
# SERVER-SIDE ATTACKS 101

PHREAKS 2600

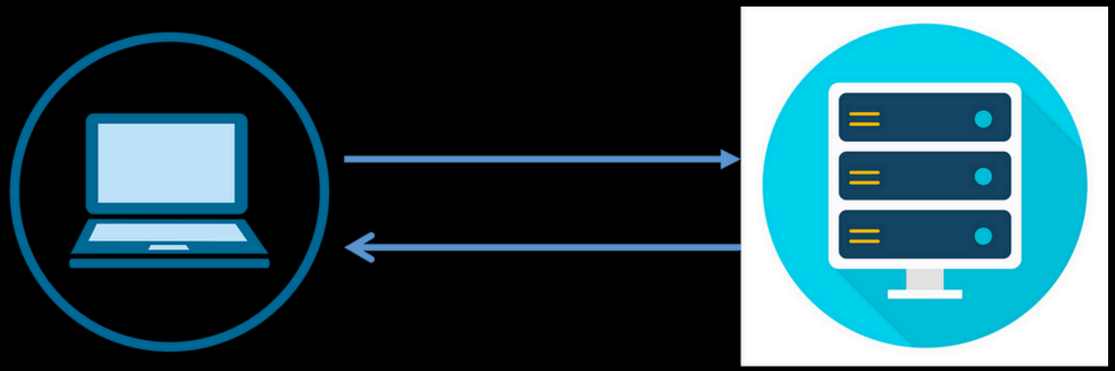
PRÉSENTÉ PAR
FÉLIX BILLIÈRES

# Les attaques Web côté serveur (Server-Side Attacks)

- Qu’est-ce qu’une attaque côté serveur ?
- Pourquoi s’y intéresser ?
- Objectif du cours : comprendre et exploiter certaines vulnérabilités serveur

# Le rôle d’un serveur web

- Un serveur web reçoit des requêtes HTTP des clients (navigateur, API)
- Il interagit avec des bases de données, d’autres services, et retourne une réponse
- **Exemple classique :**
  - Un utilisateur demande une page (GET /index.html)
  - Le serveur génère ou récupère la page et l’envoie



# Enjeux et risques des attaques côté serveur

## 🎯 Cibles principales :

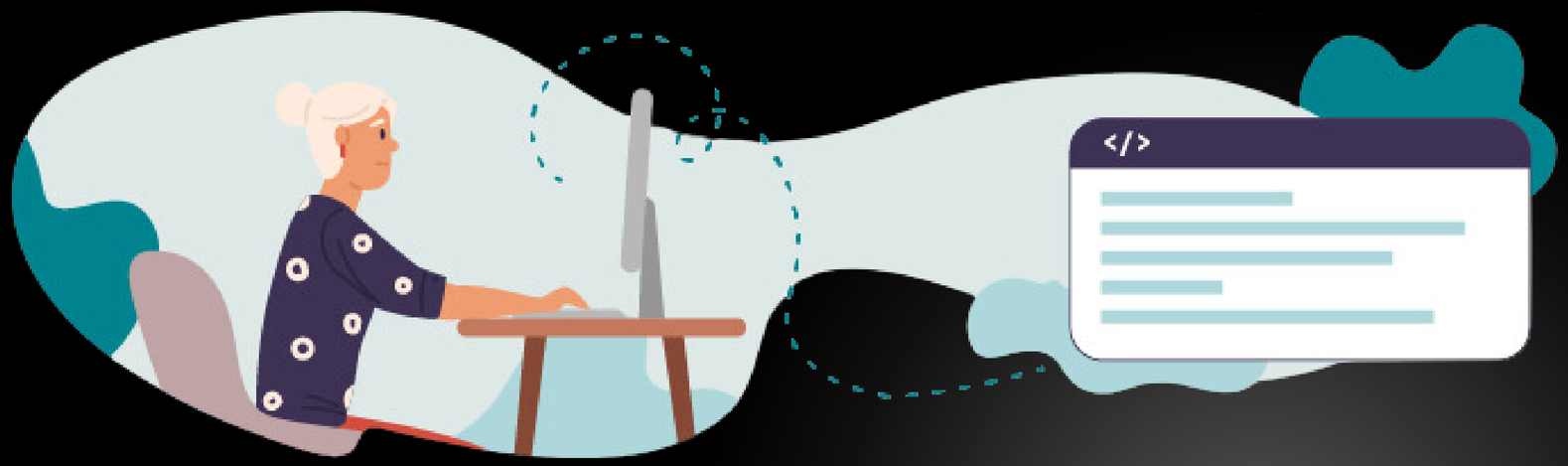
- Bases de données
- Fichiers sensibles sur le serveur
- Accès réseau interne
- Exécution de code malveillant

## ⚠ Conséquences :

- Vol de données (ex: fuite de données sensibles)
- Compromission du serveur (ex: prise de contrôle)
- Escalade de privilèges (ex: mouvement latéral sur le réseau interne)

# Origine des failles côté serveur

- **Entrées utilisateur non filtrées** → Injection possible
- **Fonctionnalités mal sécurisées** → Accès non prévu
- **Mauvaise configuration serveur** → Exploitation facile
- **Utilisation de technologies obsolètes** → Failles connues exploitées




🛠️ **EXEMPLE** : UN SERVICE PERMETTANT D'IMPORTER UN FICHIER DEPUIS UNE URL SANS VÉRIFIER LA SOURCE → **SSRF**

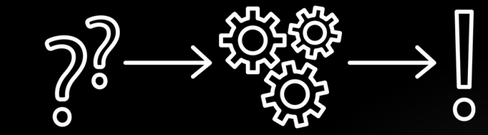
# Les vulnérabilités que nous allons voir

📌 **SSRF (Server-Side Request Forgery)**
→ Le serveur fait des requêtes vers des ressources non prévues

📌 **SSI (Server-Side Includes)**
→ Exécution de commandes via des inclusions côté serveur

📌 **SSTI (Server-Side Template Injection)**
→ Injection dans les moteurs de templates

📌 **XSLT Injection**
→ Exploitation du traitement XML



# SSRF - Server-Side Request Forgery

L’attaquant force le serveur à faire une requête HTTP pour lui, souvent vers des ressources internes

🔹 **Exemples d’utilisation courante :**

✅ Prévisualisation d’URL (ex: Twitter)

✅ Webhooks (ex: intégration API)

✅ Importation d’images via URL

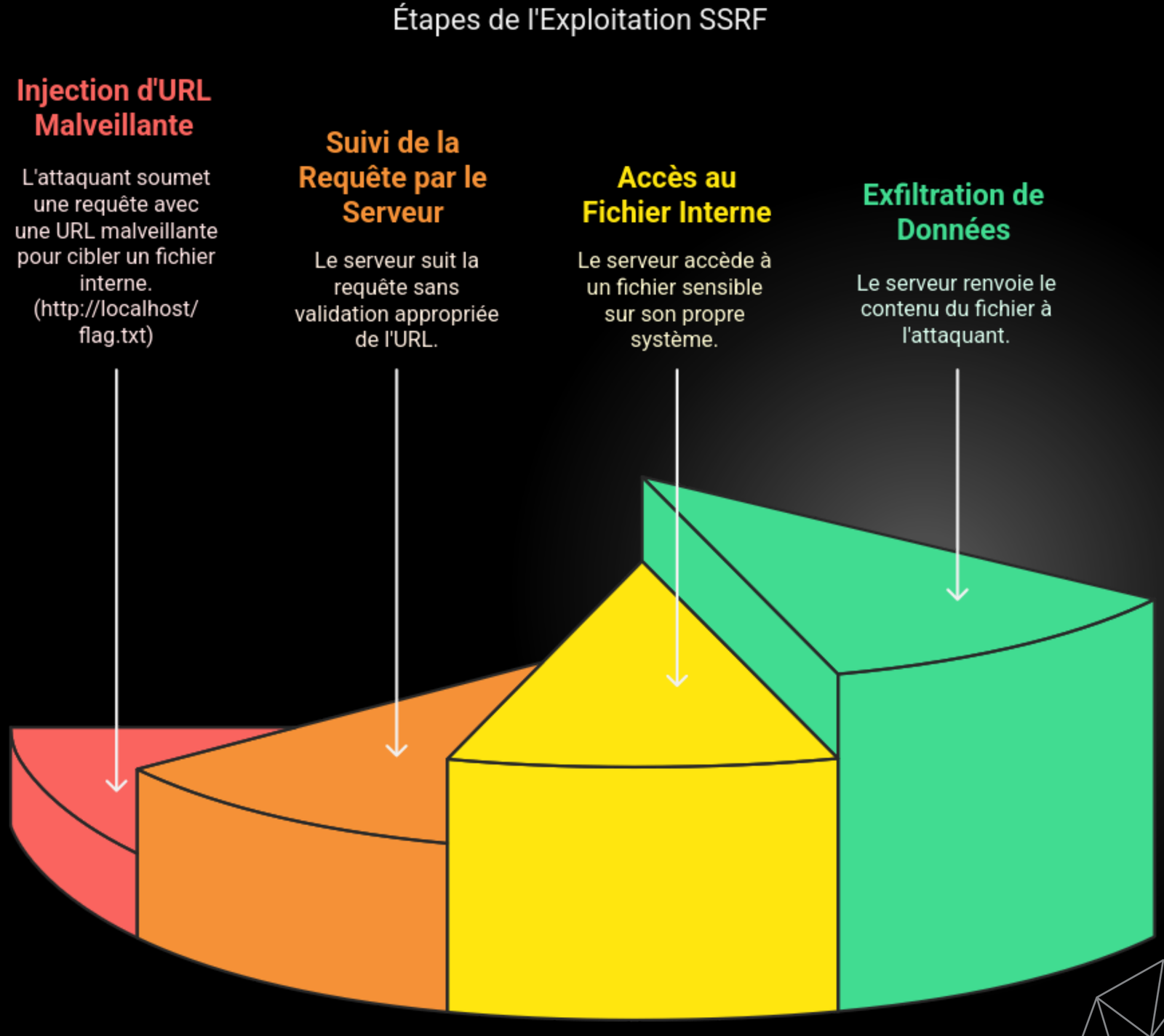
💥 **Attaque possible :**

- Lire des fichiers internes (file:// sur certains serveurs)
- Scanner le réseau interne (http://192.168.1.1/admin)

📚**Documents annexes:**

- https://youtu.be/2jtNkgIzX4E?si=5r2-3CAvx8XlGBMb
- https://portswigger.net/web-security/ssrf
- https://infosecwriteups.com/exploiting-server-side-request-forgery-ssrf-vulnerability-faeb7ddf5d0e

# Beyond SSRF - Exfiltration



# SSI - Server-Side Includes Injection

Exploite les inclusions dynamiques de fichiers dans des serveurs web configurés avec mod_include

🔹 **Cas concrets :**

✅ Serveurs Apache avec SSI activé

✅ CMS anciens utilisant l'inclusion dynamique

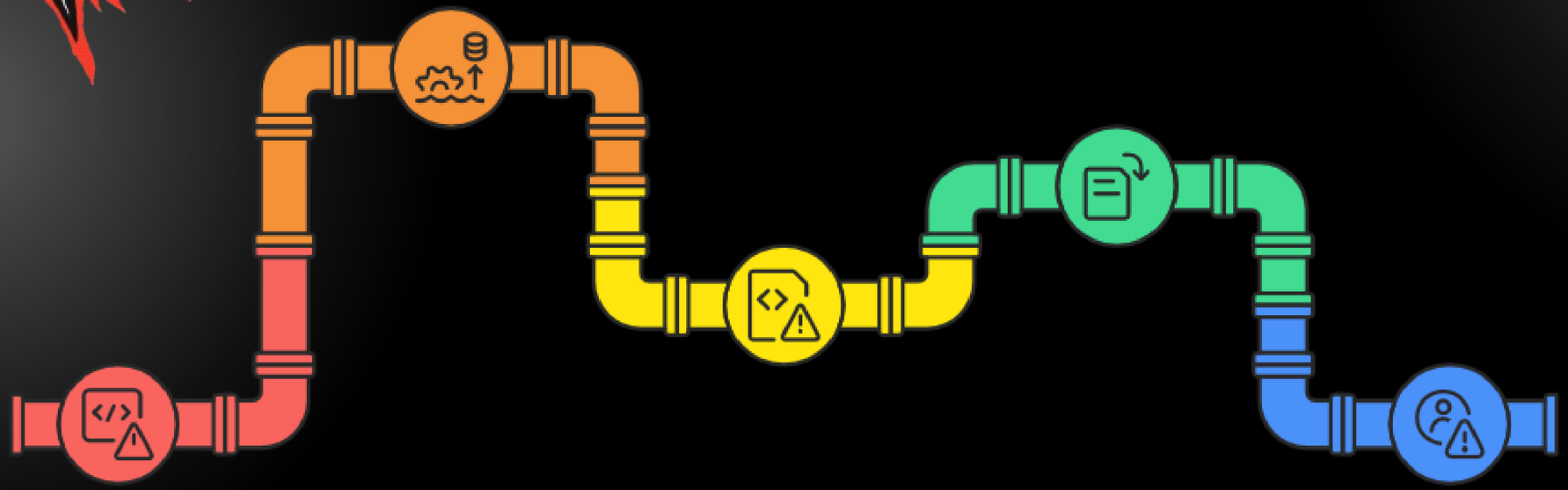
💥 **Attaque possible :**

- Exécuter une commande: (<!--#exec cmd="ls"-->)
- Lire des fichiers: (<!--#include file="/etc/passwd"-->)

📚**Documents annexes:**

- **h**ttps://httpd.apache.org/docs/current/howto/ssi.html
- https://portswigger.net/kb/issues/00101100_ssi-injection

Exploitation de l'inclusion côté serveur



# Beyond SSI - Exfiltration

01

**Injection de directive SSI**

L'attaquant envoie une requête HTTP avec une directive SSI malveillante.
ex:
<!--#include file="../../flag.txt"-->

02

**Traitement par le serveur**

Le serveur vulnérable traite la directive SSI et interprète la directive #include.

03

**Inclusion de fichier**

Le serveur inclut le fichier sensible spécifié.

04

**Réponse du serveur**

Le serveur renvoie le contenu du fichier dans la réponse HTTP.

05

**Exfiltration par l'attaquant**

L'attaquant récupère le contenu sensible de la réponse.

# XSLT Injection

Exploite la transformation de documents XML en HTML/PDF via XSLT

🔹 **Cas concrets :**

✅ Génération automatique de documents PDF

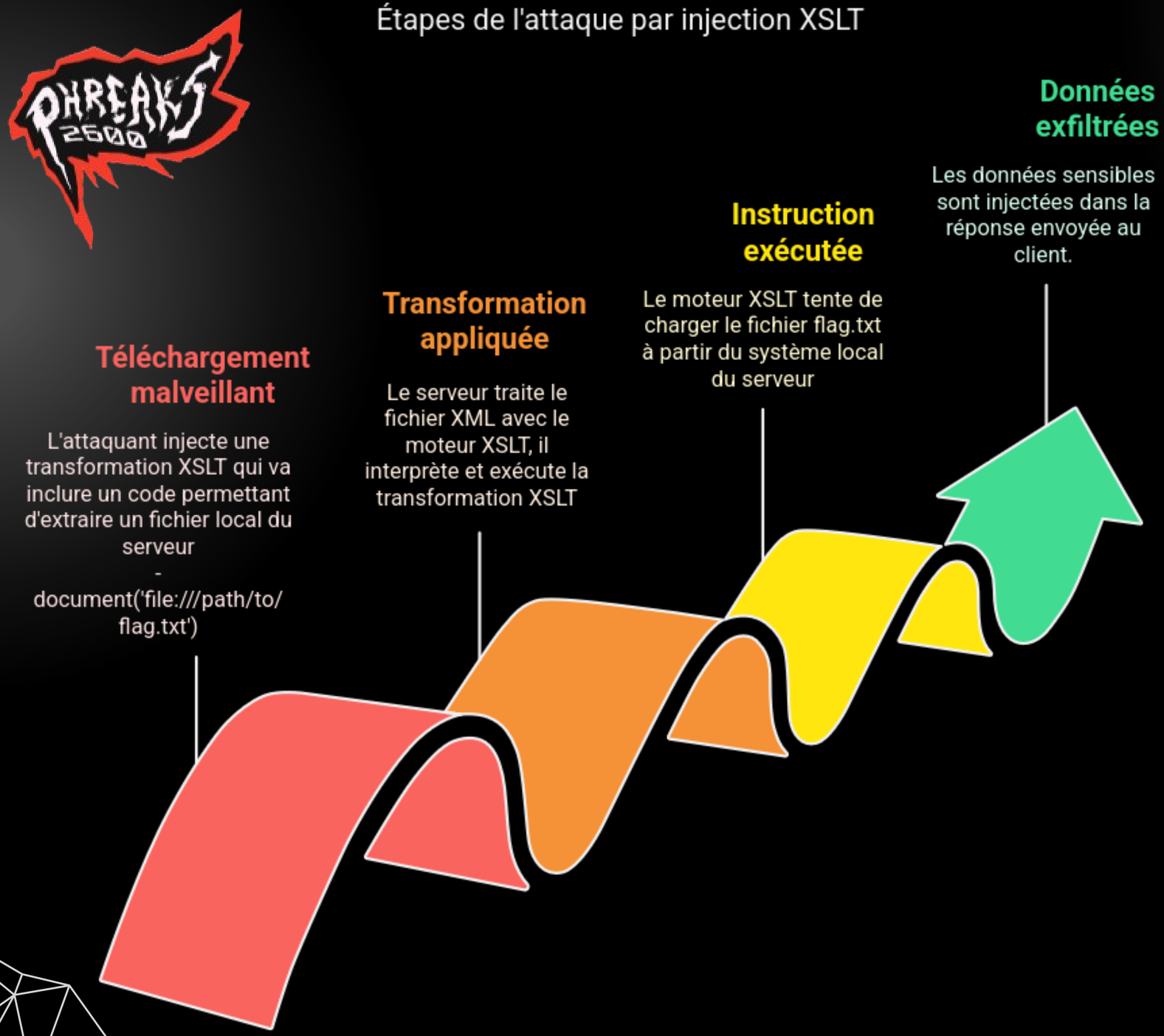
✅ API REST/XML utilisant XSLT pour transformer des données

💥 **Attaque possible :**

- Lire un fichier interne: (<xsl:value-of select="document('file:///etc/passwd')"/>)
- Exécuter du code malveillant (si extensions XSLT activées)

📚**Documents annexes:**

- https://blog.pentesteracademy.com/xslt-injections-for-dummies-a0cfbe0c42f5
- https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XSLT%20Injection/README.md
- https://felix-billieres.gitbook.io/v2/web-app/server-side-attacks/xslt




# Beyond XSLT - Exfiltration

# SSTI - Server-Side Template Injection

Exécution de code malveillant via des moteurs de templates (Jinja2, Twig, Freemarker, etc.)

🔹 **Cas concrets :**

✅ Applications utilisant des moteurs de templates pour générer des pages dynamiques

✅ Services de génération de rapports ou d'emails personnalisés

💥 **Attaque possible :**

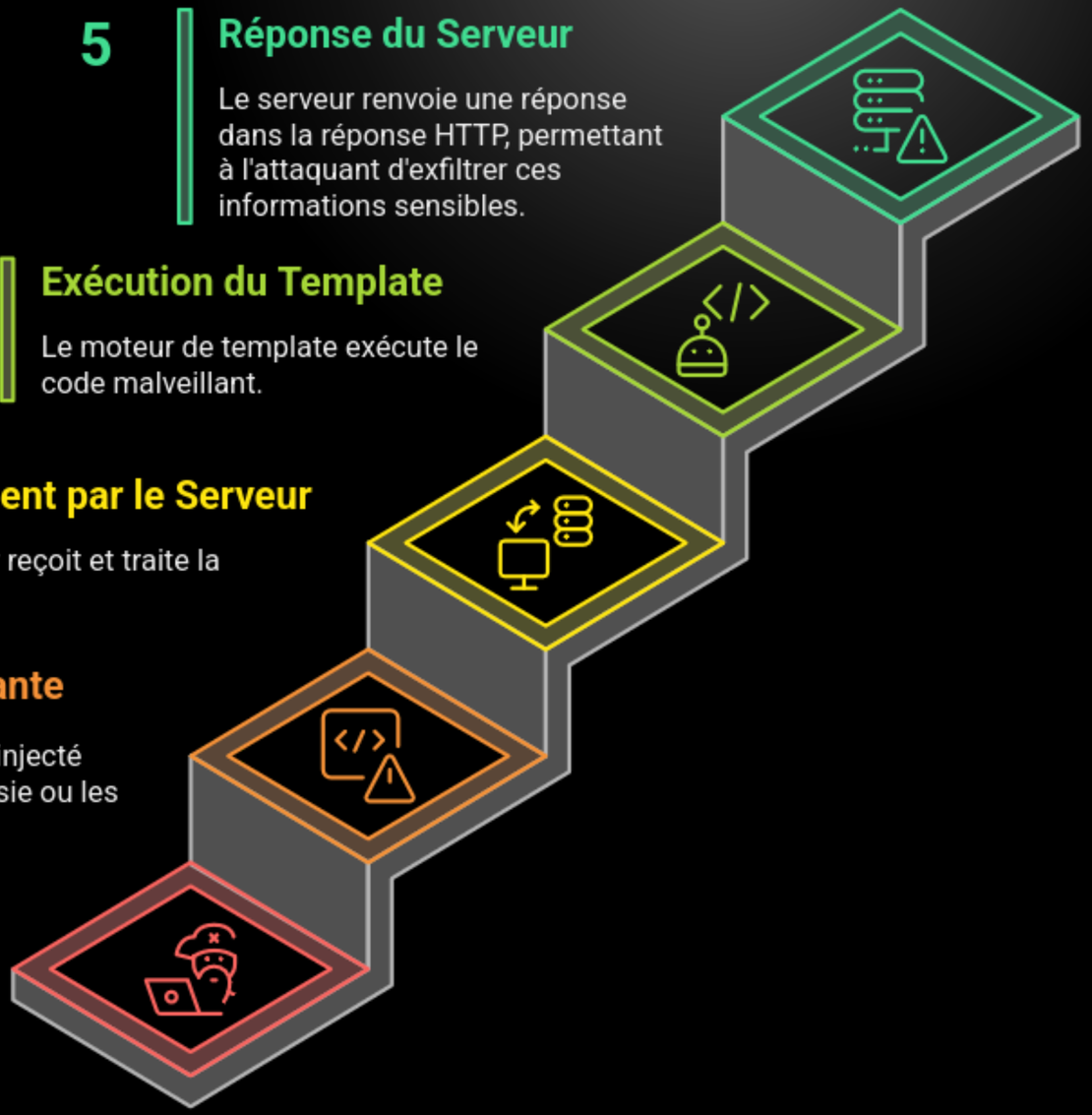
- Exécuter du code Python ({{7*7}} → 49)
- Lire des fichiers sensibles l'appel de variables globales
- RCE via les bibliothèques standard de Python

📚**Documents annexes:**

- https://portswigger.net/web-security/server-side-template-injection
- https://youtu.be/x_1A9rCxREs?si=CZ0Ua5pFoZcf3MWB
- https://youtu.be/FVm6wYc1S6A?si=ID8_mACqdH4Wq8Fg
- https://felix-billieres.gitbook.io/v2/web-app/server-side-attacks/ssti

# Beyond SSTI - Exfiltration

## Étapes d'une Attaque SSTI



### 1 Soumission de Données Malveillantes

L'utilisateur envoie une requête contenant un code de template malveillant.
ex:
{{ open('/flag.txt').read() }}

### 2 Injection Malveillante

Le code malveillant est injecté dans les champs de saisie ou les paramètres.

### 3 Traitement par le Serveur

Le serveur reçoit et traite la requête.

### 4 Exécution du Template

Le moteur de template exécute le code malveillant.

### 5 Réponse du Serveur

Le serveur renvoie une réponse dans la réponse HTTP, permettant à l'attaquant d'exfiltrer ces informations sensibles.