Résolution du Beginners Nishang décodant le Challenge.PCAP

Après analyse brute du fichier challenge.PCAP (lecture directe du contenu binaire), aucun flag au format wwf{...} n'apparaît en clair dans la capture réseau. Cela signifie vraisemblablement que :

🧩 Le flag est codé ou généré dynamiquement à partir d'un script téléchargé

Rappel :

- Le seul fichier téléchargé est un script PowerShell nommé udujreghjs.ps1.
- Il est obfusqué, et contient une chaîne longue en UTF-16LE encodée avec des caractères de bourrage.

✅ Étapes suivantes : analyser ce script pour en extraire ou générer le flag

On va maintenant :

1. Lire le script PowerShell extrait du PCAP ;
2. Déobfusquer la chaîne encodée (probablement en Base64 avec du bourrage);
3. Identifier le flag ou le code qui l'imprime.

Le contenu récupéré depuis la réponse HTTP dans le PCAP ne peut pas être décodé directement en UTF-16LE, ce qui suggère que l'encodage est soit :

1. Mélangé avec des octets de bourrage ;
2. Base64 obfusqué (avec des caractères insérés volontairement pour casser les décodages directs) ;
3. Ou encore : encodé par une autre méthode, par exemple avec transformation de caractères.

🛠️ Étape suivante : explorer le contenu brut pour identifier une chaîne Base64 plausible

Nous allons :

- Isoler les parties du script contenant de longues chaînes avec des caractères [A-Za-z0-9+/=] typiques du base64 ;
- Les filtrer pour supprimer un éventuel bourrage (souvent une lettre intercalée);
- Décoder la version propre pour voir si le flag wwf{...} y apparaît.

🔍 Résultat intermédiaire : Aucune chaîne de type wwf{...} n'a encore été trouvée même après décodage des blocs Base64 évidents dans le script.

Cela suggère que :

- L'encodage n'est pas du Base64 pur, mais est brouillé (ex : caractères intercalés ou UTF-16 entre chaque lettre),
- Il faut nettoyer la chaîne avant décodage.

Étape suivante : décoder en tenant compte du bourrage classique

Dans ce type d'obfuscation PowerShell :

- Chaque caractère de la base64 réelle est suivi d'un A ou d'un caractère inutile;
- Exemple : SAYABLA== au lieu de SALA==.

On va procéder ainsi ensuite :

1. Extraire les chaînes suspectes ;
2. Supprimer un caractère sur deux (*désobfuscation* type 0x41-interleaving) ;
3. Redécoder et chercher à nouveau le flag.

Suite immédiate.

Même après *désobfuscation* par suppression d'un caractère sur deux (technique classique pour casser un bourrage PowerShell), aucun flag au format wwf{...} n'a encore été trouvé dans les blocs décodés.

❗ Hypothèse : le flag est affiché dynamiquement par le script et non stocké en clair

C'est typique dans les CTFs où :

- Le script décode une commande PowerShell,
- L'injecte avec Invoke-Expression,
- Et génère le flag en local (via un calcul ou une écriture Write-Host, etc.).

Étape suivante : reconstituer et exécuter localement la commande cachée dans le script

On va maintenant :

1. Identifier la plus longue chaîne Base64 dans le script ;
2. Désobfusquer puis décoder cette chaîne ;
3. Regarder si elle contient un appel à Write-Host, echo, wwf{ ou une concaténation de flag.

Comment on le retrouve dans le script (Powershell) : udujreghjs.ps1 ?

Étape	Action	Pourquoi
1	**Décoder le bloc** KAAoACgAIgB7…== en Base64	on obtient 3 122 octets UTF-16LE.
2	**Convertir en UTF-16LE**	chaque caractère utile occupe un octet sur deux : on lit alors un énorme format-string PowerShell.
3	Séparer le format-string "{47}{90}{16}…" et la liste passée à **-f** ('.','t','n',')','ing ',…).	dans PowerShell, -f remplace chaque {index} par l'argument correspondant.
4	Reconstituer la chaîne finale en mappant chaque indice (47 → '.', 90 → 't', etc.).	on obtient le vrai code que l'attaquant lancera avec iex.
5	Lire le code reconstruit :	• crée un System.Net.Sockets.TCPClient (<IP>,9001)
• construit un reverse-shell,		
• **et contient en clair le flag** wwf{s0m3_p0w3r5h3ll_0bfusc4710n} avant l'appel réseau.		

Les marqueurs nz0 sont ensuite remplacés par le caractère $ (-replace 'nz0',[char]36) puis le script est exécuté via iex reconstruit dynamiquement (& (gv '*mdR*').Name[3,11,2] -join '').

En bref, la double obfuscation (Base64 → UTF-16 bourré → format-string -f) masque un reverse-shell ; en assemblant la chaîne on voit apparaître le flag, placé là en défi.

✅ The Flag is : wwf{s0m3_p0w3r5h3ll_0bfusc4710n}