Intrusions web

ISI
2 - Intrusions

Résumé du document

TBD

Table des matières

1. Rappel technologie web

1.1. Appel HTTP

1.1.1. Appel HTTP

Un appel HTTP est une requête envoyée par un client à un serveur. Il est composé de deux parties : une ligne de requête et un corps de requête. La ligne de requête contient la méthode HTTP, l'URI et la version du protocole. Le corps de requête contient les données envoyées par le client.

1.1.1.1. Exemple d'appel HTTP

```
GET / HTTP/1.1
Host: www.lemonde.fr
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) …
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: _ga=GA1.2.374864853.1475228586; xtvrn=$43260$; _cb_ls=1...
If-Modified-Since: Fri, 30 Sep 2016 16:10:47 GMT
```

1.1.2. Réponse HTTP

Une réponse HTTP est envoyée par un serveur à un client. Elle est composée de deux parties : une ligne de statut et un corps de réponse. La ligne de statut contient la version du protocole, le code de statut et le message de statut. Le corps de réponse contient les données envoyées par le serveur.

1.1.2.1. Exemple de réponse HTTP

```
HTTP/1.1 200 OK
Date: Fri, 30 Sep 2016 16:10:47 GMT
Server: Apache
Last-Modified: Fri, 30 Sep 2016 16:10:47 GMT
ETag: "1d3-53e3a1f1f7d00"
Accept-Ranges: bytes
Content-Length: 467
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

1.1.3. En-têtes HTTP

Les en-têtes HTTP sont des informations supplémentaires envoyées avec une requête ou une réponse HTTP

1.1.3.1. Exemple d'en-têtes HTTP

1.1.3.1.1. Generaux

```
Cache-Control: max-age=3600, public
Date: Tue, 15 Nov 2005 08:12:31 GMT
```

1.1.3.1.2. Requêtes

```
Accept: text/plain;q=0.5, text/html
Accept-Charset: iso-8859-5, unicode-1-1;q=0.8
From: user@heig-vd.ch
Referer: http://www.iict.ch/index.html
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
```

```
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120; foo=bar
Authorization: Basic bXl1c2VyOm15cGFzcw==
```

1.1.3.1.3. Réponse

```
Location: http://www.heig-vd.ch
Server: Microsoft-IIS/6.0
Set-Cookie: session-id=120-7333518-8165026; path=/; domain=.amazon.com; expires=Sat Feb 27…
```

1.1.3.1.4. Contenu

```
Content-Encoding: gzip
Content-Length: 3495 (en octets)
Content-Type: text/html; charset=ISO-8859-4
Last-Modified: Tue, 15 Nov 2005 12:45:26 GMT
```

1.1.4. HTTP sans état

HTTP est un protocole sans état, ce qui signifie que le serveur ne conserve pas l'état de la session entre les requêtes. Cela signifie que chaque requête est traitée indépendamment des autres requêtes.

1.1.5. Cookies

Les cookies sont des fichiers stockés sur le client qui contiennent des informations sur l'utilisateur. Ils sont envoyés avec chaque requête HTTP et permettent au serveur de conserver l'état de la session entre les requêtes.

1.1.5.1. Exemple de cookie

- Set-Cookie (dans une réponse) : dépose un cookie sur le client.
- Cookie (dans une requête) : valeur du cookie déposé auparavant.

2. Références et outils

2.1. OWASP

- Open Web Application Security Project (OWASP)
- Organisation internationale, Open Source
- Participation gratuite et ouverte à tous
- Mission : promouvoir la sécurité des applications

2.2. Outils

- Plugins pour navigateurs (Tamper data, Firebug, etc.)
- Outils externes, interception et analyse des communications
  - Exemple : Proxys d'interception
- Logiciels de proxy avec beaucoup de fonctionnalités
  - Exemple : Burp suite

2.3. Outils automatisés

- Scanner/fuzzer/spider Web
- SAST = Static Analysis Security Tools
- DAST = Dynamic Analysis Security Tools

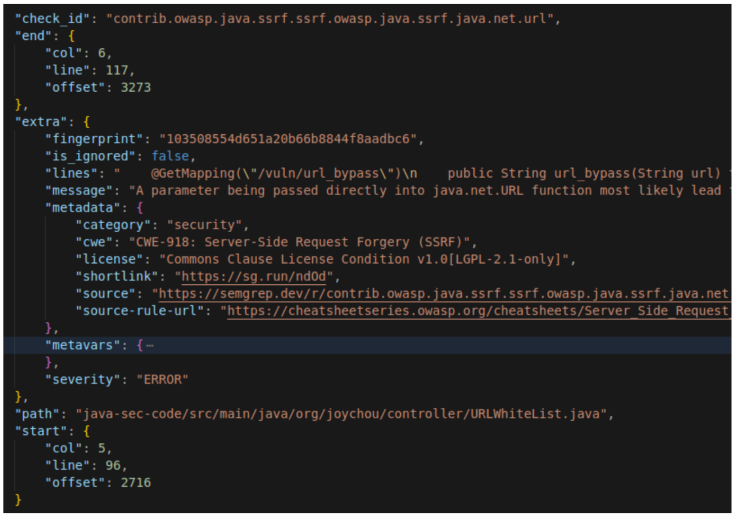
2.3.1. SAST

Outils libres et gratuits:

Semgrep
https://semgrep.dev/
Leader sur le marché
Ensemble de règles configurables
Résultats en format structuré (JSON)

Insider-CLI
https://github.com/insidersec/insider
Focalisé sur le top 10 OWASP
Résultats structurés (JSON)

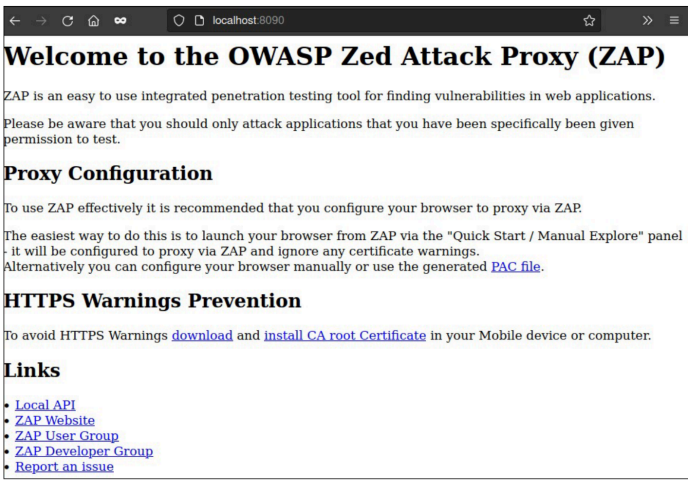
GRAudit
https://github.com/wireghoul/graudit/



2.3.2. DAST

Outils libres et gratuits:

ZAProxy
https://github.com/zaproxy/zaproxy
Leader du marché / Owasp
API pour connexions à distance
Ensemble de règles très complet et configurable
GUI Web pour tests manuels



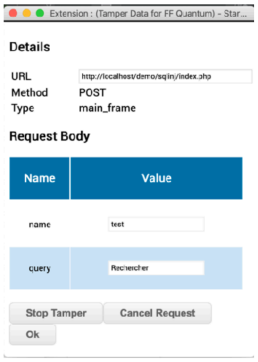
W3af
https://github.com/andresriancho/w3af
API et beaucoup de plugins disponibles
Plus vraiment maintenu?

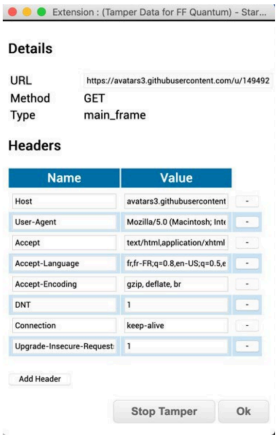
Nikto
https://github.com/sullo/nikto
Bon mais pas très facile à personnaliser

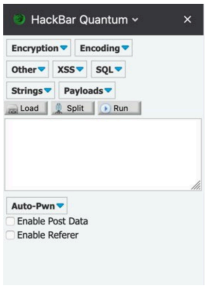
2.4. Proxys d'interception

- Règles / Interception
- Historique ou Spider
- Match-and-replace
- Fuzzer
- Scanner de vulnérabilités
- Requêtes manuelles
- Analyse des cookies/sessions/tokens

- Inclus dans le navigateur **Firefox**
  - Inspection/modifications (locales) : code HTML code, en-têtes HTTP, cookies, requêtes, etc. (ouvrir « ouRl -> développement Web »)
- **Tamper data** (pour Firefox)
  - inspection/modifications (messages): en-têtes des requêtes/réponses, HTTP/HTTPS, contenu POST, etc.
- **HackBar Quantum**
  - Utilitaire d'encodage (base64, Hex, URL), hachage, construction de requêtes (XSS, SQL)
- FoxyProxy Standard
  - Gestion des proxys (changement rapide, règles par URL)
- Cookie Manager
  - Examiner, modifier, effacer, importer, exporter les cookies
- User-agent switcher
  - Mentir au serveur sur le logiciel client utilisé
- HTTP Header Live
  - Examiner, modifier, rejouer des requêtes en live







3. Attaques WEB

- Manipulation des données (URL, cookies, etc.)
- Contournement de protections côté client
- «Session Hijacking»
- «Cross-site scripting» (XSS)
- «Cross-site request forgeries» (CSRF)
- Injections de commandes (SQL ou autres)

3.1. Objectifs d'une attaque

- Contourner un mécanisme de sécurité (authenticité ou ...)
- Extraire des données (confidentialité)
- Ajouter ou modifier des données (integrité)
- Déterminer le schéma de la base de données

3.2. Points d'injection

3.2.1. Premier ordre

- L'application traite incorrectement une donnée malveillante directement
- Exemples:
  - Entrées utilisateur / formulaires
  - Cookies
  - URLs

3.2.2. Second ordre

- L'application traite incorrectement une donnée enregistrée précédemment
- Exemples:
  - Contenu d'une base de données
  - Fichiers uploadés
  - Variables côté serveur

3.3. Protection côté client

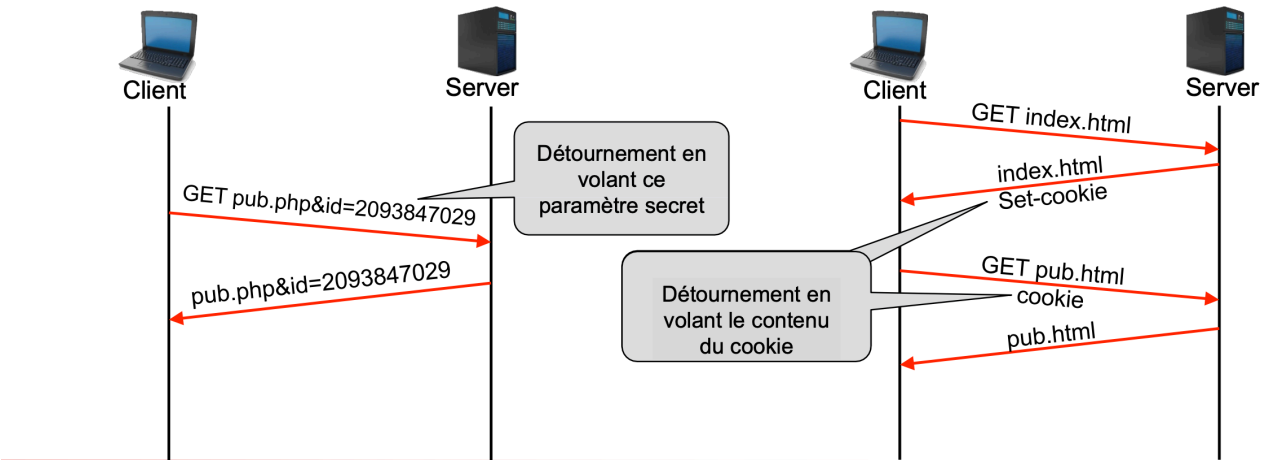
Toute protection côté client peut être contournée. C'est pourquoi il est important de mettre en place des protections côté serveur.

- Contenu des cookies
- En-tête HTTP (Referer, User-agent, etc.)
- Champs des formulaires (même si cachés, restreint au choix dans une liste, ou validés en javascript)
- Tout contrôle en javascript (fait côté client)

Une application web doit toujours répéter chaque validation côté serveur.

3.4. Détournement de session

- Un attaquant récupère un identifiant de session valide pour contourner le mécanisme d'authentification.
- La technique varie en fonction du protocole (TCP, HTTP)
  - Courant pour HTTP: vol de cookie, vol d'URL, falsification d'URL, etc.



3.5. Cross-site scripting (XSS)

3.5.1. Reflected XSS (sans persistance)

- Le client contrôle une valeur utilisée telle quelle dans la réponse
- Exemple : moteurs de recherche personnalisés.

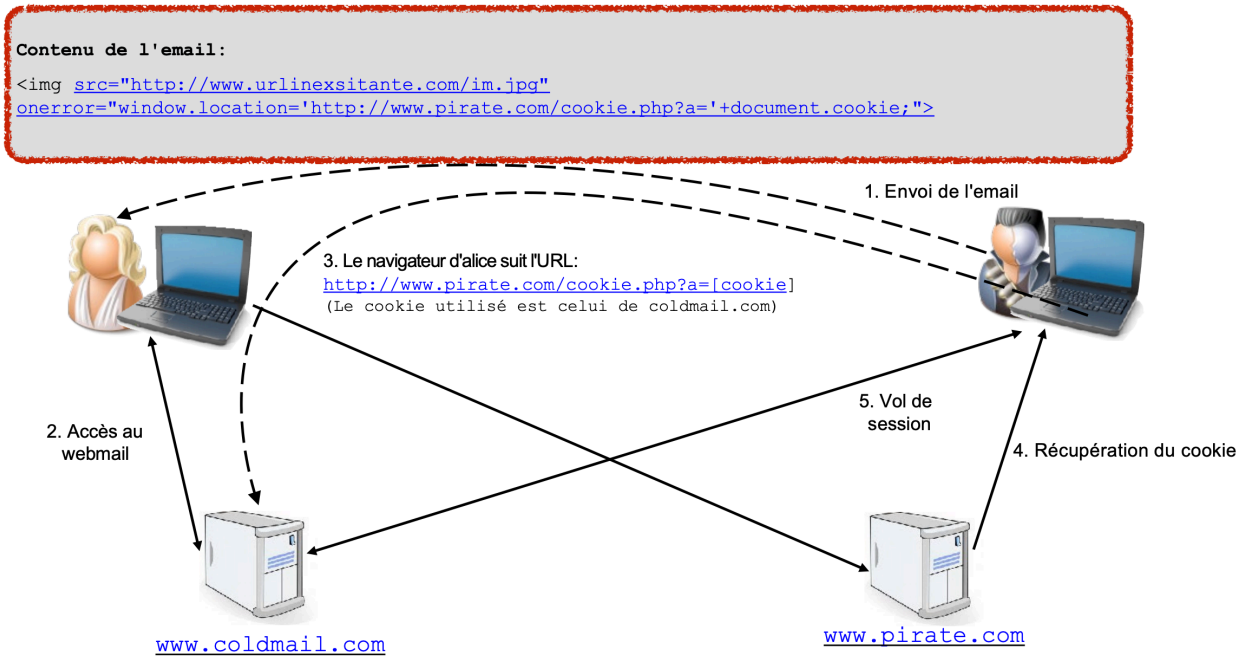
3.5.2. Stored XSS (avec persistance)

- Le client contrôle une valeur enregistrée côté serveur.
- La valeur est utilisée ultérieurement dans plusieurs autres réponses.
- Exemples : forums, livre d'or, ...

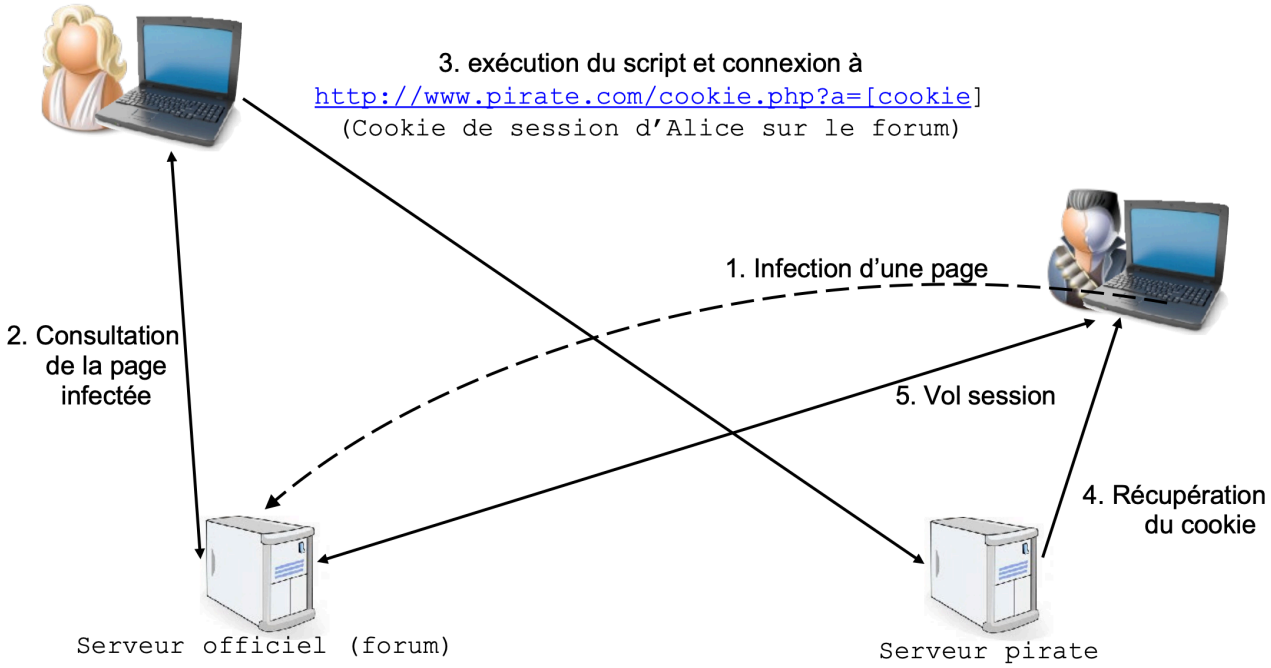
3.5.3. DOM-based XSS

- Modification du « Document Object Model » (DOM)
- La payload n'est pas dans la réponse mais dans l'URL, et exploite une faille dans le code côté client.

3.5.4. Vol de cookies (XSS par mail)



3.5.5. Vol de cookies (XSS sur un forum)



3.5.6. DOM-based XSS

```
<html>
<head>
<title>Custom Dashboard </title>
...
</head>
Main Dashboard for
<script>
        var pos=document.URL.indexOf("context=")+8;
        document.write(document.URL.substring(pos,document.URL.length));
</script>
...
</html>
```

La page est modifiée côté client par un script, qui y insère dynamiquement un morceau de l'URL

http://www.example.com/userdashboard.html?context=Mary → Générer le dashboard de Mary

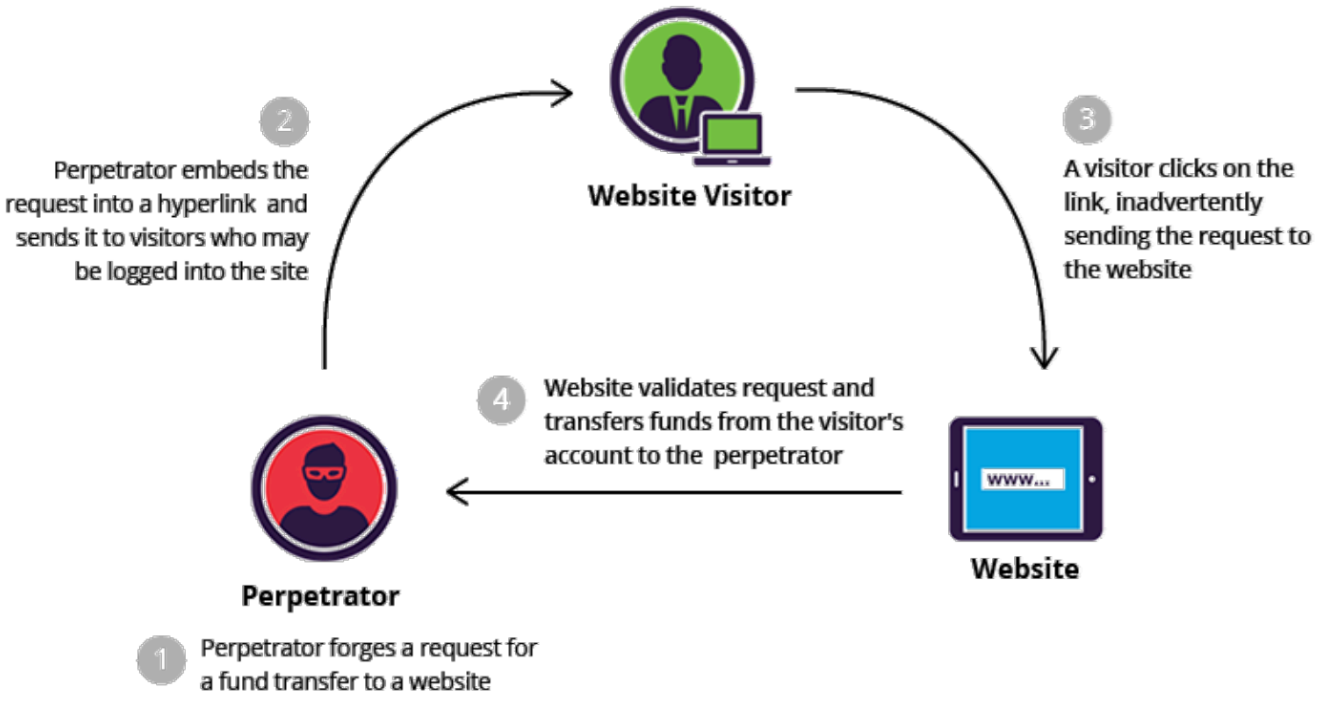
http://www.example.com/userdashboard.html#context=<script>SomeFunction(somevariable)</script>

→ Générer le dashboard de <script>SomeFunction(somevariable)</script>

https://www.acunetix.com/blog/articles/dom-xss-explained/

3.6. CSRF (Cross-site request forgery)

- Objectif = forcer la victime à exécuter une action malveillante sur une application web
- La victime ouvre une URL malveillante
- L'attaque déclenche une requête vers le site ciblé



3.7. Injection de commandes

3.7.1. Injections dans des commandes SQL

- Une donnée utilisateur sert à construire une commande SQL pour la base de données.
  - Formulaire, cookie, paramètre de l'URL, etc.
- Sans protection, il est possible de modifier le sens de la commande SQL!
- Et donc de manipuler la base de données !
  - Vol de la base de données (tout ou partie)
  - Destruction de la base de données (tout ou partie)
  - Modification/altération des données

3.7.2. Injection dans une commande système

- Même idée, mais l'application construit une commande système (shell)
  - e.g. php include (https://brightsec.com/blog/code-injection-php/)
- Il devient possible d'exécuter des commandes autres que celles prévues par le développeur

Authentification:
Name
Password

```
Request = SELECT nom, pwd FROM user_db WHERE nom ='Name' AND pwd ='Password'

If (Request != empty)
  Authentification successful
       Else
  Authentification failed
```

Authentification:
Toto
' OR '1'='1

```
Request = SELECT nom, pwd FROM user_db WHERE nom ='Toto' AND pwd ='' OR '1'='1'

If (Request != empty) then
  Authentification successful
       Else
  Authentification failed
```

Toujours vrai!