La **sécurité** *pour les développeurs*

Christophe Villeneuve
@hellosct1

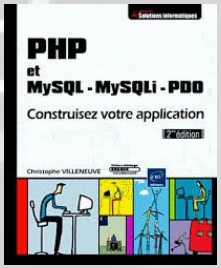
BY NC SA

16es Rencontres Mondiales du Logiciel Libre

Du 4 au 10 juillet 2015

RMLL BEAUVAIS 2015

Qui... est Christophe Villeneuve ?



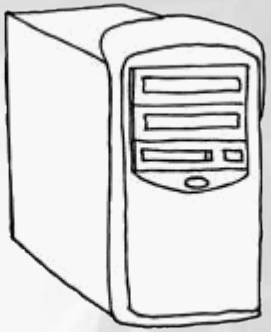
afup – lemug.fr – mysql – mariadb – drupal – democscene – firefoxos – drupagora – phptour – forumphp – solutionlinux – demoinparis – Libre à toi – eyrolles – editions eni – programmez – linux pratique – webriver – phptv – neuros - elephpant

OWASP 2013

- Failles d'injection
- Violation d'authentification et de Session
- Cross-Site Scripting (XSS)
- Référence directe non sécurisée à un objet
- Mauvaise configuration de sécurité
- Données sensibles accessible
- Manque de sécurité au niveau des rôles
- Falsification de requête (CSRF)
- Utilisation de composants connus vulnérables
- Redirections non validées

SOLUTION

Rapport
OWASP



A

M

P

Sommaire

- A => Apache => Architecture / Serveur
- M => MySQL / MariaDB => Base de données
- P => PHP => Langage / webService / API

- La sécurité dans les outils et API Métiers

A

Architecture / Serveur

- A1 - Injection en ligne de commande
- A5 - Mauvaise configuration sécurité
- A6 - Exposition de données sensibles
- NC - Exécution fichiers malicieux

Injection de ligne de commandes

Principe de l'attaque

- Configuration non à jour
- Pas de maintenance
- Mise à disposition des fonctions
  - Exec
  - System

Conséquence

- Autre manière de prise en main du système
- Serveur Zombie

Désactiver dans php.ini (pour PHP)

- exec
- passthru
- shell_exec
- system
- proc_open
- popen
- curl_exec
- curl_multi_exec
- parse_ini_file
- show_source

Affiche le nom de l'utilisateur

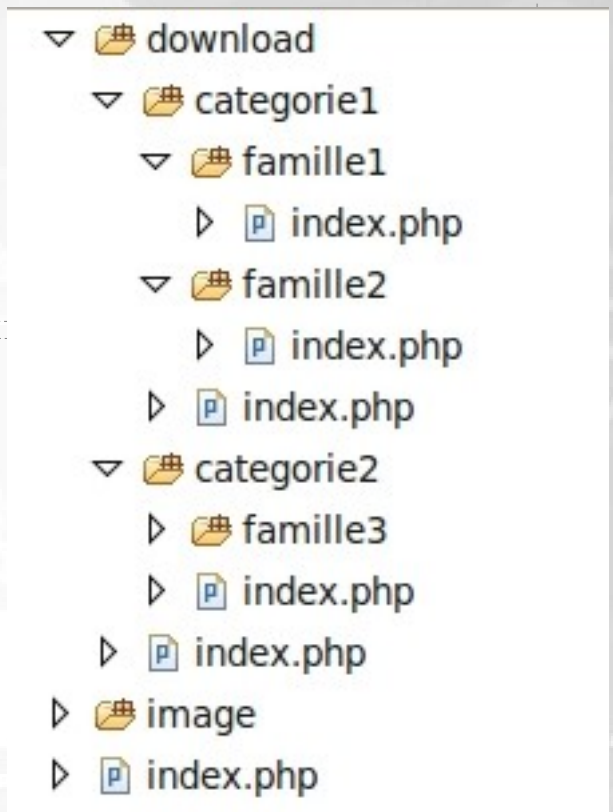
```
<?php
	echo exec('whoami');
?>
```

Par conséquent :

safe_mode = Off
allow_url_fopen=Off
allow_url_include=Off

Cacher le contenu des dossiers

- Ex : http://votreURL.com/nomDossier



- Solution : fichier index.php

```
<?php
header("Location: ../index.php");
die() ;
?>
```

Exécution fichier malicieux

Principe de l'attaque	Conséquence
✔ Exécuter un fichier (ex : PHPShell) ✔ Accès en écriture ✔ FTP, SSH, HTTP PUT, WebDav... ✔ Absence de contrôle du dépôt des fichiers ✔ Fonction include()	• Prise en main du système • Accéder à des informations non autorisées

Pour se protéger des fichiers malicieux

- <?php Include ($file) ; ?>
  - Ex : http://votreURL.com/file=toto.php
- Solution
  - http://votreURL.com/toto.php
- Ecraser le contenu de la variable
  - http://urlPirate.com/hack.gif
- Solution dans php.ini
  - allow_url_fopen = off

php_self non bloqué

Uname: Linux sd-57667 3.2.0-4-amd64 #1 SMP Debian 3.2.54-2 x86_64 [exploit-db.com]
User: ( ) **Group:** 999 (dtcgrp)
Php: 5.4.4-14+deb7u8 **Safe mode:** OFF [phpinfo] **Datetime:** 2014-06-04 05:12:47
Hdd: 1 B **Free:** 0 B (0%)
Cwd: /data/sites/dtc/d4hoster.com/subdomains/office/html/ drwxr-xr-x [home]

[Sec. Info] [Files] [Console] [Sql] [Php] [String tools] [

Server security information

Server software: Apache/2.x.x
Loaded Apache modules: core, mod_log_config, mod_logio, mod_version, prefork, http_core, mod_so, mod_action
mod_authz_default, mod_authz_groupfile, mod_authz_host, mod_authz_user, mod_autoindex, mod_cgi, mod_de
mod_log_sql_ssl, mod_mime, mod_security2, mod_negotiation, mod_reqtimeout, mod_rewrite, mod_setenvif, mod
Disabled PHP Functions: popen,pcntl_alarm,pcntl_fork ,pcntl_waitpid ,pcntl_wait ,pcntl_wifexited
,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_dis
cURL support: enabled
Supported databases: MySql (5.5.36-MariaDB)

Readable /etc/passwd: yes [view]
Readable /etc/shadow: no
OS version: Linux version 3.2.0-4-amd64 (debian-kernel@lists.debian.org) (gcc version 4.6.3 (Debian 4.6.3-14)) #
Distr name: Debian GNU/Linux 7

Useful: gcc, cc, ld, make, php, perl, python, tar, gzip, bzip2, nc, locate
Danger: clamd, iptables
Downloaders: wget, curl, lwp-mirror

HDD space:

Filesystem	Size	Used	Avail	Use%	Mounted on
rootfs	2.7T	695G	1.9T	27%	/
udev	10M	0	10M	0%	/dev
tmpfs	6.4G	268K	6.4G	1%	/run
/dev/disk/by-uuid/7691d853-2646-4458-beeb-191dab36292f	2.7T	695G	1.9T	27%	/
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	13G	0	13G	0%	/run/shm

© Source : D4.FR

Protection de base

- Php.ini
  - safe_mode = off
- Verrouillez les dossiers
  - .htaccess
  - Chmod (444) ou 665 ou 775
- HTTPs / SSL

M

Base de données

- A1 - Injection SQL
- A4 - Référence direct non sécurisés à un objet

Injection SQL

Principe de l'attaque

- Envoie du code SQL
  - Formulaire
  - GET / POST
  - Cookies
  - ...

Conséquence

- Contournement authentification
- Récupération des données de la base
- Récupération de fichiers
- Exécution de codes

Utilisation du SQL

- Risque : Requête avec des simples quotes

  SELECT * FROM 'users' WHERE 'username'='$login' AND 'password'='$pass'

- Saisie : $login = hello $pass = hello

  TRUE

  SELECT * FROM 'users' WHERE 'username'='hello' AND 'password'='hello'

- Saisie : $login = ' OR '1'='1' $pass = ' OR '1'='1'

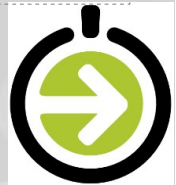
  TRUE

  SELECT * FROM 'users' WHERE 'username'='' OR '1'='1'' AND 'password'='' OR '1'='1''

- Saisie : $login = ' OR 1=1"); drop table users; $pass =

  TRUE
  Sauf si BDD lecture

  SELECT * FROM 'users' WHERE 'username'='' OR 1=1"); drop table users;' AND 'password'=''



Se protéger contre injection SQL

- addslashes()
  - *Ajoute des antislashs dans une chaîne*

*SELECT * FROM 'users'*

WHERE 'username'=' \' OR \'1\'=\'1\' '

AND 'password'=' \' OR \'1\'=\'1' '

les guillemets simples '
les guillemets doubles "
les slashes /
les caractères NULL

mysqli_real_escape_string()

  - *Protège les caractères spéciaux*

- *pdo_quote()*
  - *Place des guillemets simples autour d'une chaîne entrée*

Modifier les arguments dynamiquement

Principe de l'attaque	Conséquence
✔ Absence de contrôle dans une requête ✔ Récupération d'une valeur visible + modification ✔ Trouver d'autres données et informations ✔	✔ Usurpation d'identifiant ✔ Effectué des opérations non prévues initialement

Comportement des objets

- Exemple

  http://votreURL.com/compte?id=IdFalse

- Mauvaise utilisation

```
<?php
$sql = "SELECT * FROM users WHERE id = :id " ;
$qid = $cnx->prepare($sql);
$qid->execute (array (':id', $id) ) ;
$rows = $qid->fecthAll(PDO::FETCH_ASSOC) ;
?>
```

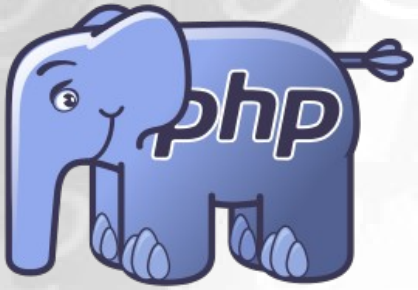
- Une solution

```
$qid->bindParam(':id', $id, PDO::PARAM_INT);
```

OU

```
$qid->bindParam(':id', $id, PDO::PARAM_STR, 12);
$qid->execute () ;
```

P



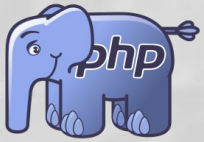
Langage

- A1 – Injection API
- A2 -Violation de gestion d'authentification et de session
- A3 - Cross Site Scripting (XSS)
- A6 - Exposition de données sensibles
- A7 - Manque de contrôle d'accès au niveau fonctionnel
- A8 - Falsification de requête intersites (CSRF)
- A9 - Utilisation de composants vulnérable
- A10 - Redirections et renvois non validés

Provenance coté front

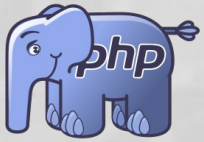
- Navigation
- Formulaire
  - Champs : Input, upload,...
- Des API Couche Métier

Gestion des données entrées



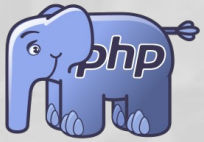
Principe de l'attaque	Conséquence
✔ Suivi des utilisateurs par SESSION ID ✔ Caractéristiques utilisateur stockées coté serveur par une variable de session ✔ Gestion des états : Cookies / Get / Post ✔ Chiffrement faible	✔ Vol des données SESSION_ID si elles ne sont pas cryptées ✔ Utilisation ailleurs

Solution de contrôle (1/3)



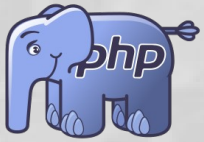
- Prévoir la présence d'une clef de hashage caché
  - Générer une clef cryptée de hachage
    - IP
    - Navigateur utilisé
    - Une durée de validité
    - ...
  - Différencier les formulaires
  - Eviter la protection en MD5 pour HASH

Solution de contrôle (2/3)



- Remède contre Session ID
  - Cryptage par HASH
  - Eviter le MD5 avec la date de connexion
  - Contenu aléatoire
- Oublier les champs Hidden avec des caractéristiques utilisateur

Solution de contrôle (3/3)



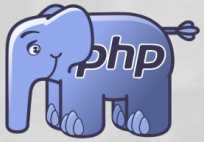
- Lors de l'envoie d'un formulaire, quelques bases
  - If isset($_POST['string']) { /* … */ }
  - If sizeof ($_POST['string'])>0 { /* ... */ }

- Attention aux superglobales

  $GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_SESSION, $_REQUEST, $_ENV

  - $str=htmlentities ($_COOKIE['string'],ENT_QUOTES) ;

Naviguer dans les pages webs

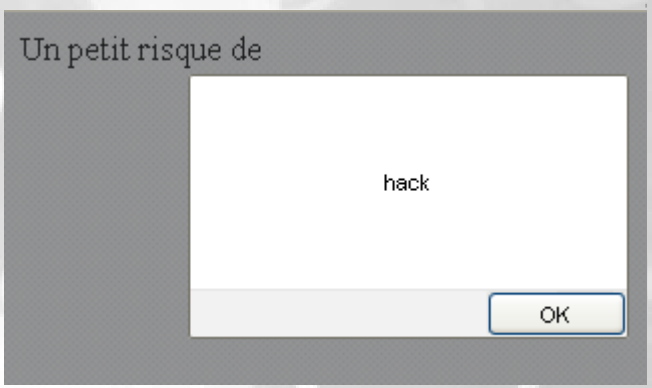


Principe de l'attaque	Conséquence
✔ XSS	✔ Vol de sessions (cookies)
✔ Risque applicatif	✔ Redirection de pages
✔ Langage de requête	✔ Scanner des pages
✔ Ldap / Xpath	✔ Usurpation d'identitée / Phishing
✔ Langage interprété du type	✔ Contrôle du navigateur
✔ eval, system, consorts	

La navigation en mode tranquille

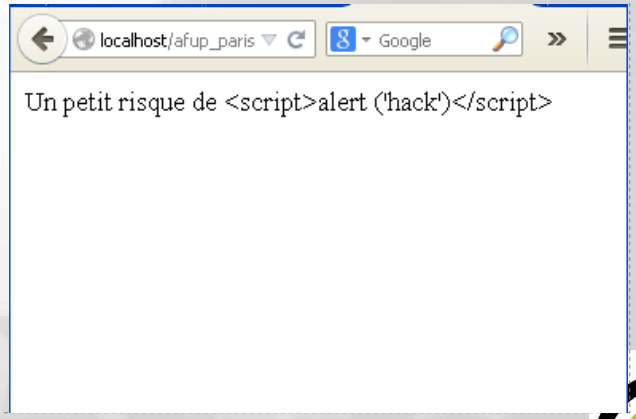
- Absence de protection

```
<?php
echo "Un petit risque de
<script>alert ('hack')</script>";
?>
```

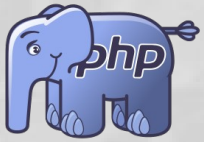


- Avec une protection

```
<?php
echo htmlentities("Un petit risque de
<script>alert ('hack')</script>");
?>
```



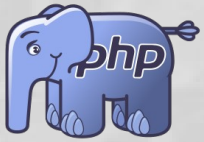
Contre mesures



- Données entrantes :
  - Valider chaque valeur (longueur, type...)

    filter_var() ou filter_input()
  - strip_tags // supprime caractère NULL
  - Htmlentities // balise html
- Données en sortie :
  - Htmlspecialchars() ;

Se protéger de PHP_SELF



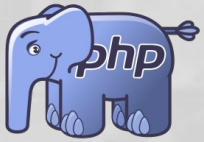
Exemple :

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>">
<input type="text" name="exemple" value="reussi" />
<input type="submit" value="Submit" />
</form>
```

- Dans un formulaire ou construction d'une URL

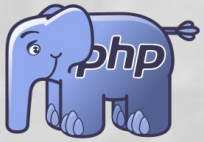
```
<?php echo htmlentities($_SERVER['PHP_SELF']); ?>
```

Contrôle d'accès au niveau fonctionnel



Principe de l'attaque	Conséquence
✔ Accéder à des pages non autorisés ✔ Modifier les droits	✔ Prise de contrôle du site ✔ Générer des actions non autorisés

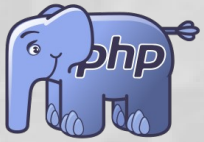
Solution contrôle d'accès



- Exemple
  - http://urlSite.com/getpage
  - http://urlSite.com/admin_getpage

- Solutions
  - Vérifier le contrôle d'accès (principe identification)
  - Vérifier les URLs

Exposition de données sensibles



Principe de l'attaque

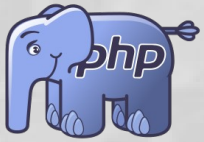
- Trouver
  - des données stockés / archivés en clair
  - Espace privée non partagée
  - Communication avec la banque
- Déterminer les algorithmes de cryptage faible

Conséquence

- Cible principale
  - Mot de passes
  - Données sensibles non chiffrées
  - Carte bleu

OWASP : A1 / A6 / A10

Filtrer les données sensibles



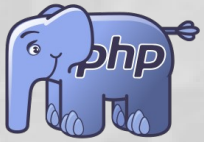
- Utilisation du port https

```
if($_SERVER['SERVER_PORT'] != 443)
{
header('Location: https://urlDuSite.com/prive.php');
exit;
}
```

- Vérifier le certificat

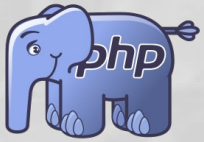
```
<?php
    if (!isset($_SERVER['SSL_CLIENT_M_SERIAL'])
        || !isset($_SERVER['SSL_CLIENT_V_END'])
        || !isset($_SERVER['SSL_CLIENT_VERIFY'])
        || $_SERVER['SSL_CLIENT_VERIFY'] !== 'SUCCESS'
        || !isset($_SERVER['SSL_CLIENT_I_DN'])
    ) { //FALSE;
    }

    if ($_SERVER['SSL_CLIENT_V_REMAIN'] <= 0) {
        // FALSE;
    }

    // TRUE;
?>
```

Modifier le comportement



Principe de l'attaque	Conséquence
✔ Falsification de requêtes (CSRF) ✔ Modification du contenu d'une page ✔ But éviter de passer par le formulaire	✔ Conduire l'utilisateur vers un site malveillant ✔ Lui forcer la main ✔ Ex : download

Imposer un comportement



- Créer un token ou un jeton de sécurité (toutes les pages)

Passage 1

```
<?php
session_start();
$token = uniqid(rand(), true);        // jeton unique
$_SESSION['token'] = $token;          // stockage

// heure de création du jeton
$_SESSION['token_time'] = time();
?>
<html><body>
<form id="form" name="form" method="post"
action="traitement.php">
 ...
   <input type="hidden" name="token"
id="token" value="<?php echo $token;?>"/>
 ...
</form>
</body></html>
```

Passage 2

```
<?php
session_start();
if(isset($_SESSION['token'])
   && isset($_SESSION['token_time'])
   && isset($_POST['token']))
{
   //Si jeton session = au formulaire
   if($_SESSION['token'] ==
                          $_POST['token'])
   {
      // exécution du code
   }
}
// sinon erreur
?>
```

Impact dans
les Outils et les API Métier

Outils applicatifs

Principe de l'attaque	Conséquence
✔ Librairy ✔ ORM ✔ API • Framework • CMS • CRM/ERP • …	✔ Voir le début de la présentation

API Métier : connexion LDAP

Principe de l'attaque	Conséquence
✔ Envoie du code SQL • Formulaire • GET / POST • Cookies • ...	✔ Contournement authentification ✔ Récupération des données de la base ✔ Récupération de fichiers ✔ Exécution de codes

Comportement injection LDAP

- Formulaire exemple

```
<input type="text" size=20 name="username">
```

- Connexion Ldap

```
String ldapSearchQuery = "(cn=" + $username + ")";
System.out.println(ldapSearchQuery);
```

- Solution
  - Valider les données avant de générer une requête de recherche

Webservices

Code Mini
PHP 5.4.x

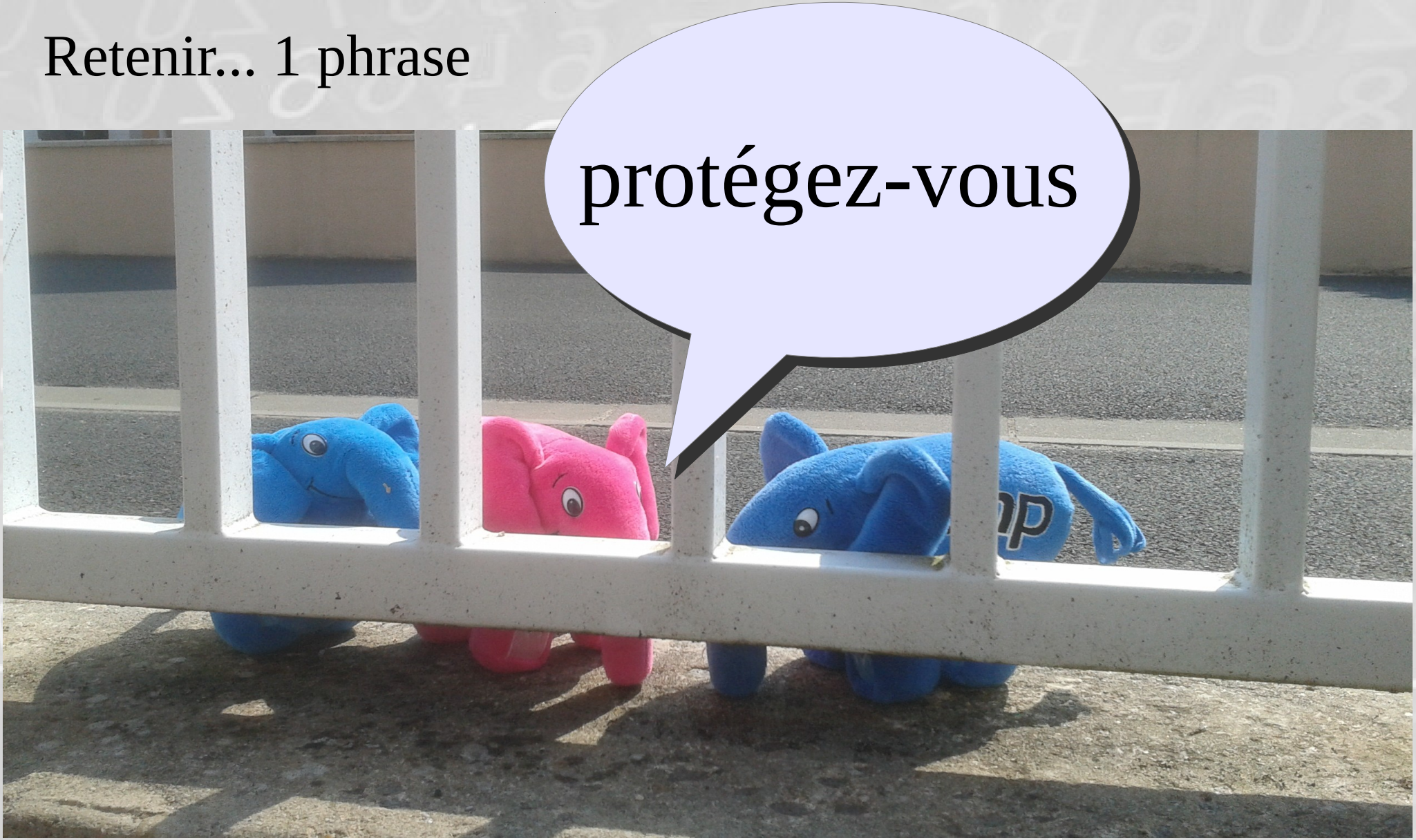
```
function encrypt_decrypt($action, $string) {
    $output = false;

    $encrypt_method = "AES-256-CBC";
    $secret_key = 'votre clef secrete';
    $secret_iv = 'vecteur aleatoire secret';

    $key = hash('sha256', $secret_key);          // hash

        // prepare une methode de cryptage sur 16 caractères
    $iv = substr(hash('sha256', $secret_iv), 0, 16);

    if( $action == 'encrypt' ) {
        $output = openssl_encrypt($string, $encrypt_method, $key, 0, $iv);
        $output = base64_encode($output);
    }
    else if( $action == 'decrypt' ){
        $output = openssl_decrypt(base64_decode($string), $encrypt_method, $key, 0, $iv);
    }

    return $output;
}
```

Resume
Objective

Les impacts pour un développement

Classement OWASP 2013	Serveur	BDD	Langage
A1-Failles d'injection	X	X	X
A2-Violation d'authentification et de Session			X
A3-Cross-Site Scripting (XSS)			X
A4-Référence directe non sécurisée à un objet		X	
A5-Mauvaise configuration de sécurité	X		
A6-Données sensibles accessible	X		X
A7-Manque de sécurité au niveau des rôles			X
A8-Falsification de requête (CSRF)			X
A9-Utilisation de composants connus vulnérables			X
A10-Redirections non validées			X
NC- Exécution fichiers malicieux	X		

Retenir... 1 phrase



www.OWASP.org