Compte-rendu SAÉ304 Découvrir le pentesting

Présenté par : Nabil CHAKER

Avec l'aide de M. **VANSTRACEELE CHRISTOPHE**

Année 2023-2024

IUT Nord Franche-Comté

BUT Réseaux & Télécommunication

INTRODUCTION :

La SAÉ 304, intitulée "Découvrir le Pentesting", avait pour but d'entraîner aux bases de la cybersécurité en utilisant des challenges pratiques. Différents types de défis ont été proposés, comme des exercices sur le web ou encore des tests de programmation. Ce projet, qui s'est déroulé sur six jours, Ma permis de pratiquer et d'explorer des outils et techniques utiles dans le domaine du pentesting.

Command & Control - niveau 5

Énoncé

Berthier, manifestement l'attaquant dispose des mots de passe des systèmes. Le programme malveillant semble être maintenu manuellement sur les machines. Le parc de la société ACME semblant à jour, c'est peut-être les mots de passe qui sont faibles. John, l'administrateur des systèmes ne vous croit pas. Prouvez-lui.

Dans cet exemple, nous allons voir comment extraire un hash utilisateur à partir d'un fichier de dump mémoire. L'opération nécessite de connaître les offsets spécifiques de la ruche **SYSTEM** et **SAM** dans la mémoire. Voici le processus détaillé :

Étape 1 : Identifier les Ruches de Registre

Pour commencer, on va utiliser **Volatility** pour localiser les emplacements des différentes ruches de registre dans le fichier de dump mémoire.

Commande utilisée :

bash

Copier le code

```
$ ./vol.py -f /tmp/e3a902d4d44e0f7bd9cb29865e0a15de.dmp --profile Win7SP1x86 hivelist
```

Sortie de code :

```
Volatile Systems Volatility Framework 2.1
Virtual    Physical   Name
---------- ---------- ----
0x8ee66740 0x141c0740 \SystemRoot\System32\Config\SOFTWARE
0x90cab9d0 0x172ab9d0 \SystemRoot\System32\Config\DEFAULT
0x9670e9d0 0x1ae709d0 \??\C:\Users\John Doe\ntuser.dat
0x9670f9d0 0x04a719d0 \??\C:\Users\John Doe\AppData\Local\Microsoft\Windows\UsrClass.dat
0x9aad6148 0x131af148 \SystemRoot\System32\Config\SAM
0x9ab25008 0x14a61008 \SystemRoot\System32\Config\SECURITY
0x9aba79d0 0x11a259d0 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0x9abb1720 0x0a7d4720 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0x82b6b140 0x02b6b140 [no name]
0x8b20c008 0x039e1008 [no name]
0x8b21c008 0x039ef008 \REGISTRY\MACHINE\SYSTEM
0x8b23c008 0x02ccf008 \REGISTRY\MACHINE\HARDWARE
0x8ee66008 0x141c0008 \Device\HarddiskVolume1\Boot\BCD
```

Dans cet exemple, les offsets pertinents sont les suivants :

- **SYSTEM** : 0x8b21c008
- **SAM** : 0x9aad6148

Une fois les offsets identifiés, on peut utiliser la commande **hashdump** de Volatility pour extraire les hashs NTLM des utilisateurs.

Commande utilisée : $ ./vol.py -f /tmp/e3a902d4d44e0f7bd9cb29865e0a15de.dmp --profile Win7SP1x86 hashdump -y 0x8b21c008 -s 0x9aad6148

```
Volatile Systems Volatility Framework 2.1
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
John Doe:1000:aad3b435b51404eeaad3b435b51404ee:b9f917853e3dbf6e6831ecce60725930:::
```

Pour finire en utilisant john the ripper on peut trouvé les mots de chaque utilisateur mais celui qui nous permet de réussir ce challenge est PASSW0RD

Active Directory - GPO

30 Points Group Policy Preferences

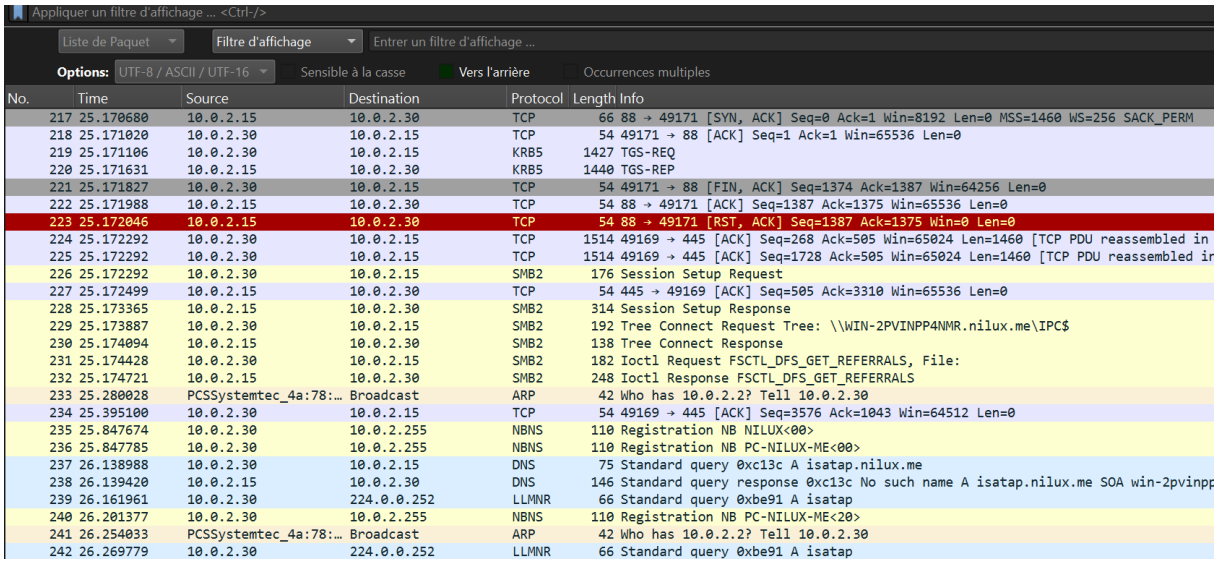
Énoncé

Une capture réseau réalisée au démarrage d'une station de travail, membre d'un domaine Active Directory, a été effectuée lors d'un audit de sécurité. Analysez cette capture et retrouvez le mot de passe de l'administrateur.

Vue d'ensemble des protocoles

Avant de plonger dans les détails des paquets, examinons une vue d'ensemble des différents protocoles utilisés :

En explorant cette liste, le protocole SMB2 est celui qui nous intéresse car c'est un Protocole souvent lié aux échanges sensibles

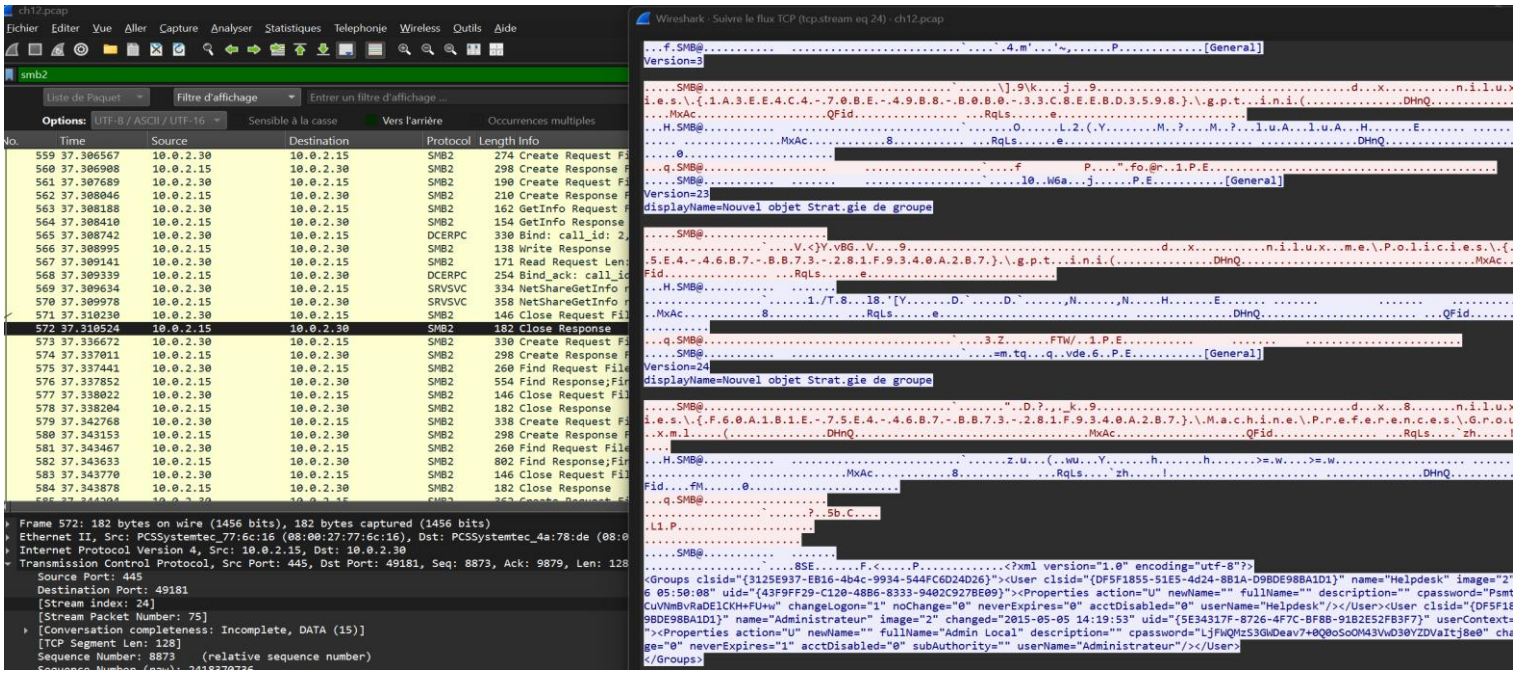


No.	Time	Source	Destination	Protocol	Length	Info
217	25.170680	10.0.2.15	10.0.2.30	TCP	66	88 → 49171 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
218	25.171020	10.0.2.30	10.0.2.15	TCP	54	49171 → 88 [ACK] Seq=1 Ack=1 Win=65536 Len=0
219	25.171106	10.0.2.30	10.0.2.15	KRB5	1427	TGS-REQ
220	25.171631	10.0.2.15	10.0.2.30	KRB5	1440	TGS-REP
221	25.171827	10.0.2.30	10.0.2.15	TCP	54	49171 → 88 [FIN, ACK] Seq=1374 Ack=1387 Win=64256 Len=0
222	25.171988	10.0.2.15	10.0.2.30	TCP	54	88 → 49171 [ACK] Seq=1387 Ack=1375 Win=65536 Len=0
223	25.172046	10.0.2.15	10.0.2.30	TCP	54	88 → 49171 [RST, ACK] Seq=1387 Ack=1375 Win=0 Len=0
224	25.172292	10.0.2.30	10.0.2.15	TCP	1514	49169 → 445 [ACK] Seq=268 Ack=505 Win=65024 Len=1460 [TCP PDU reassembled in
225	25.172292	10.0.2.30	10.0.2.15	TCP	1514	49169 → 445 [ACK] Seq=1728 Ack=505 Win=65024 Len=1460 [TCP PDU reassembled ir
226	25.172292	10.0.2.30	10.0.2.15	SMB2	176	Session Setup Request
227	25.172499	10.0.2.15	10.0.2.30	TCP	54	445 → 49169 [ACK] Seq=505 Ack=3310 Win=65536 Len=0
228	25.173365	10.0.2.15	10.0.2.30	SMB2	314	Session Setup Response
229	25.173887	10.0.2.30	10.0.2.15	SMB2	192	Tree Connect Request Tree: \\WIN-2PVINPP4NMR.nilux.me\IPC$
230	25.174094	10.0.2.15	10.0.2.30	SMB2	138	Tree Connect Response
231	25.174428	10.0.2.30	10.0.2.15	SMB2	182	Ioctl Request FSCTL_DFS_GET_REFERRALS, File:
232	25.174721	10.0.2.15	10.0.2.30	SMB2	248	Ioctl Response FSCTL_DFS_GET_REFERRALS
233	25.280028	PCSSystemtec_4a:78:...	Broadcast	ARP	42	Who has 10.0.2.2? Tell 10.0.2.30
234	25.395100	10.0.2.30	10.0.2.15	TCP	54	49169 → 445 [ACK] Seq=3576 Ack=1043 Win=64512 Len=0
235	25.847674	10.0.2.30	10.0.2.255	NBNS	110	Registration NB NILUX<00>
236	25.847785	10.0.2.30	10.0.2.255	NBNS	110	Registration NB PC-NILUX-ME<00>
237	26.138988	10.0.2.30	10.0.2.15	DNS	75	Standard query 0xc13c A isatap.nilux.me
238	26.139420	10.0.2.15	10.0.2.30	DNS	146	Standard query response 0xc13c No such name A isatap.nilux.me SOA win-2pvinpp
239	26.161961	10.0.2.30	224.0.0.252	LLMNR	66	Standard query 0xbe91 A isatap
240	26.201377	10.0.2.30	10.0.2.255	NBNS	110	Registration NB PC-NILUX-ME<20>
241	26.254033	PCSSystemtec_4a:78:...	Broadcast	ARP	42	Who has 10.0.2.2? Tell 10.0.2.30
242	26.269779	10.0.2.30	224.0.0.252	LLMNR	66	Standard query 0xbe91 A isatap

Filtrer sur SMB2 permet de :

- Réduire le nombre de paquets à examiner manuellement.
- Focaliser l'analyse sur un protocole connu pour transmettre des données exploitables.

Dans ce cas, cela a permis de localiser rapidement un fichier XML contenant un mot de passe chiffré, clé pour résoudre le défi.



name=”Administrateur”

cpassword=”LjFWQMzS3GWDeav7+0Q0oSoOM43VwD30YZDValtj8e0"

ensuite on décript le mdp chiffré avec la commande:

gpp-decrypt LjFWQMzS3GWDeav7+0Q0oSoOM43VwD30YZDValtj8e0

Flag : TuM@sTrouv3

SIP – Authentification

20 Points

Énoncé

Retrouvez le mot de passe utilisé pour s’authentifier sur l’infrastructure SIP.

L'objectif est de récupérer le mot de passe pour l'authentification de l'extension 555. Dans la première ligne de log, nous pouvons voir que l'authentification REGISTER utilise un mot de passe en texte clair avec la méthode PLAIN, et ce mot de passe est 1234.

Ainsi, en se basant sur cette première ligne de log, on peut conclure que le mot de passe pour l'enregistrement de l'extension est 1234, tel qu'indiqué dans la section "PLAIN".

Les autres lignes, qui contiennent des requêtes INVITE et BYE, utilisent une méthode d'authentification plus sécurisée, MD5, et présentent des hashs du mot de passe. Cependant, l'essentiel est que le mot de passe clair utilisé pour l'authentification REGISTER est 1234.

```
172.25.105.3"172.25.105.40"555"asterisk"REGISTER"sip:172.25.105.40"4787f7ce""""PLAIN"1234
172.25.105.3"172.25.105.40"555"asterisk"INVITE"sip:1000@
172.25.105.40"70fbfdae""""MD5"aa533f6efa2b2abac675c1ee6cbde327
172.25.105.3"172.25.105.40"555"asterisk"BYE"sip:1000@
172.25.105.40"70fbfdae""""MD5"0b306e9db1f819dd824acf3227b60e07
```

POP – APOP 15 Points

Authentification sécurisée

Énoncé :

Retrouver le mot de passe de l'utilisateur dans la trame réseau.

Pour cela rien de plus simple vue qu'une partie de la réponses se trouve dans le titre du challenge. Ce qui nous indique que deux protocoles serons principalement utilisé pour se défis POP et APOP

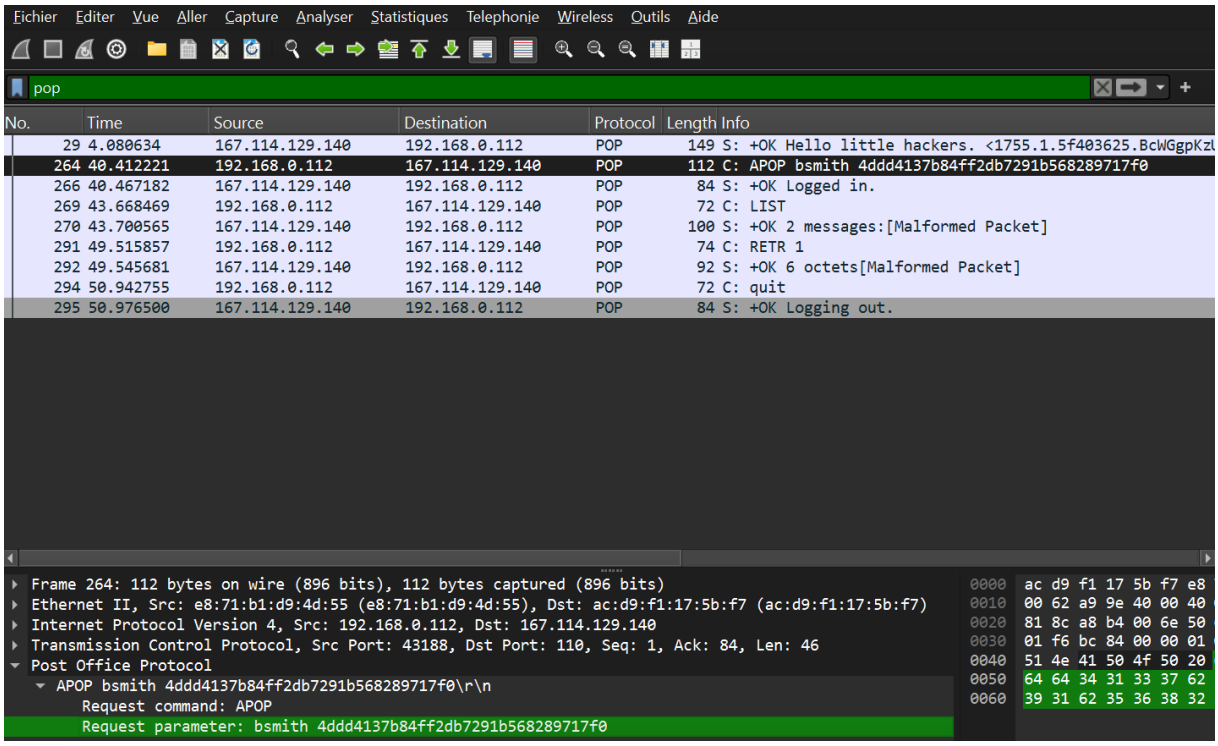
On peut voir que la trame réseau wireshark utilise APOP pour sécuriser l'authentification en envoyant un challenge unique à chaque session.

Identifier les données pertinentes

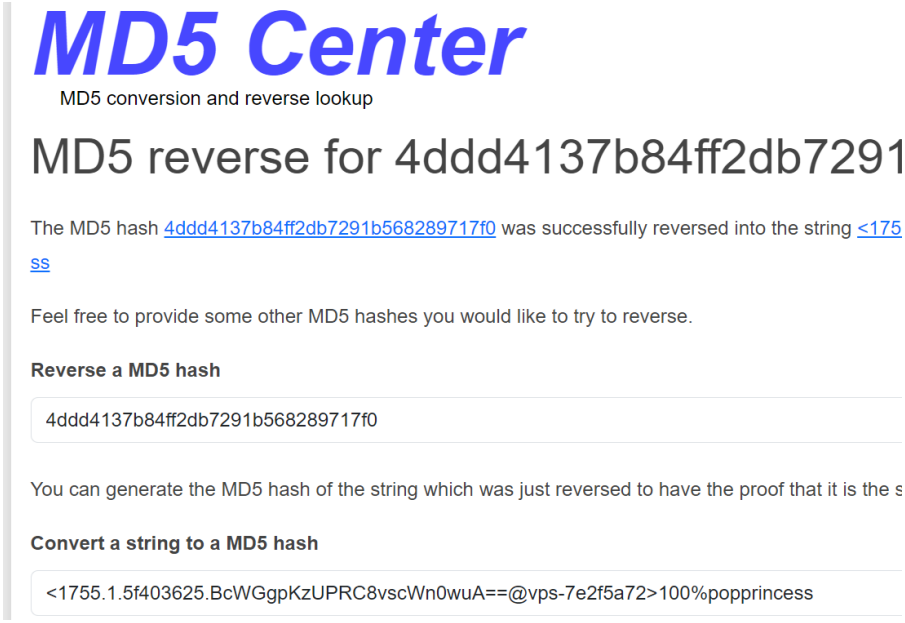
Dans le flux TCP ci-dessous, une chaîne spécifique attire l'attention :

4ddd4137b84ff2db7291b568289717f0 → mdp passe cryptée en md5

Cette chaîne est un hash MD5. Le protocole APOP génère cet hash en combinant deux éléments



Cependant, cette méthode repose sur MD5, un algorithme de hachage qui n'est plus considéré comme sécurisé face aux attaques modernes. Les bases de données de hash permettent de casser rapidement ce type de sécurité :



En combinant notre analyse réseau et des outils de décryptage de hash, nous avons pu retrouver le mot de passe utilisateur.
Le mot de passe final est : **100%popprincess**

TCP - La roue romaine

10 Points

Programmation réseau

Énoncé

Pour commencer cette épreuve utilisant le protocole TCP, vous devez vous connecter à un programme sur une socket réseau.

Vous devez décoder la chaîne de caractères encodée en ROT13 envoyée par le programme.
Vous avez 2 secondes pour envoyer la bonne réponse à partir du moment où le programme vous envoie la chaîne.
La réponse doit être envoyée sous la forme de string.

Le script commence par importer la bibliothèque **Pwntools** (via import pwn), un outil fréquemment utilisé dans le domaine de la sécurité informatique pour interagir avec des services distants. La fonction principale du script, main, établit une connexion avec un serveur situé à l'adresse **challenge01.root-me.org** sur le port **50201**


```
┌──(myenv)─(root㉿kali)-[/home/kali/script]
└─# python3 ok.py
[+] Opening connection to challenge01.root-me.org on port 52021: Done
/home/kali/script/ok.py:22: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  conn.recvuntil("string is '")
/home/kali/script/ok.py:23: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  encoded_string = conn.recvuntil("'")[:-1].decode()
/home/kali/script/ok.py:29: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  conn.sendline(decoded_string)
[+] Receiving all data: Done (80B)
[*] Closed connection to challenge01.root-me.org port 52021
. What is your answer ? [+] Good job ! Here is your flag: RM{TCP_R0man_Wh33l!!}


┌──(myenv)─(root㉿kali)-[/home/kali/script]
└─# nano ok.py
```

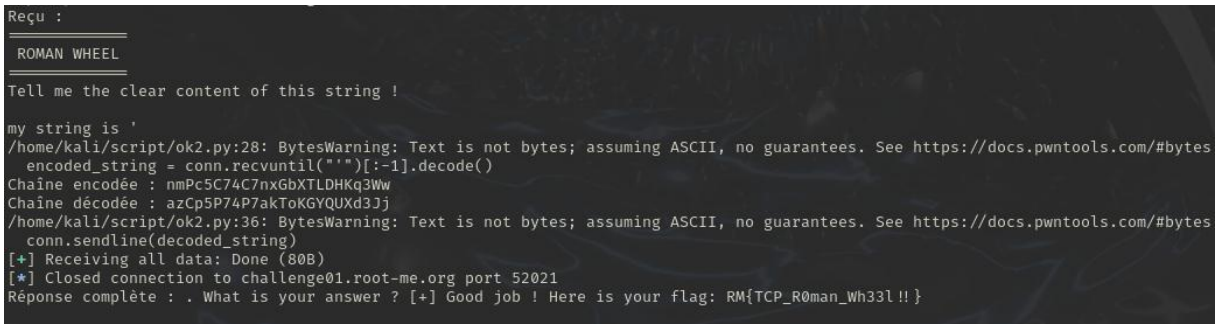
FLAG : RM{TCP_R0man_Wh33l!!}

TCP - Uncompress Me 15 Points

Programmation réseau

```
File  Actions  Edit  View  Help
  GNU nano 8.2                                                    ok2.py
import pwn

def decode_rot13(string):
    """
    Decode une chaîne encodée en ROT13.
    """
    return ''.join(
        chr((ord(char) - ord('a') + 13) % 26 + ord('a')) if 'a' ≤ char ≤ 'z' else
        chr((ord(char) - ord('A') + 13) % 26 + ord('A')) if 'A' ≤ char ≤ 'Z' else
        char
        for char in string
    )

def main():
    # Paramètres de connexion
    host = 'challenge01.root-me.org'
    port = 52021

    try:
        # Se connecter au challenge
        conn = pwn.remote(host, port)

        # Lire les données jusqu'à l'indicateur
        prompt = conn.recvuntil("string is '").decode()
        print("Reçu :", prompt)  # Affiche la partie initiale pour débogage

        # Récupérer la chaîne encodée
        encoded_string = conn.recvuntil("'")[:-1].decode()
        print("Chaîne encodée :", encoded_string)  # Affiche la chaîne encodée

        # Décoder la chaîne de caractères en ROT13
        decoded_string = decode_rot13(encoded_string)
        print("Chaîne décodée :", decoded_string)  # Affiche la chaîne décodée

        # Envoyer la réponse
        conn.sendline(decoded_string)

        # Récupérer et afficher la réponse du challenge
        response = conn.recvall().decode()
        print("Réponse complète :", response)

    except Exception as e:
        print(f"Erreur rencontrée : {e}")

    finally:
        # Fermer la connexion si elle est toujours active
        try:
            conn.close()
        except:
            pass

if __name__ == "__main__":
    main()
```

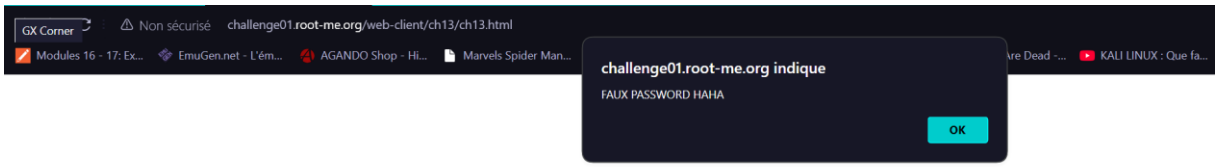
Copie d'ecran du resultat après lancement du code :



```
Reçu :
 ═══════════
  ROMAN WHEEL
 ═══════════
Tell me the clear content of this string !

my string is '
/home/kali/script/ok2.py:28: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  encoded_string = conn.recvuntil("'")[:-1].decode()
Chaîne encodée : nmPc5C74C7nxGbXTLDHKq3Ww
Chaîne décodée : azCp5P74P7akToKGYQUXd3Jj
/home/kali/script/ok2.py:36: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  conn.sendline(decoded_string)
[+] Receiving all data: Done (80B)
[*] Closed connection to challenge01.root-me.org port 52021
Réponse complète : . What is your answer ? [+] Good job ! Here is your flag: RM{TCP_R0man_Wh33l!!}
```

Javascript - Obfuscation 3 30 Points

Dans un premier temps on voit qu'on a accès à une page web avec lequel on ne peut absolument rien faire.



Tout d'abord, il est essentiel d'analyser le code source pour identifier les fonctions JavaScript disponibles.

```
<html>
<head>
    <title>Obfuscation JS</title>
    <script type="text/javascript">
    function dechiffre(pass_enc){
        var pass = "70,65,85,88,32,80,65,83,83,87,79,82,68,32,72,65,72,65";
        var tab  = pass_enc.split(',');
                var tab2 = pass.split(',');var i,j,k,l=0,m,n,o,p = "";i = 0;j = tab.length;
                        k = j + (l) + (n=0);
                        n = tab2.length;
                        for(i = (o=0); i < (k = j = n); i++ ){o = tab[i-l];p += String.fromCharCode((o = tab2[i]));
                                if(i == 5)break;}
                        for(i = (o=0); i < (k = j = n); i++ ){
                        o = tab[i-l];
                                if(i > 5 && i < k-1)
                                        p += String.fromCharCode((o = tab2[i]));
                        }
        p += String.fromCharCode(tab2[17]);
        pass = p;return pass;
    }
    String["fromCharCode"](dechiffre("\x35\x35\x2c\x35\x36\x2c\x35\x34\x2c\x37\x39\x2c\x31\x31\x35\x2c\x36\x39\x2c\x31\x31\x34\x2c\x31\x31\x36\x2c\x31\x30\x37\x2c\x34\x39\x2c\x35\x30"));

    h = window.prompt('Entrez le mot de passe / Enter password');
    alert( dechiffre(h) );

</script>
</head>

</html>
```

On remarque un appel à la fonction dechiffre() avec, en paramètre, une chaîne codée en ASCII. Cette chaîne est ensuite passée comme argument à la fonction fromCharCode() :

Décodage pas à pas de la chaîne ASCII :

1. **Analyse de \x35** :
    - En décimal, \x35 correspond à **5**.
    - Ainsi, \x35\x35\x2c donne "55,".
2. **Utilisation de String.fromCharCode(55)** :
    - Cela retourne le caractère ASCII correspondant : **7**.

\x35\x35\x2c\x35\x36\x2c\x35\x34\x2c\x37\x39\x2c\x31\x31\x35\x2c\x36\x39\x2c\x31\x31\x34\x2c\x31\x31\x36\x2c\x31\x30\x37\x2c\x34\x39\x2c\x35\x30

➔ ASCII 55,56,54,79,115,69,114,116,107,49,50
➔ Décimale : 786OsErtk12

La chaîne **"786OsErtk12"** est le mot de passe du challenge.

FLAG : 786OsErtk12

XSS - Stockée 1 30 Points

Du gateau !

Énoncé

Volez le cookie de session de l'administrateur et utilisez le pour valider l'épreuve.

Défi XSS - Stockée 1 sur Root-Me

Objectif du défi :

Exploiter une vulnérabilité XSS pour voler les cookies de l'administrateur via un script malveillant et utiliser ces cookies pour valider le défi.

Tester une injection XSS

- Injectez un script simple dans le champ "Message" pour tester la vulnérabilité, comme :

html

Copier le code

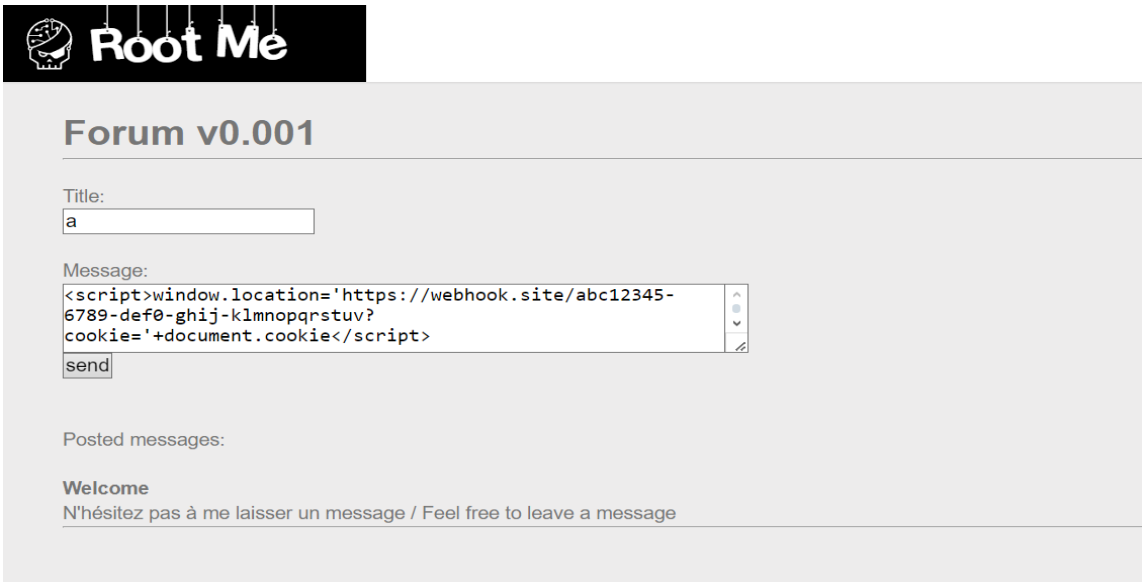
<script>alert('XSS')</script>

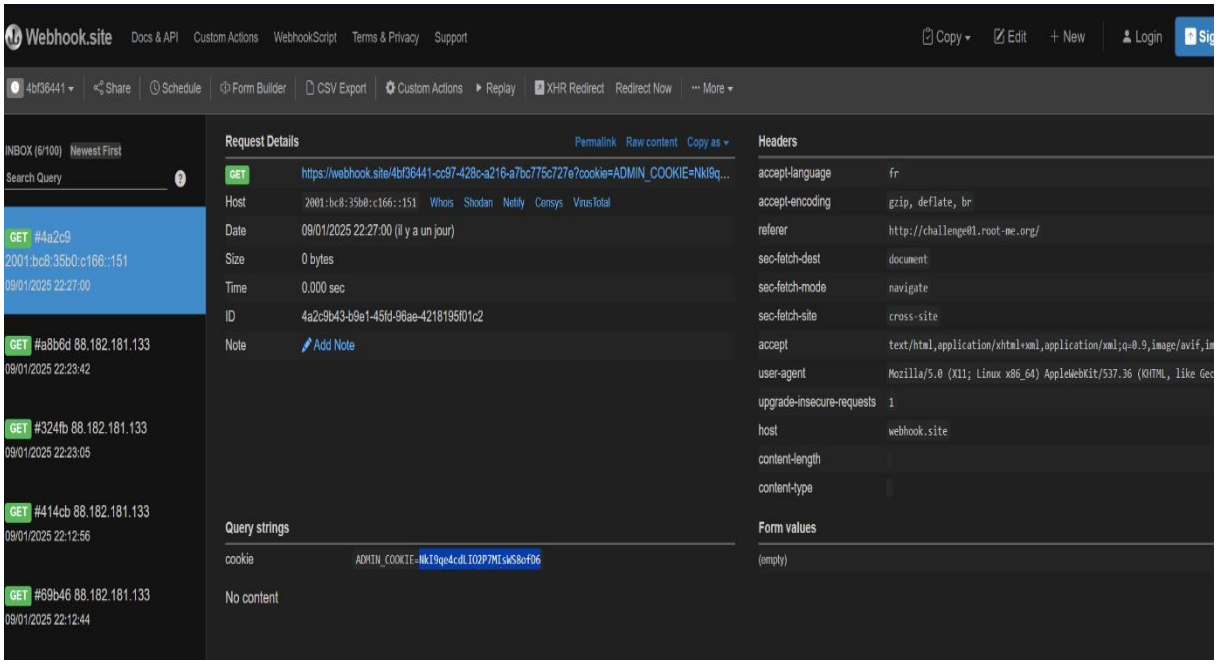
- Soumettez le formulaire. Si une fenêtre d'alerte s'affiche lors du rechargement de la page, cela signifie que le site est vulnérable au XSS stocké.

Attendre l'exécution par l'administrateur

- Une fois que l'administrateur consulte la page contenant votre script, son navigateur exécute le script et envoie ses cookies de session à votre URL Webhook.site.

- Vérifiez les logs sur Webhook.site pour récupérer les cookies.





Conclusion

Le défi **XSS - Stockée 1** est un excellent moyen d'apprendre à repérer et exploiter les vulnérabilités XSS stockées. Cette attaque est courante et peut être évitée avec de bonnes pratiques de sécurité.

HTTP - Cookies

Lien : Challenge HTTP - Cookies
Score : 20 points

Description :

> **Indice** : Il semblerait que Bob, l'administrateur, ait une passion pour les cookies...

En interceptant la réponse envoyée lors de la soumission d'un email dans le formulaire, on peut observer un commentaire spécifique. Et pour sa on vas les voir dans burpsuite.

	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
01.root-me.o...	GET	/web-serveur/ch7/?c=visiteur	✓		200	563	HTML					212.129.38.224
01.root-me.o...	GET	/favicon.ico			404	316	HTML	ico	404 Not Found			212.129.38.224
01.root-me.o...	GET	/web-serveur/ch7/?c=visiteur	✓		200	563	HTML					212.129.38.224
01.root-me.o...	POST	/web-serveur/ch7/?c=visiteur	✓		200	584	HTML					212.129.38.224
e01.root-me....	GET	/web-serveur/ch7/?c=admin	✓		401	766	HTML		401 Authorization Req...		✓	212.129.38.224
01.root-me.o...	GET	/web-serveur/ch7/?c=admin	✓		200	536	HTML					212.129.38.224
01.root-me.o...	POST	/web-serveur/ch7/?c=visiteur	✓		200	584	HTML					212.129.38.224
01.root-me.o...	POST	/web-serveur/ch7/?c=visiteur	✓		200	584	HTML					212.129.38.224
01.root-me.o...	POST	/web-serveur/ch7/?c=visiteur	✓		200	584	HTML					212.129.38.224
e01.root-me....	GET	/web-serveur/ch7/?c=admin	✓		401	766	HTML		401 Authorization Req...		✓	212.129.38.224
01.root-me.o...	GET	/web-serveur/ch7/?c=admin	✓		200	536	HTML					212.129.38.224
01.root-me.o...	POST	/web-serveur/ch7/?c=visiteur	✓		200	584	HTML					212.129.38.224
01.root-me.o...	POST	/web-serveur/ch7/?c=visiteur	✓		200	584	HTML					212.129.38.224

Voici ce qu'on obtient de lors des requet envoyé en et reçus

Request

Pretty Raw Hex

```
1 POST /web-serveur/ch7/?c=visiteur HTTP/1.1
2 Host: challenge01.root-me.org
3 Content-Length: 18
4 Cache-Control: max-age=0
5 Accept-Language: en-US,en;q=0.9
6 Origin: http://challenge01.root-me.org
7 Content-Type: application/x-www-form-urlencoded
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
  x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/131.0.6778.86 Safari/537.36
10 Accept:
  text/html,application/xhtml+xml,application/xml;q
  =0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
  pplication/signed-exchange;v=b3;q=0.7
11 Referer:
  http://challenge01.root-me.org/web-serveur/ch7/?c
  =visiteur
12 Accept-Encoding: gzip, deflate, br
13 Cookie: ch7=visiteur; _ga=
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: nginx
3 Date: Sat, 11 Jan 2025 11:08:55 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: keep-alive
6 Vary: Accept-Encoding
7 Set-Cookie: ch7=visiteur
8 Content-Length: 379
9
10
11 <br/>
12 <br/>
13 <fieldset>
14
15   <form method="POST" action="" name="a">
16     Email<br/>
17     <input type="text" name="mail" size="20" class=
       "post2" value="">
       <br/>
```

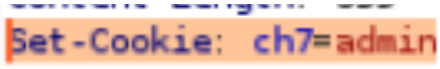
En se basant sur l'indice, on modifie la réponse pour nous attribuer le cookie correspondant, puis on envoie à nouveau un formulaire de manière normale en faisant une injection.

oot-me.o...	GET	/web-serveur/ch7/?c=admin	✓	200	536	HTML

On peux voir maintenant que les cookie on changé :

Avant :

Après :



Enfin, nous récupérons le *flag* : **ml-SYMPA**.

File upload - Null byte

Énoncé

Votre objectif est de compromettre cette galerie photo en y uploadant du code PHP.

Comprendre le "null byte"

Un "null byte" (%00 ou \0) permet de contourner les filtres de sécurité en séparant une extension interdite d'une extension autorisée. Par exemple, en injectant %00 entre .php et .png, le serveur ignore .php mais garde .png comme extension, ce qui permet de contourner les restrictions de type de fichier.

2. Injection dans la requête d'upload

Pour exploiter cette vulnérabilité, modifiez la requête d'upload en insérant un "null byte" dans le nom du fichier. Exemple :

bash

Copier le code

Content-Disposition: form-data; name="file"; filename="shell.php%00.png"

Content-Type: application/x-php

3. Exploitation du fichier téléchargé

Une fois le fichier téléchargé, on peut y accéder pour l'exploiter. Le serveur exécutera le fichier PHP déguisé en .png, se qui nous permetd'obtenir un shell et de prendre le contrôle du serveur. Par exemple, vous pouvez y accéder via l'URL suivante :

http://challenge01.root-me.org/web-serveur/ch22/galerie/upload/3cbefe99062d8c49ac5d38709bc83e8a/shell.php

4. Validation du challenge

Après avoir pris le contrôle du Shell, on peut récupérer le mot de passe qui nous permettra de valider le challenge. Exemple de message de succès :

Well done! You can validate this challenge with the password: YPNchi2NmTwygr2dgCCF

CONCLUSIONS

Cette SAE a été une expérience vraiment intéressante, qui m'a permis de progresser en cybersécurité à travers différents challenges pratiques. J'ai notamment travaillé sur l'analyse de vulnérabilités web et la programmation, ce qui m'a aidé à découvrir et à utiliser des outils comme Burp Suite. Certains exercices, en particulier ceux en programmation, m'ont donné du fil à retordre, mais ils m'ont permis de mieux comprendre mes points faibles et de voir où je dois encore progresser. À côté de ça, j'ai aussi pu mettre en avant certaines de mes forces, comme ma capacité à analyser et résoudre des problèmes plus concrets. Cette SAE m'a vraiment aidé à mieux cerner mes compétences et les domaines où je peux encore m'améliorer.