

M183 - CheatSheet

Révision Module

sécurité

Par Karel Vilém Svoboda
21 Octobre 2021



Théorie

[Pourquoi Sécuriser ?](#)

[Comment Sécuriser ?](#)

[Les Failles Courantes](#)

[Editeur hexadécimal](#)

[Décompilation](#)

[Les langages de programmation](#)

Exercices de Hacking

[Crackage de la mémoire](#)

Théorie

Pourquoi Sécuriser ?

- Protéger les **données sensibles** des applications
- Les application sont **simples d'accès grâce à internet**
- L'enjeu **économique** car les application sont le **travail des sociétés**
- Garantir une **bonne image** des entreprises et empêcher des **pertes financières** à ces dernières
- Empêcher le **vol de données sensibles**
- Empêcher la **destruction de données** ou de logiciels
- Empêcher l'**interruption des services**

Comment Sécuriser ?

- Sécuriser lors de la **conception et du développement**.
- Identifier les **données critiques** et ajouter des "**mécanismes de sécurité**".
- Ne **jamais** faire confiance à l'utilisateur.
- **Tester la sécurité des produits** pendant le développement et avant la mise en production (**audit de sécurité**)
- Faire attention aux différents **intervenants et partenaires** qui participent à la mise en service du logiciel.
 - Faire attention aux **failles des librairies** ou des langages.
 - Faire attention à son **hébergeur**.
- Analyser les **logs** afin de **repérer les attaques**
- Technologies utilisées :
 - **Anti-Robots** : captcha, login, vérification mail
 - **Anti-DDOS** : taille des requêtes, temps d'exécution, délais de traitement (dans un login attendre 3 min après 5 essais), délais lors d'erreurs
 - **Anti-Piratage de données** : encryption des données sensibles, sécurité physiques (portes scellées, alarmes etc.), réplication (avoir les données sur plusieurs serveurs)
 - **Anti-Hijacking (détournement, vol de données)** : authentification forte avec challenge, géolocalisation, communication cryptée
 - **Anti-Virus et malwares** : analyse continue de la mémoire, des fichiers, hachage etc.

Coût VS Sécurité

Sécuriser coûte cher, c'est pour cette raison que les entreprises cherchent à protéger les données les plus sensibles en priorité. Il cherche à atteindre un équilibre entre le coût et la sécurité.

Les Failles Courantes

- Mauvaise gestion de l'accès aux données (droits, authentification)
- Injections (mémoire, code, sql, fichiers)
- Utilisation d'acteurs externes non sécurisés (applications avec failles etc.)
- Exposition d'informations (message d'erreur, débbug, affichage de l'arborescence etc.)
- Communication non sécurisée (attaque de l'homme du milieu etc.)
- DDOS et brute force (recherche de mot de passe ou de clé en utilisant toutes les combinaisons possibles)

Editeur hexadécimal

- Permet de visualiser et d'éditer le contenu d'un fichier, programme etc.
 - Permet de visualiser les métadonnées (par ex: dimension d'une image) qui sont normalement invisibles sans logiciels spécialisés
- Analyse les octets et les affiche en hexadécimal
 - Hexadécimal = compter en base 16 (0 à F)
- Éditeurs hexadécimaux : Free Hex Editor, HexEdite, HxD, WinHex
- Afin d'éviter de compromettre le fichier, il faut remplacer les nombre exactes d'octets.
 - Ex : 09 A8 07 -> 80 F6 79

Sécurité des applications

- Fonctionnent en local avec la mémoire et des fichiers
- Les élément sont facilement modifiables par l'utilisateur
- Sécuriser une application locale
 - mécanismes pour sécuriser la mémoire ou les fichiers
- Les Failles des application logiciels :
 - édition de la mémoire via logiciels tiers
 - édition des fichiers dans une application
 - décompilation
- Mécanismes de sécurité [...]

Logiciels Malveillants ou Malwares


- Virus
 - Infectent un ordi et se dupliquent
 - pas forcément destructeurs ou nocifs
- Vers
 - Ont comme priorité la réplication et la survie
 - Permettent de rendre inutilisables des milliers d'ordinateurs et de faire tomber un système informatique
 - Propagation
 - Courrier électronique / messagerie
 - Failles logiciels
 - Fichiers contaminés
- Rootkit et backdoor
 - Permet d'avoir accès à un ordi à distance
 - Installé souvent à partir d'un cheval de troie
 - Laisse un backdoor (porte ouverte)

Checksum

- Permet de parer les attaques sur les fichiers binaires (Éditeurs Hexadécimaux)
- Empreinte faite sur la base d'un fichier binaire à l'aide d'un algorithme (généralement du hachage)

Décompilation

- Un décompilateur est un outil servant à reconstituer partiellement ou totalement le code source d'un logiciel à partir d'un exécutable.
 - Ex de décompilateurs : ILSpy ou dotPeek
- C'est un outil dangereux car si on stocke par exemple un mot de passe dans le programme, le hacker n'aura aucune peine à le retrouver.
- Il peut également s'avérer problématique si on développe un logiciel et que nous ne voulons pas qu'il se fasse reproduire.

- 
- Pour prévenir la décompilation il faut utiliser un logiciel externe. Un bon exemple pour les applications sous le Framework .NET est le logiciel ConfuserEX. Ce dernier rendra le code illisible par les décompilateurs.

Les langages de programmation (Lecture 1 - 1)

- Il n'existe pas de langages sans failles.
- Un programme, même parfaitement sécurisé, peut être victime d'une attaque à cause de ces failles.
- Les langages C, C++, C#, Erlang, GO, PHP et Java possèdent plus souvent des failles de sécurité que les langages TypeScript, Clojure ou Rust (réputé le plus sécurisé).
- Les failles :
 - Erreurs de runtime : Javascript
 - Erreurs de types de données : TypeScript
 - Erreurs de mémoire : C++, Objective-C, Java
 - Accès concurrent : CoffeeScript, PHP, Ruby, TypeScript
 - Bugs : PHP
 - Erreurs de conception dues aux données sensibles difficiles à maîtriser : Java
 - Enchaînement de failles : PHP
 - téléverser des données déformée sur un serveur
- Pour corriger PHP, il faudrait revoir complètement le cœur du langage à cause des problèmes de désérialisation.
 - Le processus par lequel un format de niveau inférieur (par exemple, qui a été transféré sur un réseau ou stocké dans un magasin de données) est traduit en un objet lisible ou une autre structure de données.
- Les composants tels que des bibliothèques peuvent ajouter des failles et compromettre la sécurité du programme.
- Les failles les plus fréquentes dans les composants :
 - type cross-scripting (50%)
 - injection de contenu
 - Injection SQL (33%)
- Comment se protéger :
 - Tester pendant le développement et avant la mise en production.
 - Informer et former les développeurs des risques inhérents des langages.
 - Utiliser des composants stables (bon support, peu de failles corrigées en peu de temps)
 - Corriger au plus vite les failles des langages.

Les attaques intérieures (Lecture 1 -2)

- 70% des entreprises ont eu recours à des attaques venant de l'intérieur (60% dans l'année).
- Avec le cloud, les attaques sont moins détectables.
- Les conséquences sont difficilement évaluables.
- 24% des brèches de sécurité ont été produites par des employés négligents
- Les raisons de ces attaques intérieures
 - Non intentionnels
 - tâche effectuée trop rapidement
 - La pression de l'environnement
 - Fatigue
 - la méconnaissance
 - Partage d'information avec la mauvaise personne
 - Click sur un lien de phishing ciblé ou non
 - Absence d'outil de sécurité
 - Intentionnelles
 - Sabotage d'entreprise
 - Vol de données
 - Pour les revendre
 - Pour avoir un meilleur poste dans une autre entreprise
- Comment contrer les attaques internes ?
 - Programme de formation
 - Programme de gouvernance de sécurité des données
 - Supervision des bases de données
 - Double authentification
 - Supervision du comportement
 - Monitoring du comportement
 - Surveillance des employés
 - Suivi des comptes à privilèges
 - Interactions avec les éléments du système d'information comme les bases de données.
 - Détection des comportement anormaux ou abusif qui pourrait indiquer une attaque
 - Un employé qui commence à récupérer des données dans différents systèmes et qui consulte des sites d'emploi prépare sûrement son départ avec les données de l'entreprise.
 - Un employé à qui on vient de refuser une augmentation ou une promotion risque de vouloir se venger en sabotant le système ou en détruisant des données.

Exemple de déchiffrement symétrique

TRANSPOSITION COMPLEXE PAR COLONNES

« LEHEVGE ELA MIC ONTRIYSH »

On remplit les colonnes d'une matrice de largeur 6 et on calcul la hauteur qui est 4 car il y a 24 caractères dans le message ($24 / 6 = 4$)

S E R G I O

6 1 5 2 3 4

Ordre alphabétique



6 1 5 2 3 4

I	L	O	V	E	M
Y	E	N	G	L	I
S	H	T	E	A	C
H	E	R			

Le message est placé dans les colonnes en commençant par la 1 et en terminant par la 6


3 cases vides

Sécuriser des programmes locaux

Le problèmes des programmes c'est que leur données sont stockées en mémoire et donc peuvent être manipulées par des acteurs externes comme des éditeurs hexadécimaux (HxD par exemple).

Réponse aux questions :

1. a sert le logiciel hxd

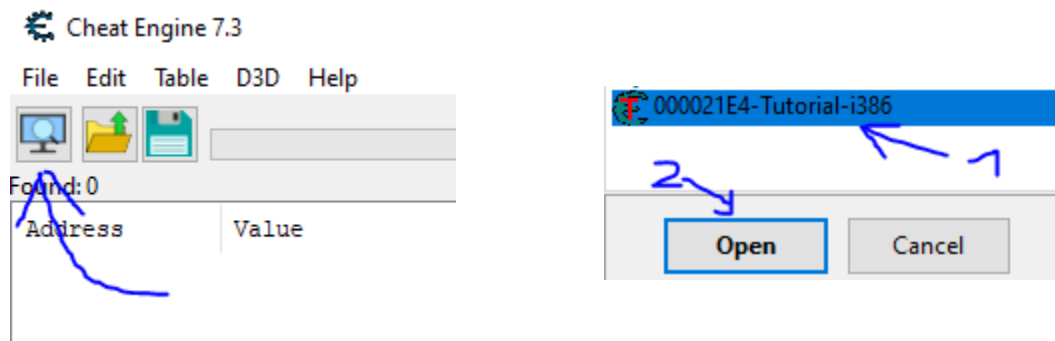
- 
- a. C'est un éditeur hexadécimal qui permet de modifier le contenu des fichiers et programmes.
 2. a sert le logiciel obfuscator
 - a. Un logiciel obfuscator sert à rendre un programme très difficile à lire par un humain. Il est notamment utilisé dans le javascript car le code est facilement accessible par n'importe quel navigateur.
 3. Quel programmes pour regarder le code d'une application
 - a. Un décompilateur, par exemple ILSpy
 4. Comment verifier des mails
 - a. Ne pas ouvrir les pièces jointes et ne pas cliquer sur les liens sans confirmation de la personne
 - b. Vérifier l'adresse emails
 5. 2 type de malware ?
 - a. Spyware, un logiciel espion
 - b. Ransomware, un logiciel qui bloque l'accès aux données contre une rançon.
 6. par quel moyen les malwares s'introduisent
 - a. Souvent par des chevaux de troie.

Exercices de Hacking

Crackage de la mémoire

Logiciel : CheatEngine

1 : appuyer sur l'icône de PC et trouver le programme à hacker

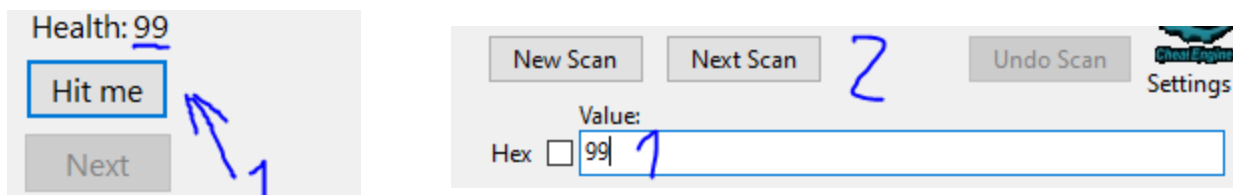


2 : Trouver la première valeur dans le programme et l'entrer dans *Valeur* : et appuyer sur *First Scan*



Health: 100

3 : Changer la valeur dans le programme (ex. de se prendre des dégâts).
Ajouter la nouvelle valeur dans *Valeur* : et appuyer sur *Next Scan* :



4 : Rechercher la nouvelle valeur dans la liste sur la droite

Note : Si il y a plusieurs éléments dans le tableau, répéter l'opération précédente jusqu'à qu'il y'en aie plus qu'une seule.

Address	Value	Previous	First
01958008	99	99	100

5 : Une fois trouvé, double clic sur l'élément et ce dernier va s'afficher dans la fenêtre en bas du programme

Address	Value	Previous	First
01958008	99	99	100

Active	Description	Address	Type	Value
<input type="checkbox"/>	No description	01958008	4 Bytes	99

6 : Double click sur value, ce qui aura comme effet d'ouvrir une fenêtre où nous pouvons changer la valeur

Change Value

what value to change this to?

99

OK

Cancel

Change Value

what value to change this to?

9999

OK

7: Appuyer sur ok et quand vous referez une opération qui descendra votre vie, la valeur changera

Health: 9998

Hit me

BOU DE CODE A LA MISE EN FORME EXPLOSE

Chiffrement MD5 + SALT (chiffrer et déchiffrer des messages) :

Les usings :

```
using System;  
using System.IO;  
using System.Security.Cryptography;  
using System.Text;  
using System.Windows.Forms;
```

Fonction MD5 :

```
// Chiffrer en MD5 une chaîne de caractère  
  
public static string CreateMD5(string input)  
{  
    // Use input string to calculate MD5 hash  
    using (System.Security.Cryptography.MD5 md5 =  
        System.Security.Cryptography.MD5.Create())  
    {  
        byte[] inputBytes = System.Text.Encoding.ASCII.GetBytes(input);  
        byte[] hashBytes = md5.ComputeHash(inputBytes);  
  
        // Convert the byte array to hexadecimal string  
        StringBuilder sb = new StringBuilder();  
        for (int i = 0; i < hashBytes.Length; i++)  
        {  
            sb.Append(hashBytes[i].ToString("X2"));  
        }  
        return sb.ToString();  
    }  
}
```

```
}  
}
```

Créer une clef à l'aide de cette fonction :

```
key = CreateMD5(tbxMessage.Text);
```

Fonction d'encryption :

// Fonction permettant d'encrypter, @value -> chaîne à encrypter @password -> la clef MD5,
@salt -> le salt

```
public static string Encrypt(string value, string password, string salt)  
{  
    DeriveBytes rgb = new Rfc2898DeriveBytes(password, Encoding.Unicode.GetBytes(salt));  
    SymmetricAlgorithm algorithm = new TripleDESCryptoServiceProvider();  
    byte[] rgbKey = rgb.GetBytes(algorithm.KeySize >> 3);  
    byte[] rgbIV = rgb.GetBytes(algorithm.BlockSize >> 3);  
    ICryptoTransform transform = algorithm.CreateEncryptor(rgbKey, rgbIV);  
    using (MemoryStream buffer = new MemoryStream())  
    {  
        using (CryptoStream stream = new CryptoStream(buffer, transform,  
CryptoStreamMode.Write))  
        {  
            using (StreamWriter writer = new StreamWriter(stream, Encoding.Unicode))  
            {  
                writer.Write(value);  
            }  
        }  
    }  
    return Convert.ToBase64String(buffer.ToArray());  
}
```

```
}
```

```
}
```

Fonction de décryptage :

// Fonction permettant de décrypter, @value -> chaine à encrypté @password -> la clef MD5, @salt -> le salt

```
public static string Decrypt(string text, string password, string salt)
{
    DeriveBytes rgb = new Rfc2898DeriveBytes(password, Encoding.Unicode.GetBytes(salt));
    SymmetricAlgorithm algorithm = new TripleDESCryptoServiceProvider();
    byte[] rgbKey = rgb.GetBytes(algorithm.KeySize >> 3);
    byte[] rgbIV = rgb.GetBytes(algorithm.BlockSize >> 3);
    ICryptoTransform transform = algorithm.CreateDecryptor(rgbKey, rgbIV);
    using (MemoryStream buffer = new MemoryStream(Convert.FromBase64String(text)))
    {
        using (CryptoStream stream = new CryptoStream(buffer, transform,
CryptoStreamMode.Read))
        {
            using (StreamReader reader = new StreamReader(stream, Encoding.Unicode))
            {
                return reader.ReadToEnd();
            }
        }
    }
}
```

Encrypter ou Décrypter :

```
// Décrypter un message à l'aide de la clef MD5 et d'un SALT
1 référence
private void BtnDechiffre_Click(object sender, EventArgs e)
{
    tbxContent.Text += "Decrypted : " + Decrypt(crypted, key, "salt") + Environment.NewLine;
}

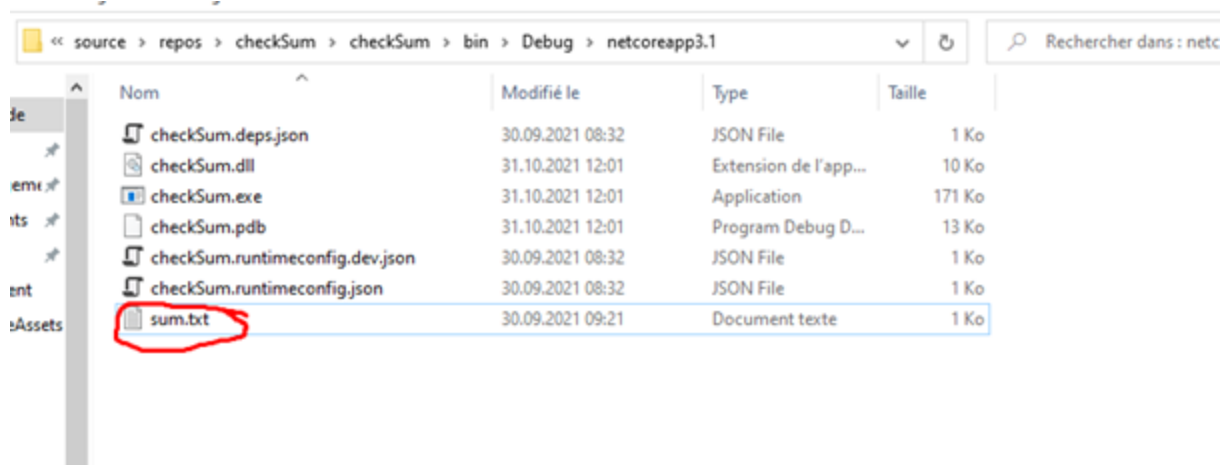
// Encrypter un message à l'aide de la clef MD5 et d'un SALT
1 référence
private void btnChiffrer_Click(object sender, EventArgs e)
{
    crypted = Encrypt(tbxToCrypt.Text, key, "salt");
    tbxContent.Text += "Crypted : " + crypted + Environment.NewLine;
}
```

Vérifier si un fichier a été modifier :

Les usings :

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;
using System.Windows.Forms;
```

Stocker le sum de notre .exe dans un txt :



Récupérer le sum du projet :

```
public string GetSumOfProject(string path)
{
    return File.ReadAllLines(path)[0];
}
```

Fonction pour calculer SUM du .exe lancer :

```
/// <summary>
/// Passer une chaîne de caractère en MD5
/// </summary>
/// <param name="strInput">Chaîne de caractère à passer en MD5</param>
/// <returns>La chaîne en MD5</returns>
3 références
public static string CalculateMD5Hash(Stream content)
{
    MD5 md5 = System.Security.Cryptography.MD5.Create();
    byte[] hash = md5.ComputeHash(content);
    StringBuilder sb = new StringBuilder();
    return BitConverter.ToString(hash);
}
```

Vérifier au chargement de la form si le .exe lancer est pas modifié :

```
private void Form1_Load(object sender, EventArgs e)
{
    if (GetSumOfProject("sum.txt") != CalculateMD5Hash(File.OpenRead("checkSum.exe")))
    {
        Application.Exit();
    }
}
```

Sélectionner le fichier 1 :

```
private void btnSelectFile_Click(object sender, EventArgs e)
{
    string fileName;

    opfDiag = new OpenFileDialog();
    opfDiag.InitialDirectory = @"c:\\";

    if (opfDiag.ShowDialog() == DialogResult.OK)
    {
        fileName = opfDiag.FileName;

        lblFileName.Text = fileName;

        string[] allLines = File.ReadAllLines(fileName);

        tbxCheckSum.Text = CalculateMD5Hash(File.OpenRead(fileName));

        CheckResult();
    }
}
```

Sélectionner le fichier 2 :

```
private void btnSelectFile2_Click(object sender, EventArgs e)
{

```

```

string fileName;

opfDiag = new OpenFileDialog();
opfDiag.InitialDirectory = @"c:\";

if (opfDiag.ShowDialog() == DialogResult.OK)
{
    fileName = opfDiag.FileName;

    lblFileName2.Text = fileName;

    string[] allLines = File.ReadAllLines(fileName);

    tbxCheckSum2.Text = CalculateMD5Hash(File.OpenRead(fileName));

    CheckResult();
}
}

```

Fonction pour comparer 2 fichiers (comparaison de deux clés md5) :

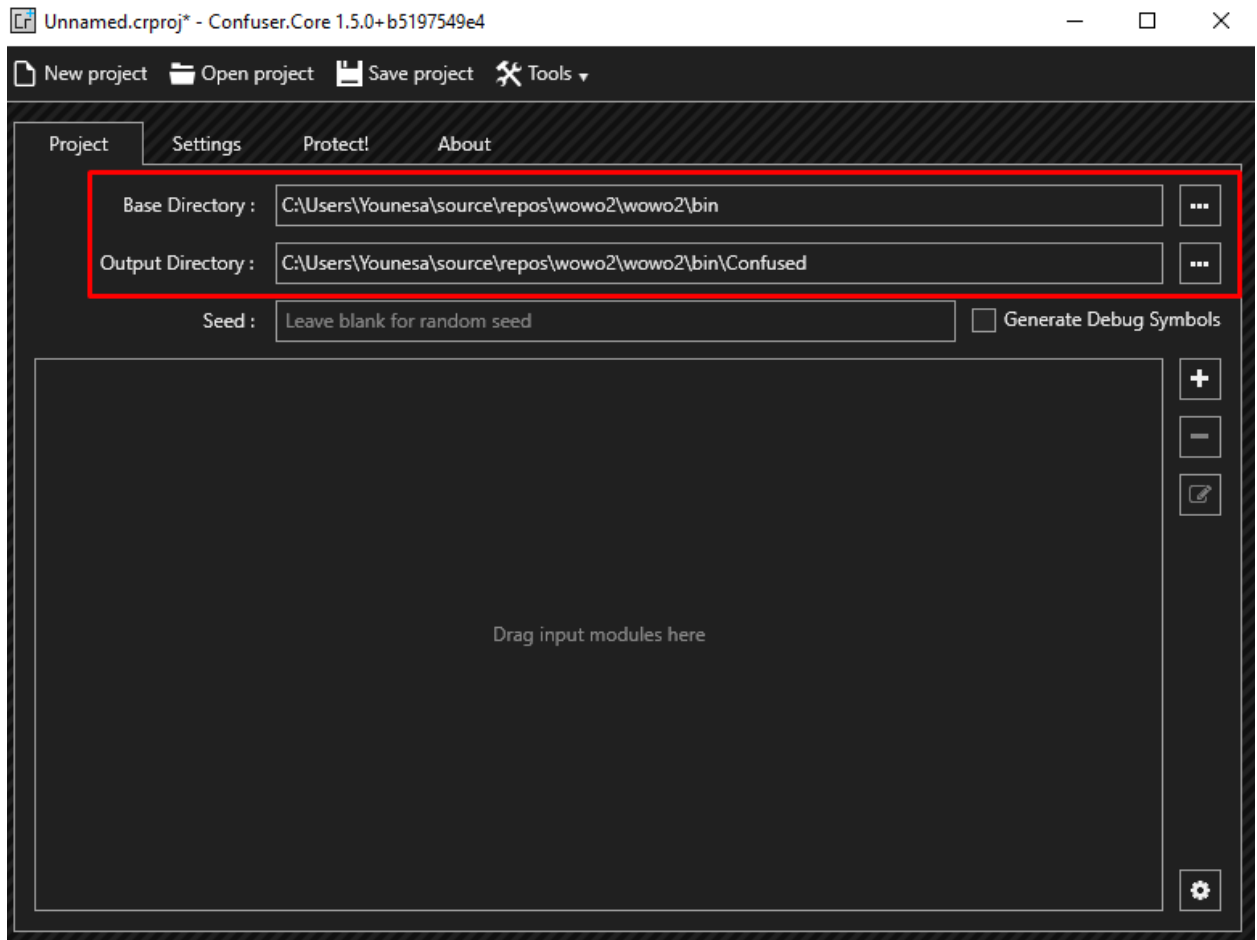
```

2 références
public void CheckResult()
{
    if (tbxCheckSum.Text != tbxCheckSum2.Text)
    {
        lblResult.Text = "Les fichiers sont différents !";
    }
    else
    {
        lblResult.Text = "Le fichier est le bon !";
    }
}

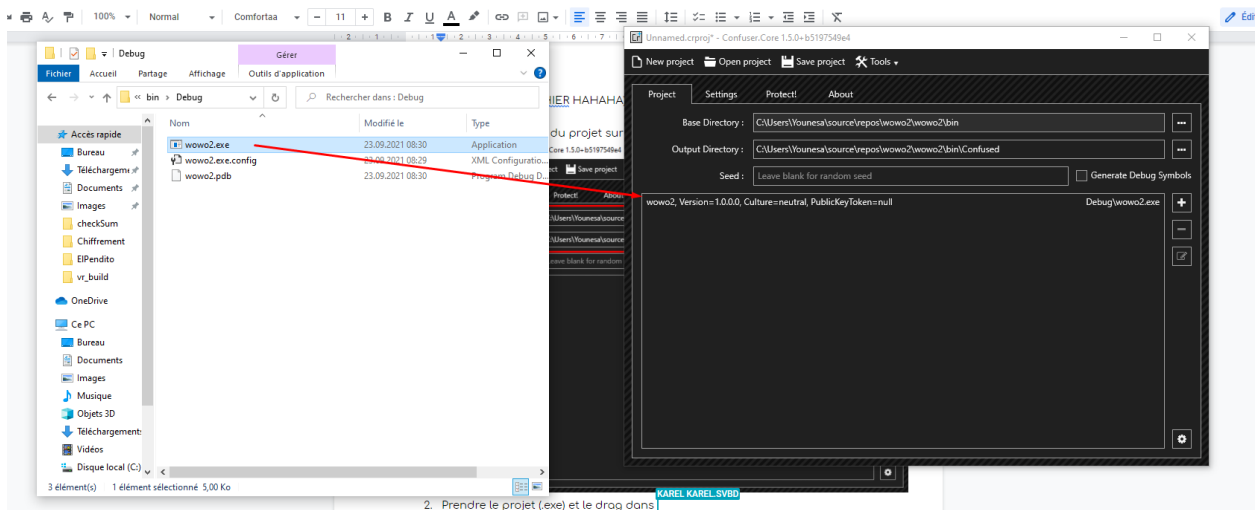
```

OBFUSKÉ 1 FICHIER (FI CHIER HAHHAHA)

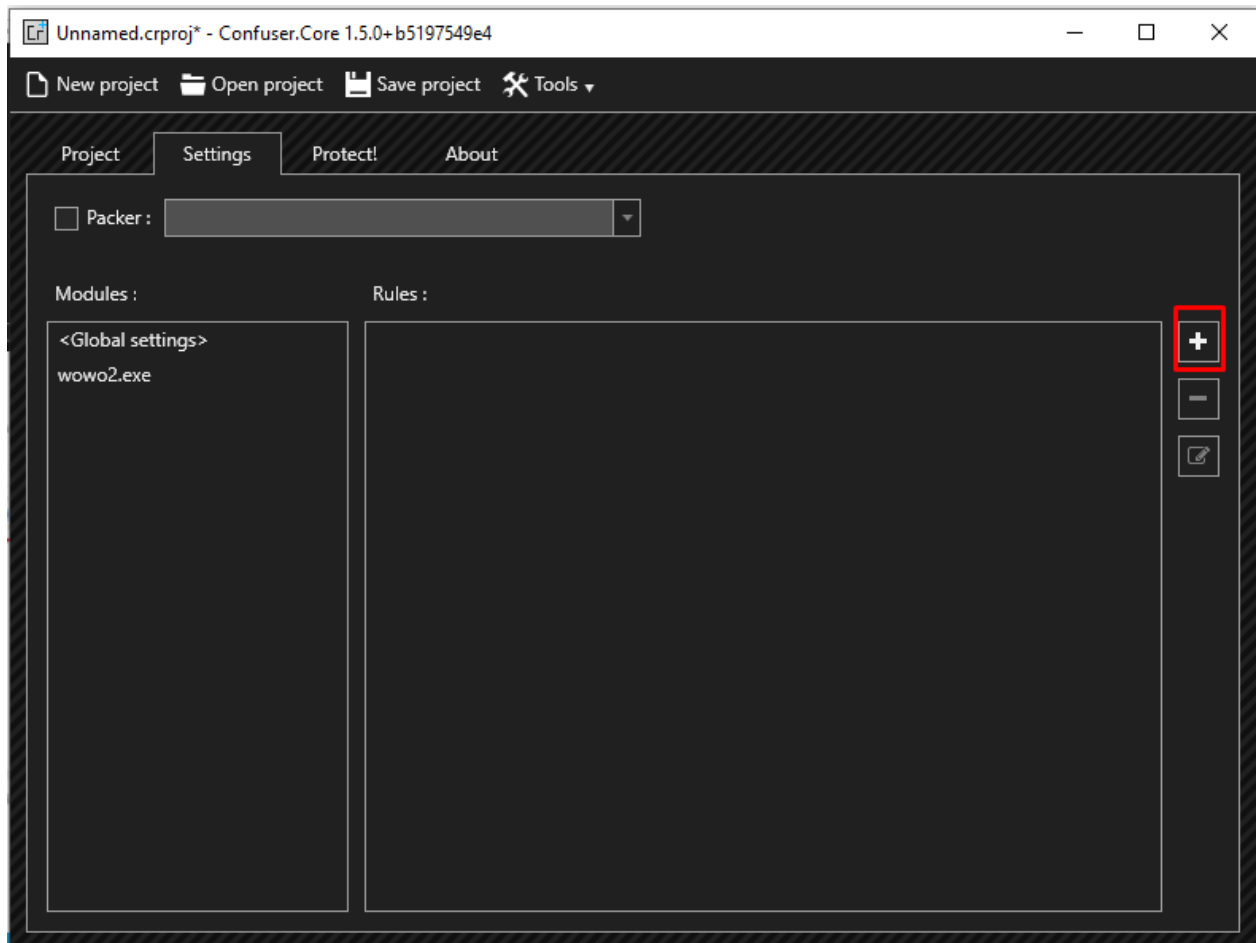
1. Sélectionner le bin du projet sur confuserEx



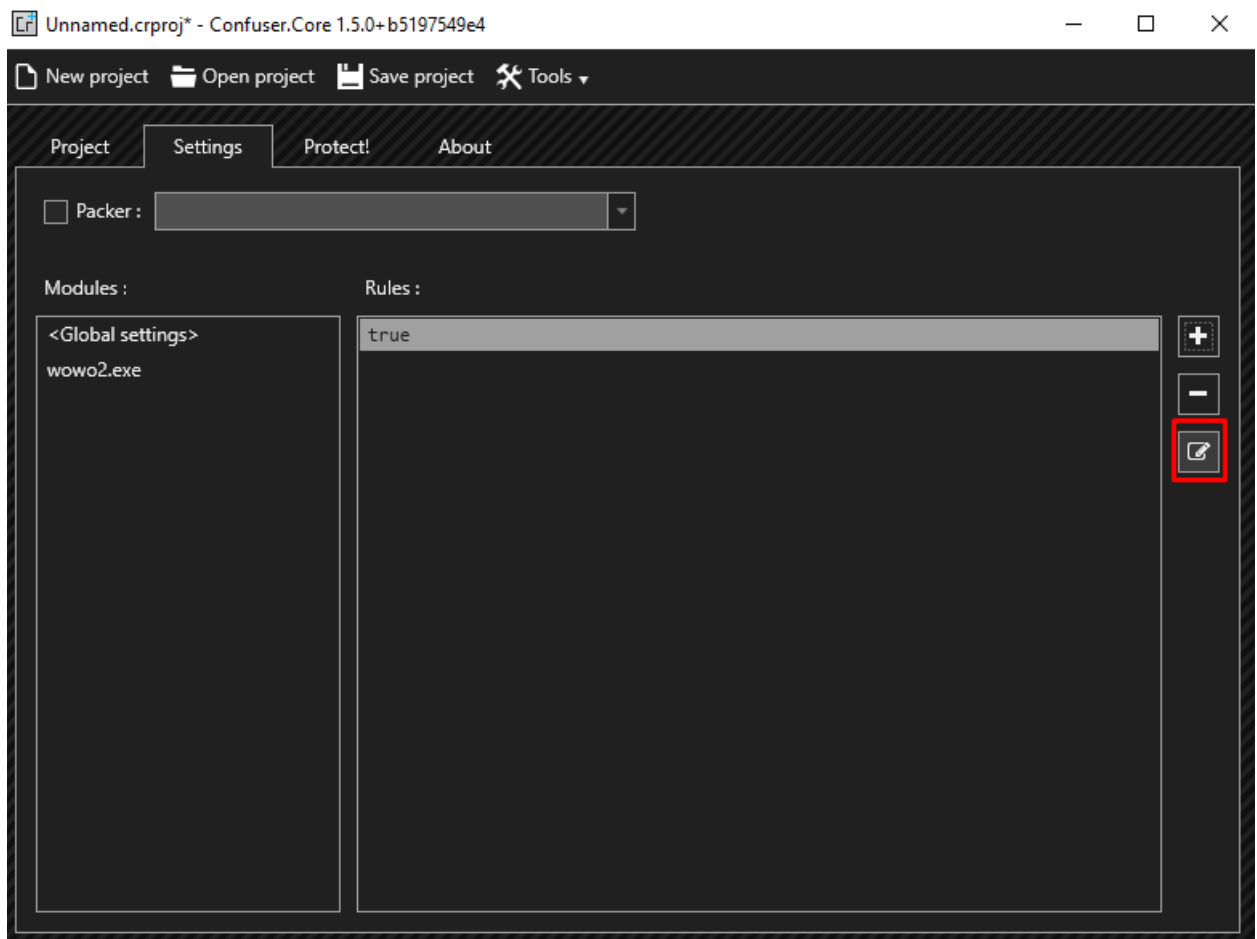
2. Prendre le projet (.exe) et le drag dans le rectangle



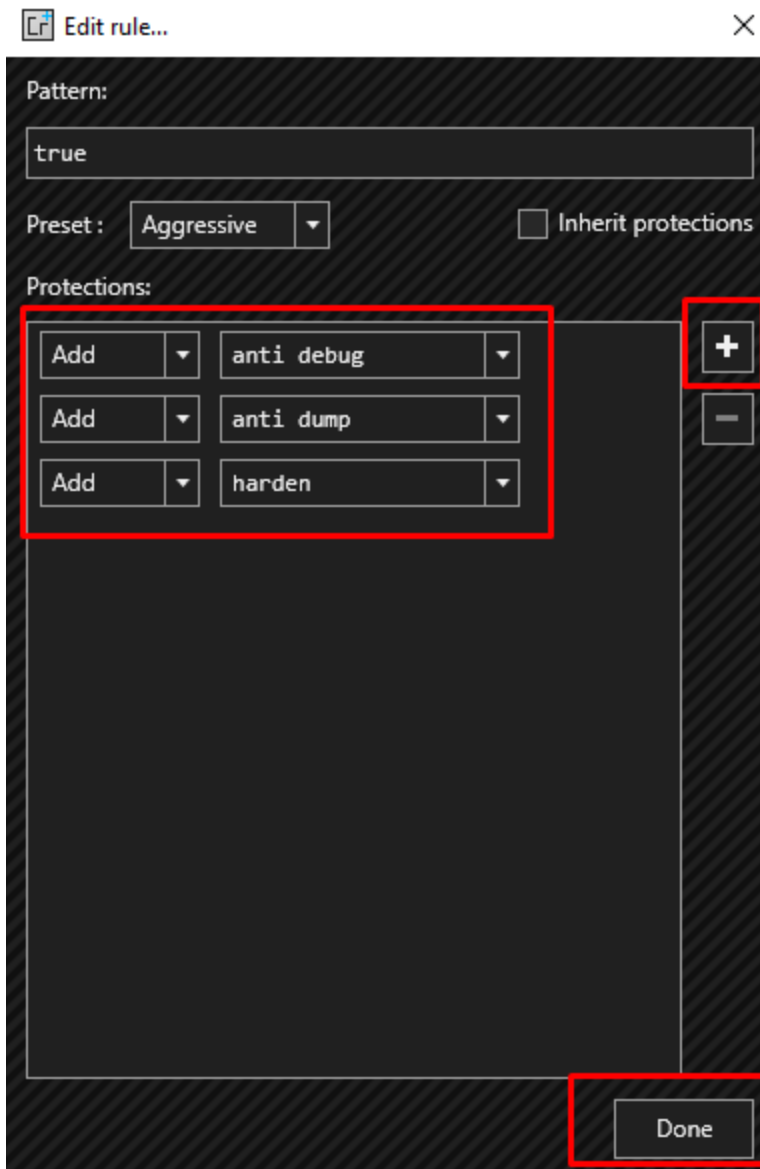
3. Dans setting, appuyer sur plus pour ajouter une sécurité



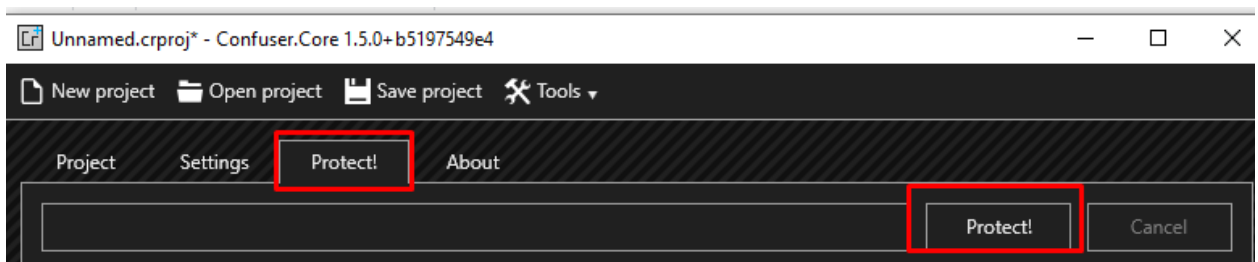
4. Appuyer sur le bouton de modification afin d'ajouter des paramètres de sécurité



5. Mété cé paramétr il von tré b, appuyer sur done
 - a. N'oubliez pas de sélectionner votre .exe



6. Aller sur l'onglet protect et cliquez bande de salopes sur protect



7. ET LA SI C VERT C BON, vous avez votre projet sécurisé dans le fichier "confused"

INFO Done
Finished at 18:50, 0:01 elapsed.

Confused	01.11.2021 18:50	Dossier de fichiers
Debug	23.09.2021 08:30	Dossier de fichiers

POUR DECOMPILER VOILA DES TRUCS :

ILSpy, DotPeek, JustDecompile ou Reflector.