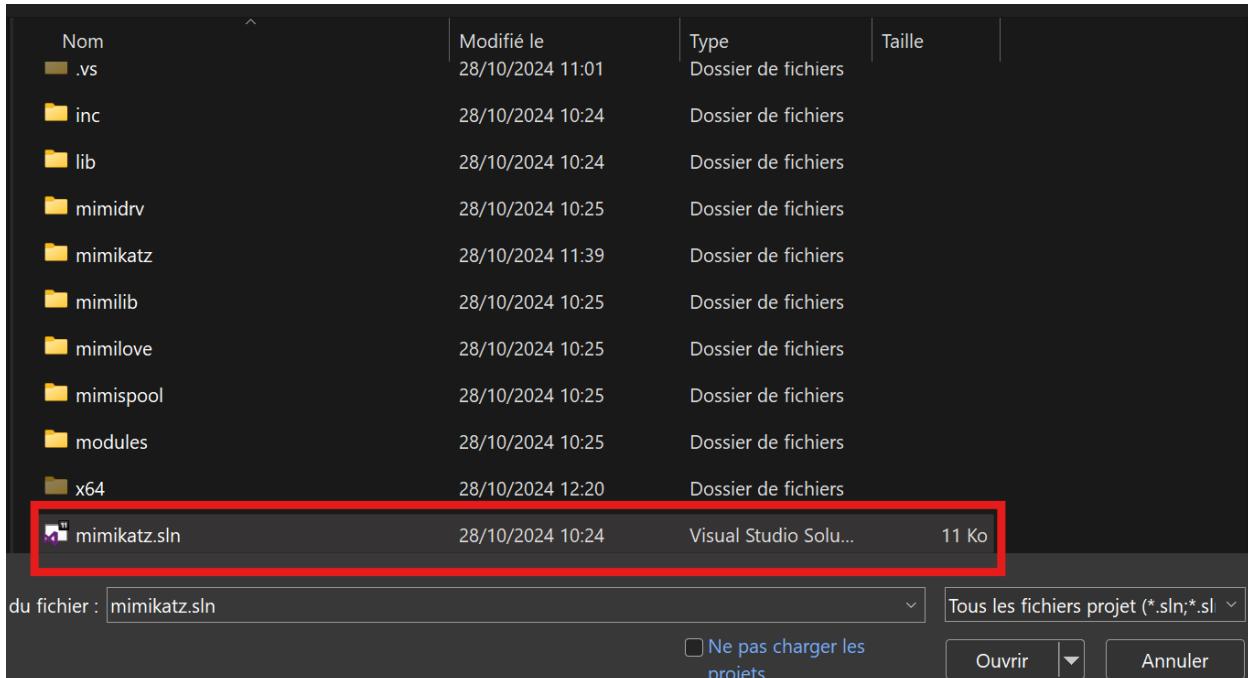
Cliquez sur le bouton "Code" puis sélectionnez "Download ZIP" pour télécharger la dernière version de *Mimikatz*.

Nom	Modifié le	Type	Taille
📁 lib	28/10/2024 10:24	Dossier de fichiers	
📁 mimidrv	28/10/2024 10:25	Dossier de fichiers	
📁 mimikatz	28/10/2024 11:39	Dossier de fichiers	
📁 mimilib	28/10/2024 10:25	Dossier de fichiers	
📁 mimilove	28/10/2024 10:25	Dossier de fichiers	
📁 mimispool	28/10/2024 10:25	Dossier de fichiers	
📁 modules	28/10/2024 10:25	Dossier de fichiers	
📁 x64	28/10/2024 12:20	Dossier de fichiers	
📄 appveyor.yml	28/10/2024 10:24	Fichier source Yaml	2 Ko
📄 kiwi_passwords.yar	28/10/2024 10:24	Fichier YAR	3 Ko
💻 mimikatz.sln	28/10/2024 10:24	Visual Studio Solu...	11 Ko
📄 notrunk.lst	28/10/2024 10:24	MASM Listing	1 Ko
📄 README.md	28/10/2024 10:24	Fichier source Mar...	6 Ko
📄 trunk.lst	28/10/2024 10:24	MASM Listing	1 Ko

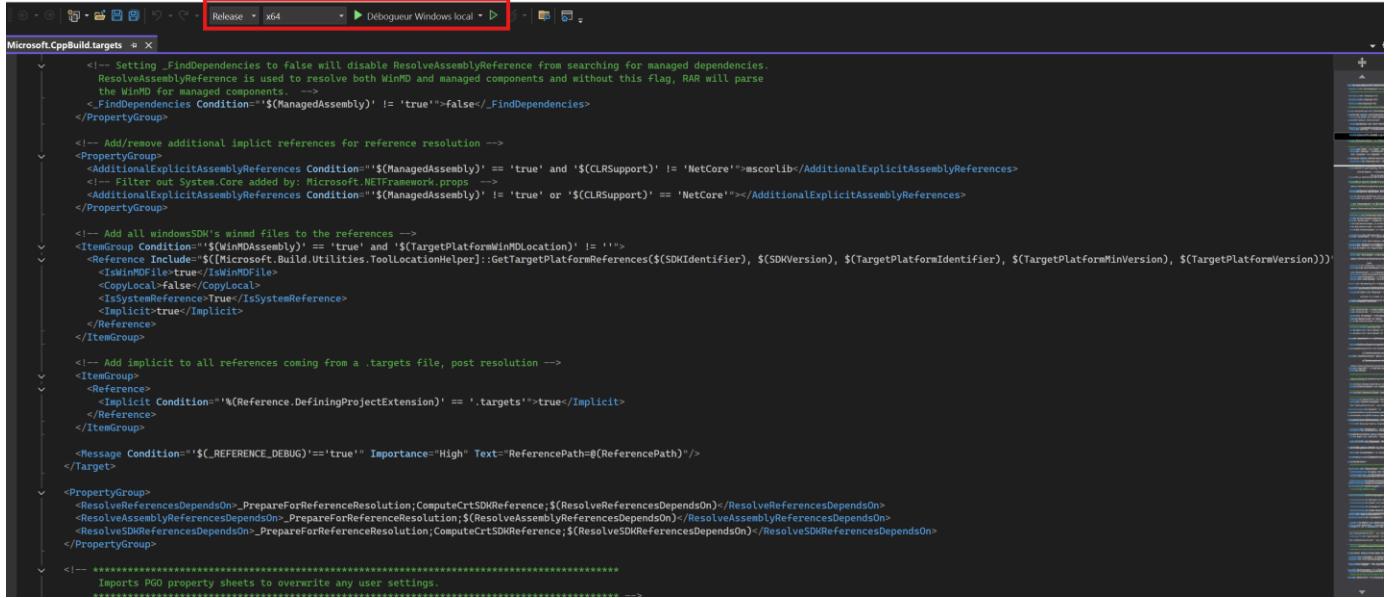
- **Étape 3 : Exécuter Mimikatz dans Visual Studio**



- Ouvrir un projet ou une solution



Selectionné le fichier .sln



```

<!-- Setting _FindDependencies to false will disable ResolveAssemblyReference from searching for managed dependencies.
    ResolveAssemblyReference is used to resolve both WinMD and managed components and without this flag, RAR will parse
    the WinMD for managed components. -->
<_FindDependencies Condition="$(ManagedAssembly) != 'true'>false</_FindDependencies>
</PropertyGroup>

<!-- Add/remove additional implicit references for reference resolution -->
<PropertyGroup>
    <AdditionalExplicitAssemblyReferences Condition="$(ManagedAssembly) == 'true' and '$(CLRSupport)' != 'NetCore'>mscorlib</AdditionalExplicitAssemblyReferences>
    <!-- Filter out System.Core added by: Microsoft.NETFramework.props -->
    <AdditionalExplicitAssemblyReferences Condition="$(ManagedAssembly) == 'true' or '$(CLRSupport)' == 'NetCore'></AdditionalExplicitAssemblyReferences>
</PropertyGroup>

<!-- Add all windowsSDK's winmd files to the references -->
<ItemGroup Condition="$(WinMDAssembly) == 'true' and '$(TargetPlatformWinMDLocation)' != ''>
    <Reference Include="$(Microsoft.Build.Utilities.ToolLocationHelper):GetTargetPlatformReferences($SDKIdentifier), $(SDKVersion), $(TargetPlatformIdentifier), $(TargetPlatformMinVersion), $(TargetPlatformVersion))>
        <IsWinMDFile>true</IsWinMDFile>
        <CopyLocal>false</CopyLocal>
        <IsSystemReference>true</IsSystemReference>
        <Implicit>true</Implicit>
    </Reference>
</ItemGroup>

<!-- Add implicit to all references coming from a .targets file, post resolution -->
<ItemGroup>
    <Reference>
        <Implicit Condition="$(Reference.DefiningProjectExtension) == '.targets'">true</Implicit>
    </Reference>
</ItemGroup>

<Message Condition="$_REFERENCE_DEBUG == 'true'" Importance="High" Text="ReferencePath=@(ReferencePath)"/>
</Target>

<PropertyGroup>
    <ResolveReferencesDependsOn>_PrepareForReferenceResolution;ComputeCrtSDKReference;$(ResolveReferencesDependsOn)</ResolveReferencesDependsOn>
    <ResolveAssemblyReferencesDependsOn>_PrepareForReferenceResolution;$(ResolveAssemblyReferencesDependsOn)</ResolveAssemblyReferencesDependsOn>
    <ResolveSDKReferencesDependsOn>_PrepareForReferenceResolution;ComputeCrtSDKReference;$(ResolveSDKReferencesDependsOn)</ResolveSDKReferencesDependsOn>
</PropertyGroup>

<!-- **** Imports PGO property sheets to overwrite any user settings. -->

```

Puis lancer le Débogeur . Cela va créer un fichier .exe . Lancer le et vous avez fini l'install.

- **Étape 4 : Glossaire des commandes de base Mimikatz**

### 1. privilege::debug

- Description** : Élève les priviléges nécessaires pour interagir avec des processus système, notamment pour accéder à la mémoire des processus protégés comme LSASS.
- Utilisation** : Toujours exécuter cette commande en premier pour permettre l'accès à certaines fonctions de *Mimikatz*.

### 2. sekurlsa::minidump <path\_to\_LSASS.dmp>

- Description** : Permet de charger un fichier *dump* du processus LSASS pour analyse hors ligne. LSASS contient souvent des informations d'authentification importantes.
- Utilisation** : Remplace <path\_to\_LSASS.dmp> par le chemin du fichier LSASS à analyser.

### 3. sekurlsa::process

- Description** : Affiche les informations des sessions utilisateur, avec les identifiants et mots de passe stockés en mémoire active.
- Utilisation** : À lancer après `privilege::debug` pour visualiser les mots de passe en clair lorsque c'est possible.

#### 4. **sekurlsa::logonpasswords**

- a. **Description** : Liste tous les mots de passe des utilisateurs actuellement stockés en mémoire, ainsi que des informations d'authentification.
- b. **Utilisation** : Commande commune pour les audits, affichant les mots de passe stockés de manière non sécurisée.

#### 5. **lsadump::sam**

- a. **Description** : Extrait les *hashes* NTLM et d'autres informations d'authentification du *SAM* (Security Account Manager).
- b. **Utilisation** : Fournit une liste des utilisateurs et de leurs *hashes*, souvent pour vérifier les vulnérabilités de mot de passe.

#### 6. **sekurlsa::pth /user:<username> /domain:<domain> /ntlm:<NTLM\_hash>**

- a. **Description** : Commande pour une attaque *Pass-the-Hash* (PTH), permettant d'utiliser directement un *hash* NTLM pour accéder à des ressources sans le mot de passe en clair.
- b. **Utilisation** : Remplacer <username>, <domain>, et <NTLM\_hash> par les informations spécifiques de l'utilisateur et du domaine cible.

#### 7. **kerberos::ptt <ticket\_file>**

- a. **Description** : Charge un ticket Kerberos pour authentifier une session sur un autre système sans mot de passe.
- b. **Utilisation** : Remplace <ticket\_file> par le chemin vers le fichier du ticket Kerberos.

#### 8. Commandes avancées :

- a. **crypto::certificates /export** : Exporte les certificats numériques des utilisateurs.
- b. **kerberos::golden** : Crée un *Golden Ticket*, une méthode avancée d'exploitation Kerberos.
- c. **kerberos::silver** : Crée un *Silver Ticket* pour obtenir des accès limités mais persistants dans un environnement Active Directory.