

IMT MINES ALÈS - SITE CLAVIÈRES

DÉPARTEMENT SYSTÈMES ET RÉSEAUX (SR)

Ethical Hacking - TryHackMe BurpSuite

Nathan MARTEL

Groupe : SR
IMT Mines ALÈS

Table des matières

1 Introduction	2
2 Burp Suite : Repeater	3
2.1 Task 1 : Introduction	3
2.2 Task 2 : What is Repeater ?	4
2.3 Task 3 : Basic Usage	4
2.4 Task 4 : Message Analysis Toolbar	4
2.5 Task 5 : Inspector	4
2.6 Task 6 : Practical Example	5
2.7 Task 7 : Challenge	7
2.8 Task 8 : Extra-mile Challenge	11
2.9 Task 9 : Conclusion	15
3 Conclusion	16

1 Introduction :

Le challenge Burp Suite Repeater de TryHackMe permet d'avoir un aperçu approfondi dans l'utilisation de Burp Suite, un outil incontournable pour l'analyse de sécurité des applications web. Ce dernier permet de se concentrer sur l'utilisation du module Repeater de Burp Suite, un outil clé pour tester et manipuler les requêtes HTTPS.

URL du challenge : <https://tryhackme.com/r/room/burpsuiterepeater>

@uthor : Nathan Martel.




Le document est classifié sous la marque **TLP :RED** (Traffic Light Protocol), ce qui signifie que le partage du document doit se limiter uniquement aux destinataires individuels, et qu'aucune autre divulgation n'est autorisée sauf avis favorable du propriétaire.

Ce document est privé et est uniquement déposé dans le répertoire Git de l'auteur. Merci de ne pas le diffuser, l'utiliser ou le modifier sans autorisation.

2 Burp Suite : Repeater :

2.1 Task 1 : Introduction :

Aucune réponse n'est demandée pour cette partie.

Task 1  Introduction  

Welcome to the Burp Suite Repeater room!

▶ Start Machine

In this room, we will explore the advanced capabilities of the Burp Suite framework by focusing on the Burp Suite Repeater module. Building upon the foundational knowledge covered in the [Burp Basics room](#), we will delve into the powerful features of the Repeater tool. You will learn how to manipulate and resend captured requests, and we will explore the various options and functionalities available in this exceptional module. Throughout the room, we will provide practical examples, including a real-world exercise, to solidify your understanding of the concepts discussed.

If you are new to [Burp Suite](#) or have not completed the Burp Basics room, we recommend doing so before proceeding. The Burp Basics room establishes the fundamental knowledge necessary for this room and will enhance your learning experience.

Deploy the target VM attached to this task by pressing the green **Start Machine** button. Also, start the AttackBox by pressing the blue **Start AttackBox** button at the top of this room if you are not using your own machine. Then, start Burp and follow along with the next tasks.

Answer the questions below

Let's get started!

No answer needed

✓ Correct Answer

FIGURE 2.1 – Capture Task1

2.2 Task 2 : What is Repeater? :

Which sections gives us a more intuitive control over our requests?

⇒ La réponse est **Inspector**.

L'inspector permet d'analyser et de modifier les requêtes avec une interface propre à Burp qui est plus accessible que l'éditeur brut. Il affiche les différentes parties d'une requête (comme les en-têtes, les paramètres, le corps, les cookies, etc.). On peut aussi éditer directement les champs spécifiques de la requête (e.g. changer une valeur de paramètre ou modifier un cookie) sans avoir à naviguer dans tout le texte brut.

2.3 Task 3 : Basic Usage :

Which view will populate when sending a request from the Proxy module to Repeater?

⇒ La réponse est **Request**.

Lorsqu'une requête est envoyée depuis le module Proxy vers le Repeater, elle s'affiche dans la Request view. On peut visualiser et modifier tous les détails de la requête avant de l'envoyer au serveur pour analyse ou test. En modifiant une requête et en la renvoyant, on peut observer comment le serveur réagit à différents changements pour identifier des vulnérabilités ou comprendre le comportement d'une application.

2.4 Task 4 : Message Analysis Toolbar :

Which option allows us to visualize the page as it would appear in a web browser?

⇒ La réponse est **Render**.

Cette option permet de visualiser la réponse telle qu'elle apparaît dans un navigateur WEB (images, styles, etc.). Elle peut être utile pour examiner le rendu visuel d'une page lorsqu'il est nécessaire de vérifier son apparence ou de tester des éléments interactifs.

2.5 Task 5 : Inspector :

Which section in Inspector is specific to POST requests?

⇒ La réponse est **Body Parameters**.

Cette section affiche les données envoyées avec une requête POST. Elle permet de visualiser et de modifier les données avant de renvoyer la requête au serveur. C'est donc utile pour tester des scénarios où des données sont envoyées dans le corps de la requête, comme dans des formulaires ou des APIs.

2.6 Task 6 : Practical Example :

What is the flag you receive ?

Après avoir capturé la requête, je l'envoie vers le Repeater où je vais pouvoir injecter un en-tête appelé FlagAuthorised pour par la suite la renvoyer au serveur :

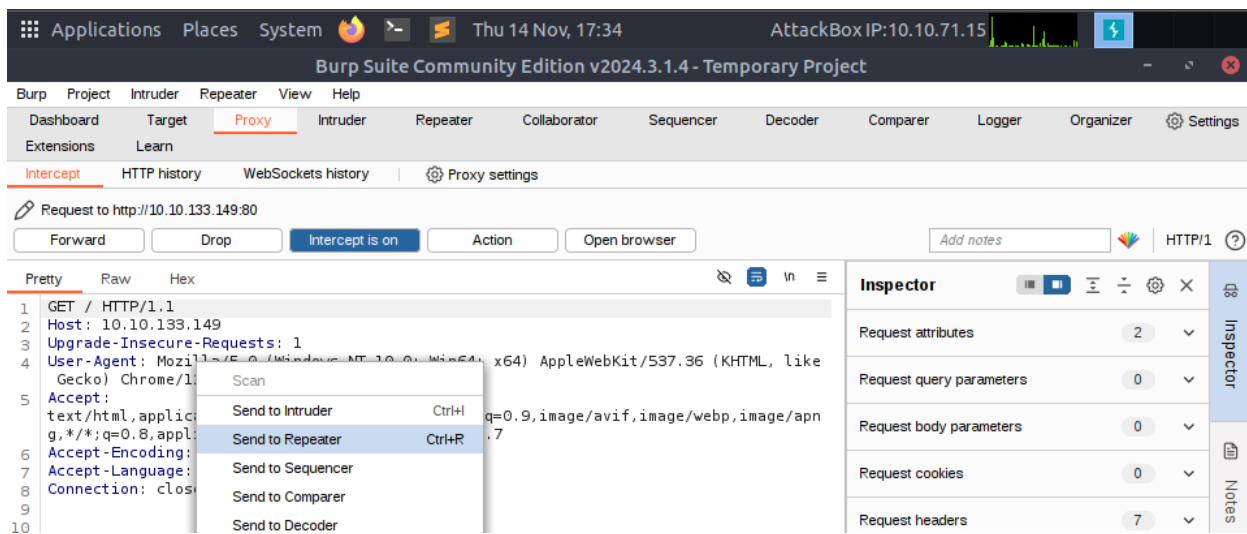


FIGURE 2.2 – Requête envoyée vers le Repeater

Je modifie alors la requête en ajoutant l'en-tête FlagAuthorised avec l'inspector :

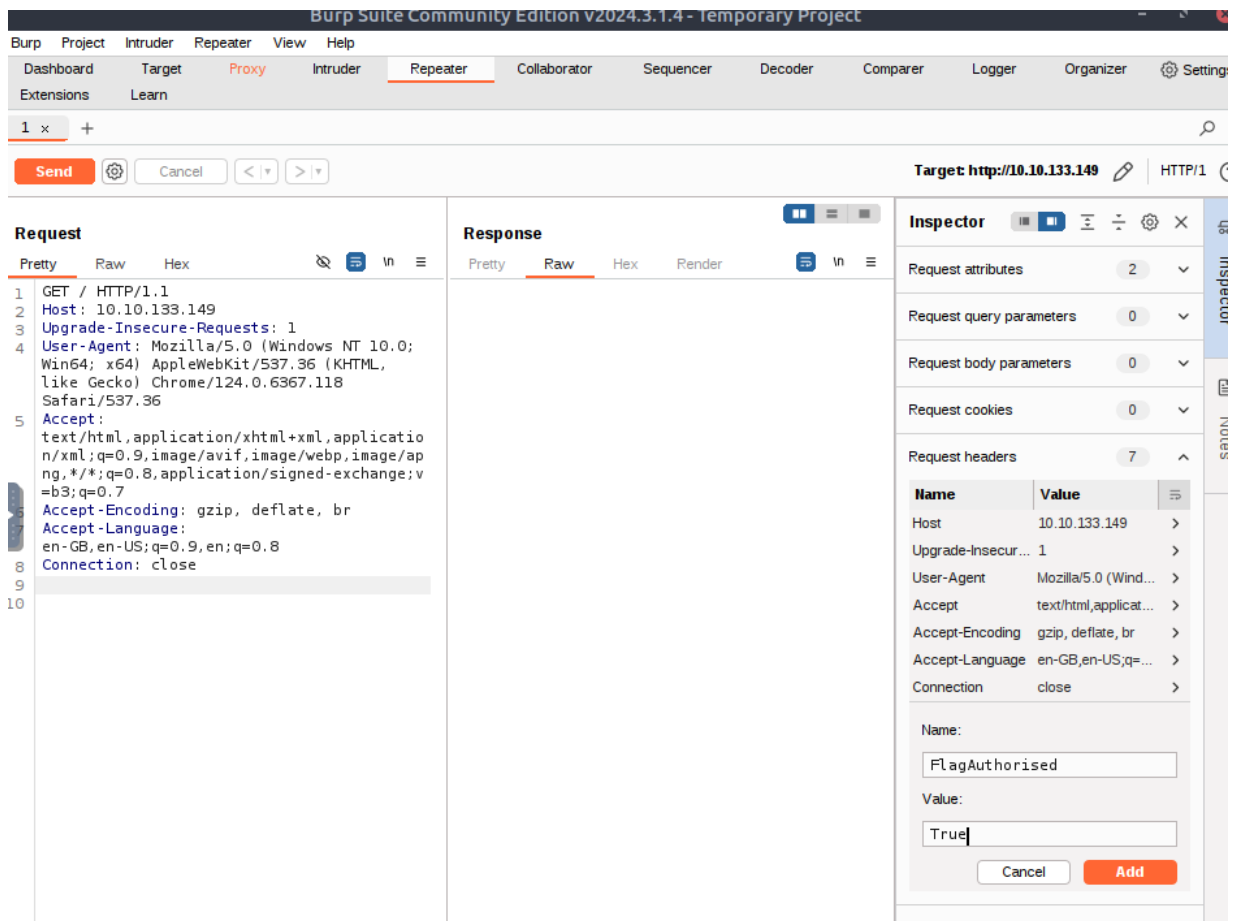


FIGURE 2.3 – Modification de l'en-tête avec le FlagAuthorised

Une fois que l'en-tête a été injecté dans la requête, je renvoie celle-ci au serveur et j'obtiens le flag suivant :

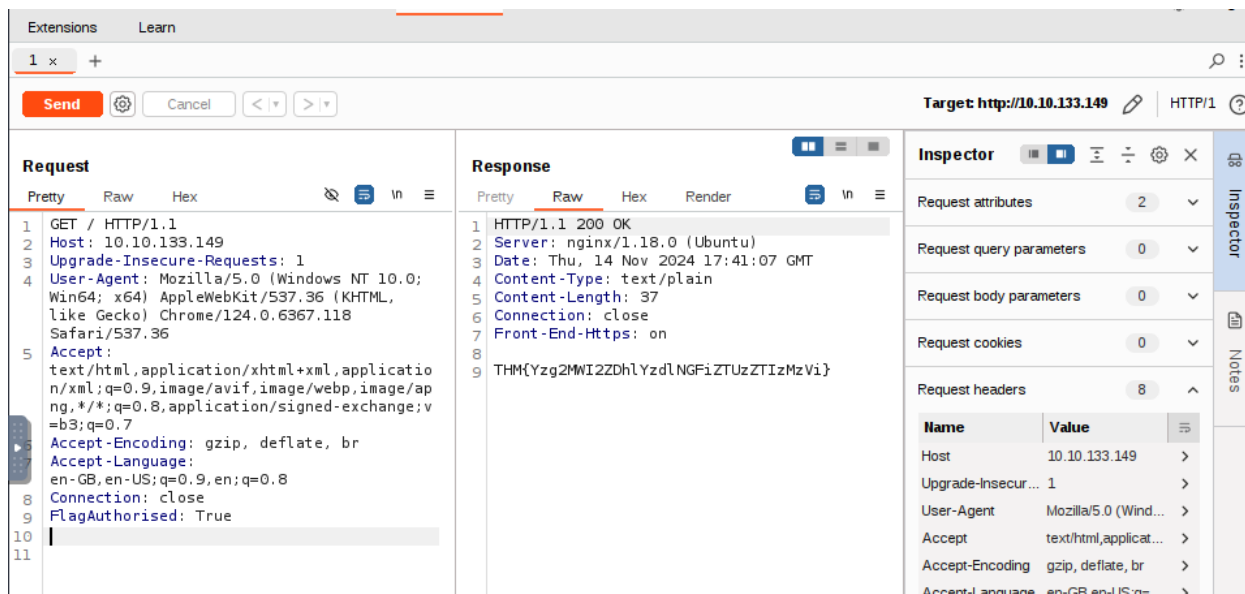


FIGURE 2.4 – Récupération du flag pour la tâche 2.6

⇒ La réponse est **THM{Yzg2MWI2ZDhlYzdlnGFiTUZZTIZMzVi}**.

2.7 Task 7 : Challenge :

Avec l'interception désactivée, j'accède à la page suivante :

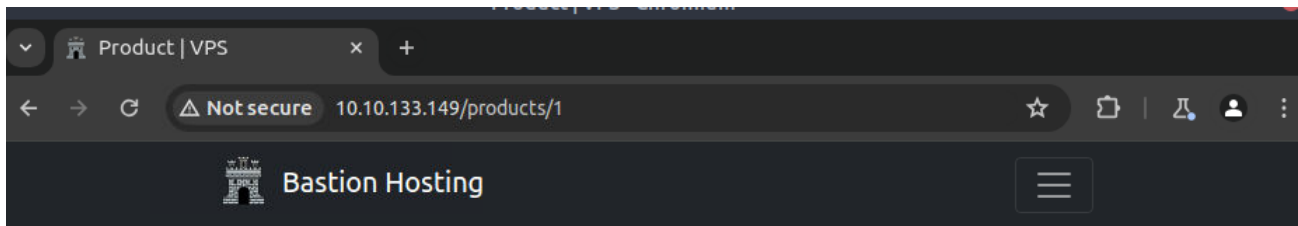


FIGURE 2.5 – Accès avec le navigateur de Burp à la page products/1

Aucune réponse n'est demandée pour la première question de cette partie.

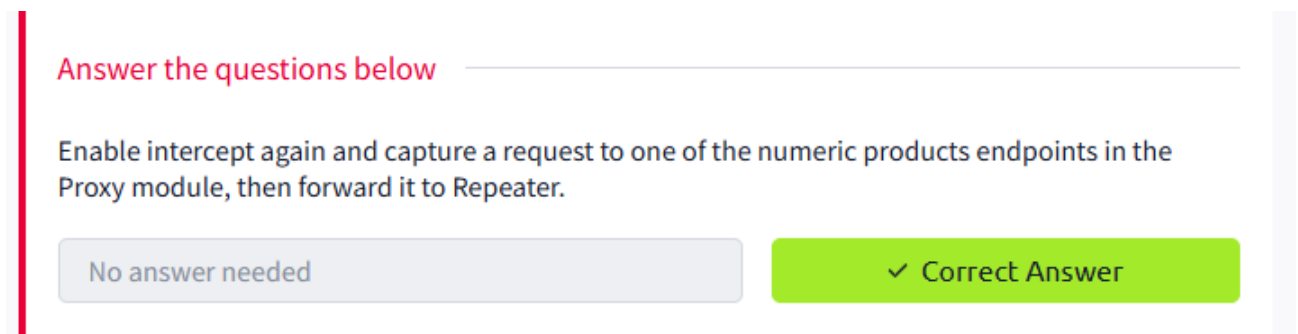


FIGURE 2.6 – Capture Task7 partie 1

Ensuite, j'active l'interception et je requête l'URL suivante : `http://10.10.133.149/products/1`.

J'envoie la requête au Repeater :

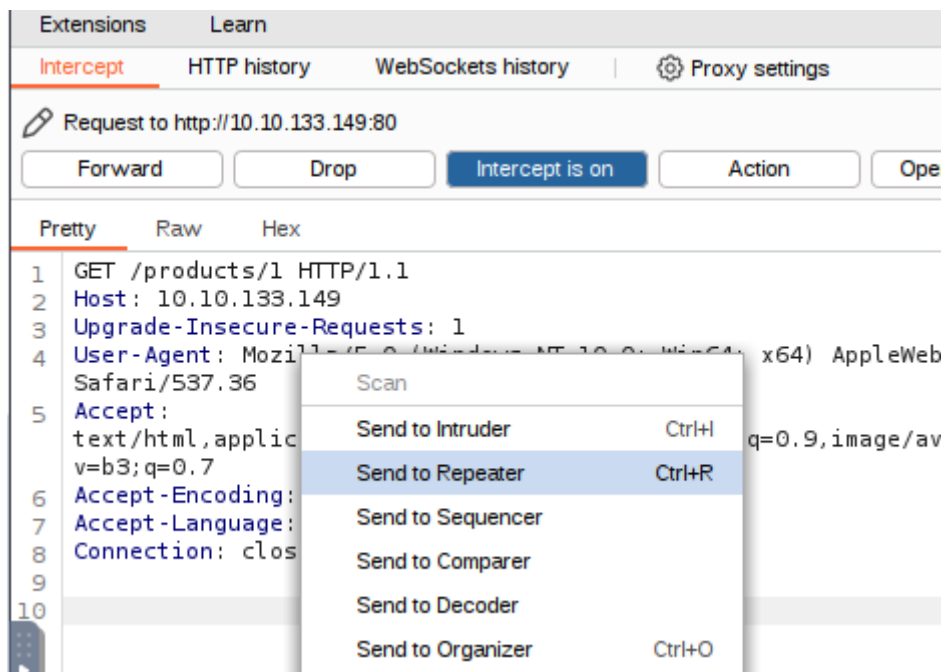


FIGURE 2.7 – Envoi de la requête au Repeater

Il nous est demandé ensuite de modifier le numéro de fin de la requête en entrées extrêmes. Alors, dans le repeater, je change le 1 en 3000.

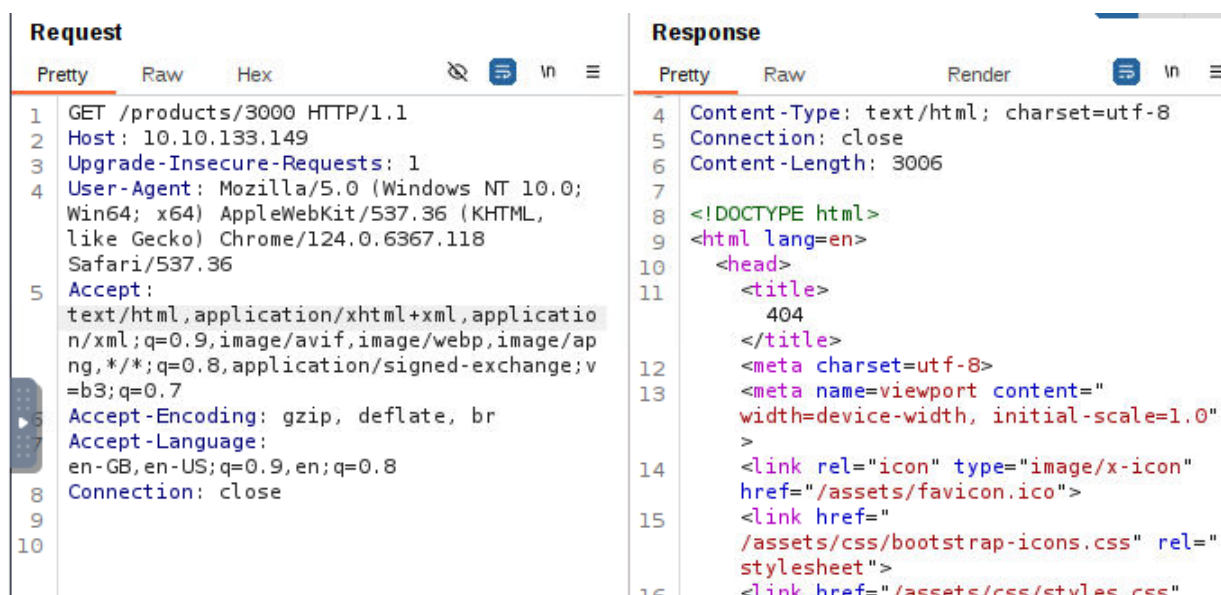


FIGURE 2.8 – Réponse du serveur pour la page /products/3000

Il faut que le serveur génère un code « 500 Internal Server Error » or ici, je n'ai qu'une erreur 404. Je change donc le numéro de fin de requête en changeant 3000 en texte (test) :

Request		Response	
Pretty	Raw	Pretty	Raw
1	GET /products/test HTTP/1.1	1	HTTP/1.1 404 NOT FOUND
2	Host: 10.10.133.149	2	Server: nginx/1.18.0 (Ubuntu)
3	Upgrade-Insecure-Requests: 1	3	Date: Thu, 14 Nov 2024 17:49:46 GMT
4	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36	4	Content-Type: text/html; charset=utf-8
5	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7	5	Connection: close
6	Accept-Encoding: gzip, deflate, br	6	Content-Length: 3006
7	Accept-Language: en-GB,en-US;q=0.9,en;q=0.8	7	
8	Connection: close	8	<!DOCTYPE html>
9		9	<html lang=en>
10		10	<head>
		11	<title>
			404
			</title>
		12	<meta charset=utf-8>
		13	<meta name=viewport content="width=device-width, initial-scale=1.0">
		14	<link rel="icon" type="image/x-icon" href="/assets/favicon.ico">
		15	<link href="/assets/css/bootstrap-icons.css" rel="stylesheet">

FIGURE 2.9 – Réponse du serveur pour la page /products/test

Avec un GET de /products/test, j'obtiens encore une erreur 404. Je décide alors de remplacer le texte (test) par un chiffre négatif : par -1 :

Request		Response		Inspector
Pretty	Raw	Pretty	Raw	
1	GET /products/-1 HTTP/1.1	1	HTTP/1.1 500 INTERNAL SERVER ERROR	Request attributes 2
2	Host: 10.10.133.149	2	Server: nginx/1.18.0 (Ubuntu)	Request query parameters 0
3	Upgrade-Insecure-Requests: 1	3	Date: Thu, 14 Nov 2024 17:50:07 GMT	Request body parameters 0
4	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36	4	Content-Type: text/html; charset=utf-8	Request cookies 0
5	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7	5	Content-Length: 3034	Request headers 7
6	Accept-Encoding: gzip, deflate, br	6	Connection: close	Response headers 5
7	Accept-Language: en-GB,en-US;q=0.9,en;q=0.8	7		
8	Connection: close	8	<!DOCTYPE html>	
9		9	<html lang=en>	
10		10	<head>	
		11	<title>	
			500	
			</title>	
		12	<meta charset=utf-8>	
		13	<meta name=viewport content="width=device-width, initial-scale=1.0">	
		14	<link rel="icon" type="image/x-icon" href="/assets/favicon.ico">	
		15	<link href="/assets/css/bootstrap-icons.css" rel="stylesheet">	

FIGURE 2.10 – Réponse du serveur pour la page /products/-1

Et j'obtiens une erreur « 500 Internal Server Error ». En aiguisant la réponse du serveur, on retrouve le flag causé par l'erreur 500 :

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Request' pane on the left displays a GET request to /products/. The 'Response' pane on the right shows an HTML response. The response contains a navigation menu with a link to /products/ and a section titled 'features' which contains a code block with the flag THM{N2MzMzFhMTAlMmZiYjA2YWQ4M2ZmMzhl}.

```

Request
1 GET /products/-1 HTTP/1.1
2 Host: 10.10.133.149
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
8 Connection: close
9
10

Response
31 /ticket/">
32 Support
33 </a>
34 </li>
35 <li class="nav-item">
36 <a class="nav-link" href="/products/">
37 Products
38 </a>
39 </li>
40 </ul>
41 </div>
42 </div>
43 </nav>
44 <section class="py-5" id="features">
45 <div class="container px-5 my-5">
46 <div class="text-center mb-5">
47 <h1 class="jumbotron">
48 500
49 </h1>
50 <h2>
51 <code>
52 THM{N2MzMzFhMTAlMmZiYjA2YWQ4M2ZmMzhl}
53 </code>
54 </h2>
55 </div>
56 </div>
57 </section>
58 </div>

```

FIGURE 2.11 – Récupération du flag pour la tâche 2.7

⇒ La réponse est **THM{N2MzMzFhMTAlMmZiYjA2YWQ4M2ZmMzhl}**.

2.8 Task 8 : Extra-mile Challenge :

Dans cette partie, j'ai simplement suivi les étapes de la tâche.

Le défi est d'identifier et d'exploiter une vulnérabilité d'injection SQL Union présente dans le paramètre ID de la page (« /about/ID »). En capturant le trafic, je requête tout d'abord la page « /about/ID ». J'obtiens alors la requête dans mon interface Burp que j'envoie au Repeater.

Ensuite, la deuxième étape : il nous est demandé de confirmer qu'une vulnérabilité existe. Pour cela, on ajoute une seule apostrophe « ' ». Elle permet de provoquer une erreur du serveur lorsqu'un simple SQLi est présent :

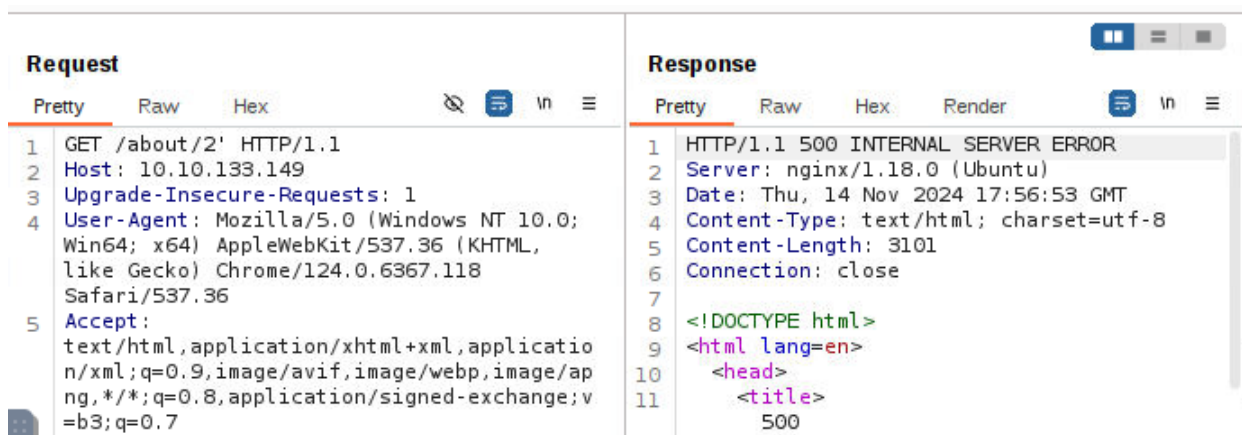


FIGURE 2.12 – Ajout d'un apostrophe pour provoquer une erreur. Vérification de la présence d'une vulnérabilité

On remarque la réponse du serveur : « 500 Internal Server Error ». Dans la réponse du serveur, on remarque la requête SQL exécutée :

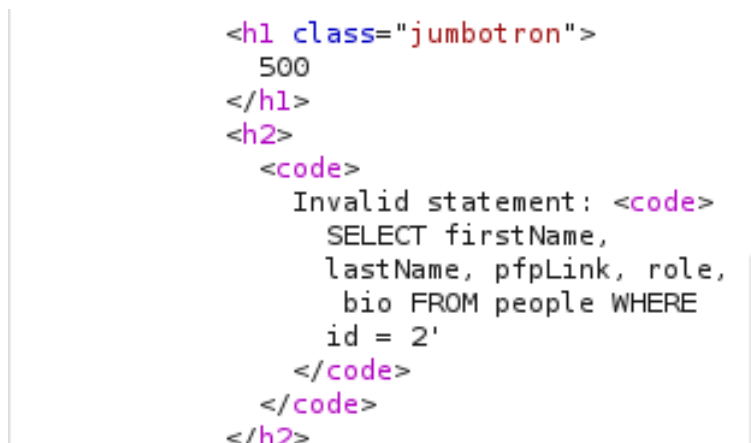


FIGURE 2.13 – Requête SQL exécutée par le serveur

On sait alors que la table que l'on requête s'appelle « people ». La requête sélectionne 5 colonnes dans la table : « firstName, lastName, pfpLink, role et bio ».

Avec le nom de la table et le nombre de lignes, je peux utiliser une requête avec le mot-clé UNION pour sélectionner les noms des colonnes de la table « people » à partir de la table « columns » dans la base de données par défaut information_schema.

Voici la requête injectée au serveur : /about/0 UNION ALL SELECT column_name,null,null,null,null FROM information_schema.columns WHERE table_name="people" :

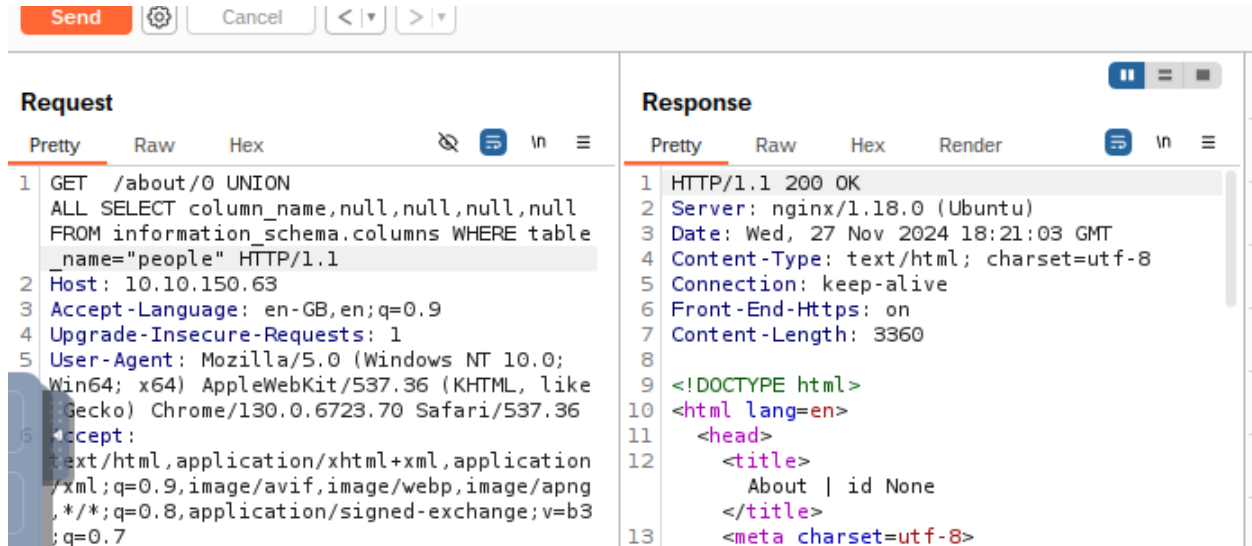


FIGURE 2.14 – Requête SQL pour sélectionner les noms des colonnes de la table « people » à partir de la table « columns » dans la base de données

Cela crée une requête UNION et sélectionne la cible, puis quatre colonnes nulles (pour éviter que la requête ne génère une erreur). De plus, l'ID est fixé à 0 pour s'assurer de ne rien récupérer avec la requête d'origine, celle légitime.

Le serveur répond favorablement à la requête et dans la réponse du serveur, on remarque que le nom de la première colonne (ID) a été inséré dans le titre de la page :

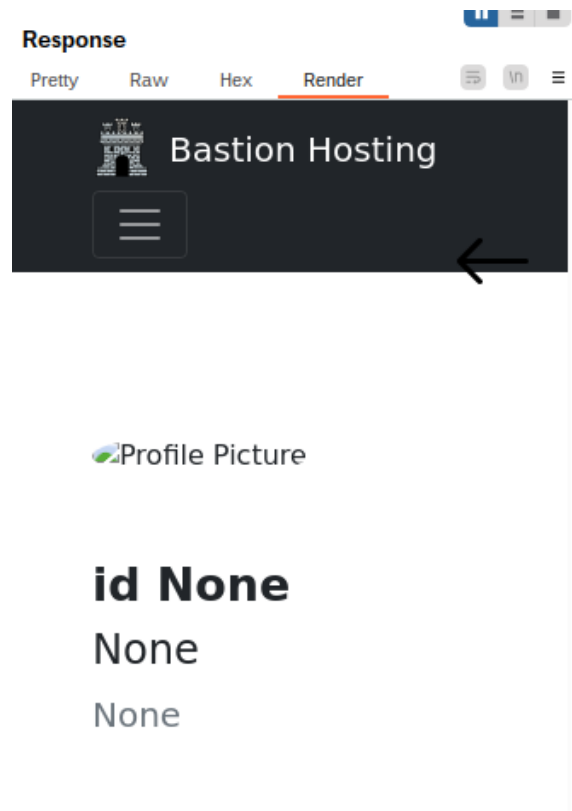


FIGURE 2.15 – Le nom de la première colonne (ID) a été inséré dans le titre de la page

Le premier élément s'affiche sur la page. Maintenant, on peut utiliser cette injection SQL pour regrouper les résultats : avec la fonction « group_concat() », on peut fusionner tous les noms de colonnes en une seule sortie :

Voici la requête injectée au serveur : `/about/0 UNION ALL SELECT group_concat(column_name),null,null,null,null FROM information_schema.columns WHERE table_name="people"`

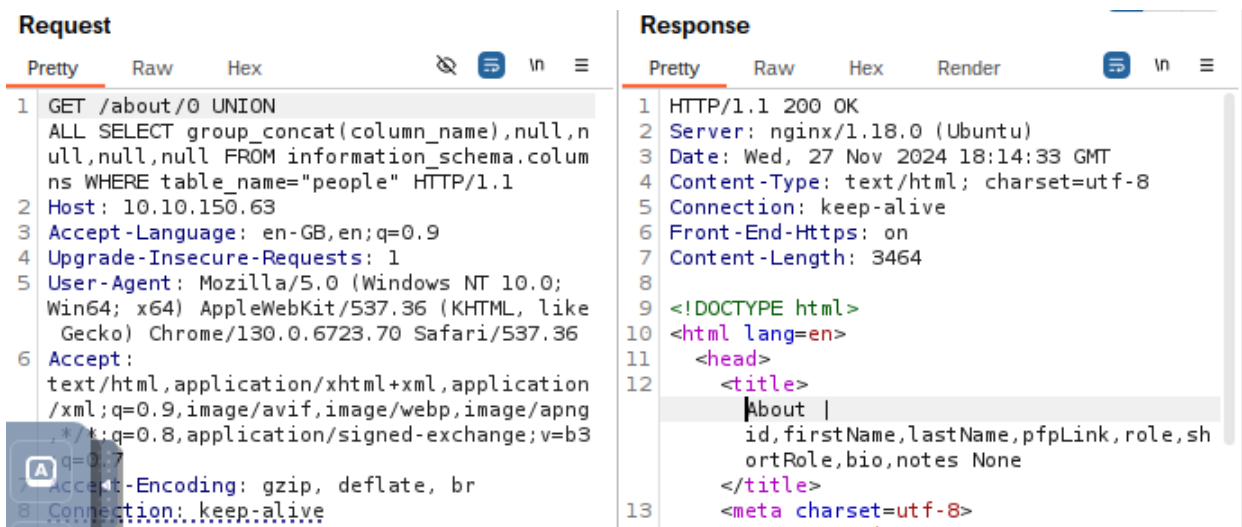


FIGURE 2.16 – Requête SQL pour regrouper les résultats : noms de colonnes

La table contient 8 colonnes : id, id, firstName, lastName, pfpLink, role, shortRole, bio, et notes. On a alors le nom de la table (people), le nom de la colonne cible (notes), l'ID du PDG (1). Il suffit maintenant de créer une requête pour extraire l'indicateur

Voici la requête injectée au serveur : /about/0 UNION ALL SELECT notes,null,null,null,null FROM people WHERE id = 1

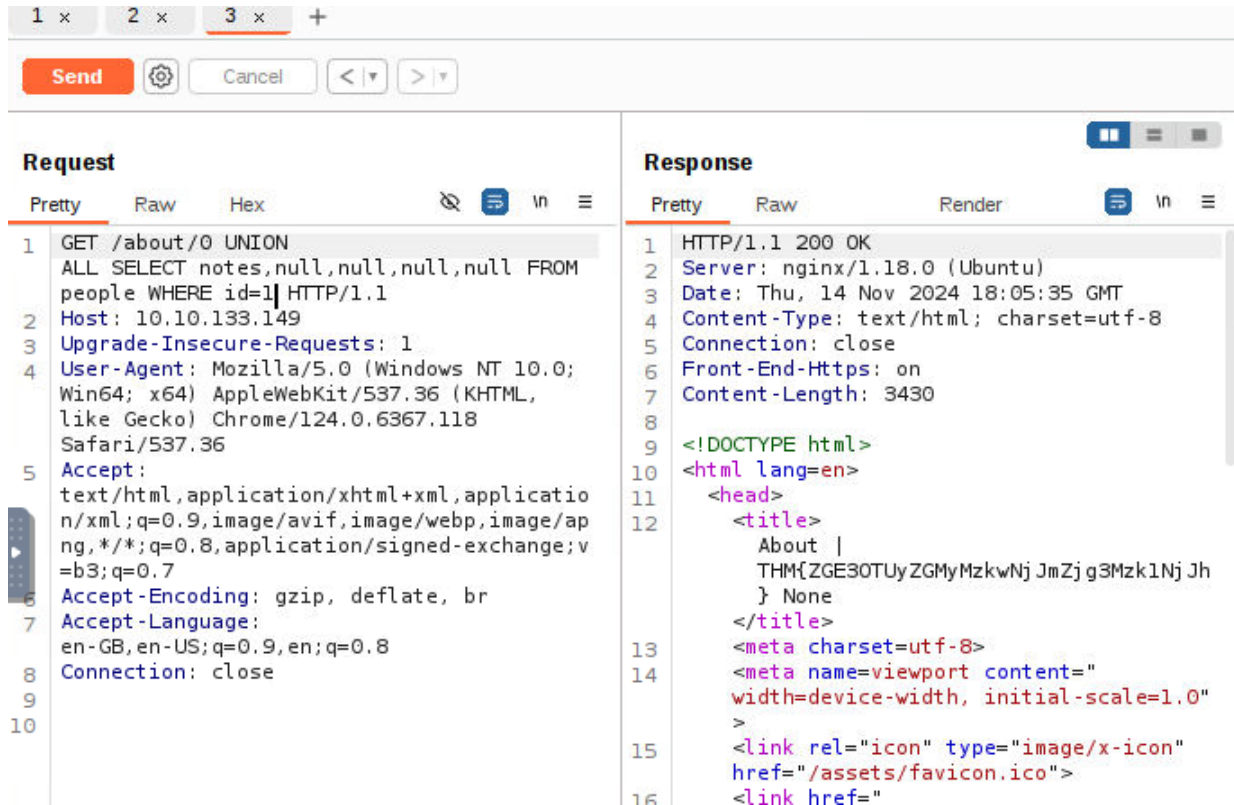


FIGURE 2.17 – Requête pour extraire le flag

On retrouve le flag dans la réponse du serveur.

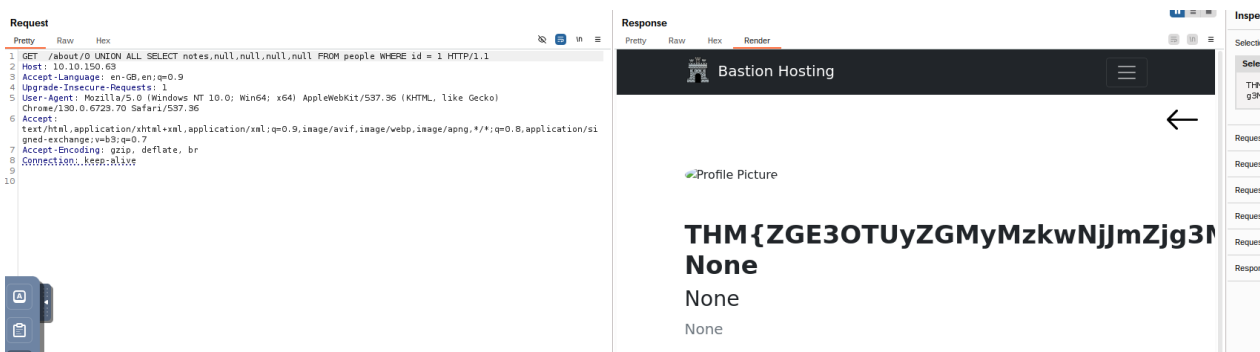




FIGURE 2.18 – Récupération du flag pour la tâche 2.7

⇒ La réponse est **THM{ZGE30TUyZGMyMzkwNjJmZjg3Mzk1NjJh}**.

2.9 Task 9 : Conclusion :

Aucune réponse n'est demandée pour cette partie.

Task 9  Conclusion 

Congratulations on completing the [Burp Suite Repeater](#) room!

By now, you should have a solid understanding of effectively utilising Repeater to edit, manipulate, and resend requests. Additionally, you should have gained insight into the numerous practical applications of this tool.

In the next room of the module, we will explore the [Burp Suite Intruder](#) module. Intruder allows for automated and customizable attacks, making it a powerful tool for various security testing scenarios.

Good luck with the next room, and enjoy exploring the capabilities of [Burp Suite Intruder](#)!

Answer the questions below

I can use Burp Suite Repeater!

FIGURE 2.19 – Capture Task9

3 Conclusion :

Ce challenge TryHackMe a permis d'avoir des notions autour de la suite Burp. Il permet, en particulier, d'utiliser le module Repeater qui est essentiel pour tester, manipuler et analyser des requêtes HTTPS. J'ai pu découvrir/revoir l'inspecteur pour faciliter l'analyse et la modification des requêtes, la Request view où sont visualisées les requêtes envoyées depuis le Proxy vers le Repeater et d'autres options comme le Render etc.

J'ai pu également comprendre comment exploiter une application avec l'injection SQL ou la manipulation d'en-têtes.

Fin du rapport.

Rapport écrit par Nathan Martel du 25/11/2024 au 27/11/2024.

Version : v1.0

Outils utilisés : VM TryHackMe et VM Kali Linux

Logiciel utilisé : Texworks

Langage et systèmes de composition : LaTeX

Console : MiKTeX

Format du document : PDF

Table des figures

2.1	Capture Task1	3
2.2	Requête envoyée vers le Repeater	5
2.3	Modification de l'en-tête avec le FlagAuthorised	6
2.4	Récupération du flag pour la tâche 2.6	6
2.5	Accès avec le navigateur de Burp à la page products/1	7
2.6	Capture Task7 partie 1	7
2.7	Envoi de la requête au Repeater	8
2.8	Réponse du serveur pour la page /products/3000	8
2.9	Réponse du serveur pour la page /products/test	9
2.10	Réponse du serveur pour la page /products/-1	9
2.11	Récupération du flag pour la tâche 2.7	10
2.12	Ajout d'un apostrophe pour provoquer une erreur. Vérification de la présence d'une vulnérabilité	11
2.13	Requête SQL exécutée par le serveur	11
2.14	Requête SQL pour sélectionner les noms des colonnes de la table « people » à partir de la table « columns » dans la base de données .	12
2.15	Le nom de la première colonne (ID) a été inséré dans le titre de la page	13
2.16	Requête SQL pour regrouper les résultats : noms de colonnes	13
2.17	Requête pour extraire le flag	14
2.18	Récupération du flag pour la tâche 2.7	14
2.19	Capture Task9	15