Examen - Sécurité des Applications

Thibaut HENIN Corinne HENIN INSA Centre Val de Loire

2019-2020

Contexte et mission

La société *Speed-e-Devs* est un éditeur de logiciel web et compte pénétrer le marché des CMS et autres moteurs de blogs.

L'année dernière, elle a fait appel à des ingénieurs spécialistes de la sécurité des applications pour auditer le code de son application. Des vulnérabilités ont été découvertes et ont fait l'objet de recommandations. Suite à ces dernières, l'équipe de *Recherche & Développement* a produit une nouvelle version du moteur de blog.

Avant sa mise sur le marché, *Speed-e-Devs* vous confie ce nouvel audit du code source de son application. Il vous est donc demandé de lire le code source fourni à la recherche des vulnérabilités présentes. Pour chaque vulnérabilité trouvée, vous devrez préciser les éléments suivants :

- Les fichiers et ligne de code concernées ;
- Des indications pour exploiter la vulnérabilité ;
- Les conséquences d'une exploitation ;
- Des indications sur la manière de la corriger.

Structure de l'application

L'application est développée principalement en `PHP` pour la gestion des requêtes web, en `SQL` pour la persistance des données et en `C` pour l'implémentation des primitives cryptographiques.

L'ensemble du projet doit être déployé dans le répertoire `/var/www/`. Les fichiers sont organisés en sous-répertoires comme suit :

- `classes` contient le code `PHP` orienté objet, réparti en quatre *namespaces* :
  - `Dal` (Database Access Layer) pour les accès à la base de donnée,
  - `Model` pour le code *métier* et la description des objets,
  - `Utils` pour les classes fournissant des services divers,
  - `View` pour la construction des pages web.
- `html` contient le code php disponible au travers du Web, c'est ici que pointe de *DocumentRoot* de l'application. Deux répertoires (`posts` et `users`) permettent d'organiser les services disponibles aux utilisateurs.
- `mysql` contient les scripts de création de la base de donnée.

Le fichier `vernam.c` disponible à la racine du projet doit être compilé pour fournir un exécutable `vernam`.

```php
<?php // classes/Dal/Dao.php
namespace Dal ;
use \PDO ;
class Dao
{

    /* Database credentials */
    private static $hostname = "localhost";
    private static $username = "speededev";
    private static $password = "azerty321" ;
    private static $database = "blog" ;

    /* Global PDO Instance */
    private static $instance ;

    public static function getInstance() {

        if (! isset(self::$instance)) {
            self::$instance = new PDO(
                "mysql" .
                ":dbname=" . self::$database .
                ";host="   . self::$hostname ,
                self::$username ,
                self::$password ,
                [
                    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
                ]) ;
        }

        return self::$instance ;
    }

    /* Helpers */
    public static function execute($request, $params = []) {
        $pdo = self::getInstance() ;
        $st = $pdo->prepare($request) ;
        $st->execute($params) ;
        return $st ;
    }

    public static function lastInsertId()
    {
        $pdo = self::getInstance() ;
        return $pdo->lastInsertId()
    }
}
```

```php
<?php // classes/Model/Posts.php
namespace Model ;
class Post {
    public $id ;
    public $title ;
    public $author ;
    public $content ;
    public function __construct($id, $title, $author, $content) {
        $this->id      = $id ;
        $this->title   = $title ;
        $this->author  = $author ;
        $this->content = $content ;
    }

    private static function fromRow($row) {
        return new Post($id, $row["title"], $row["author"], $row["content"]) ;
    }
    public static function create($title, $author, $content) {
        \Dal\Dao::execute("insert into Post "
                    . "( title,  author,  content) values "
                    . "(:title, :author, :content)",
                    [
                        "title"   => $title,
                        "author"  => $author,
                        "content" => $content
                    ]) ;
        return self::read(\Dal\Dao::lastInsertId()) ;
    }
    public static function read($id) {
        $st = \Dal\Dao::execute("select * from Post where id = {$id}") ;
        return self::fromRow($st->fetch()) ;
    }
    public function update() {
        \Dal\Dao::execute("update Post set "
                    . "title    = :title, "
                    . "author   = :author, "
                    . "content  = :content "
                    . "where id = :id",
                    [
                        "id"      => $this->id,
                        "title"   => $this->title,
                        "author"  => $this->author,
                        "content" => $this->content
                    ]) ;
    }
    public function delete() {
        \Dal\Dao::execute("delete from Post where id = {$this->id}") ;
    }

    public static function readAll() {
        $res = [] ;
        foreach (\Dal\Dao::execute("select * from Post") as $row) {
            $res[] = self::fromRow($row) ;
        }
        return $res ;
    }

    public static function readLast($limit) {
        $res = [] ;
        foreach (\Dal\Dao::execute("select * from Post limit $limit") as $row)
{
            $res[] = $res[] = self::fromRow($row) ;
        }
        return $res ;
    }
}
```

```php
<?php // classes/Model/User.php
namespace Model ;
class User
{
    public static function get() {
        return @$_SESSION["user"] ;
    }

    public static function register($login, $password) {
        \Dal\Dao::execute("insert into User (login, pass) "
                        . " values (:login, :pass)",
                        [
                            "login" => $login,
                            "pass" => \Utils\Crypto::Hash($password)
                        ]) ;
        $_SESSION["user"] = $login ;
    }

    public function login($login, $password)
    {
        $st = \Dal\Dao::execute("select * from User where"
                            . " login = :login and"
                            . " pass  = :pass",
                        [
                            "login" => $login,
                            "pass" => \Utils\Crypto::Hash($password)
                        ]) ;

        if ($st->fetch() !== false) {
            $_SESSION["user"] = $login ;
        }
    }

    public function logout()
    {
        $_SESSION["user"] = null ;
    }

}

session_start() ;
```

```php
<?php // classes/Utils/Crypto.php

namespace Utils ;

class Crypto
{

    public function hash($password, $salt = "A1D0C6E83F027327D8461063F4AC58A6")
    {
        return \Utils\Exec::cmd("/var/www/vernam", $password, $salt) ;
    }

}
```

```php
<?php // classes/Utils/Exec.php

namespace Utils ;

class Exec
{

    public static function cmd($cmd, ...$args)
    {
        $cmdline = $cmd . " " . implode(" ", $args) ;
        $escaped = escapeshellcmd ($cmdline) ;
        error_log($escaped) ;

        return shell_exec($escaped) ;
    }

}
```

```php
<?php // classes/View/Page.php

namespace View ;

class Page {

    public function __construct($title)
    {
        echo "<html>" ;
        echo "<head>" ;
        echo "  <title>$title</title>" ;
        echo "</head>" ;
        echo "<body>" ;
        echo "  <h1>$title</h1>" ;
        if (\Model\User::get() != null) {
            echo "      <a href=\"/users/logout.php\">Se Déconnecter</a>" ;
            echo "      <a href=\"/posts/add.php\">Publier</a>" ;
        } else {
            echo "      <a href=\"/users/login.php\">Se connecter</a>" ;
            echo "      <a href=\"/users/register.php\">S'enregistrer</a>" ;
        }
    }

    public function __destruct()
    {
        echo "  <p>Powered by <em>Speed-e-Devs</em>.</p>" ;
        echo "</body>" ;
        echo "</html>" ;
    }

    public static function redirect($url)
    {
        http_response_code(302) ;
        header('Location: ' . $url);
        exit(0) ;
    }
}
```

```php
<?php // html/autoload.php

/** PSR-4 Autoloading
 *
 * Cette fonction permet, lorsqu'on utilise une classe, de charger le fichier
 * correspondant automatiquement.
 *
 * Le système va appeler la fonction enregistrée ci-après en lui passant le
 * nom de la classe en paramètre. La fonction va alors transformer ce nom
 * de classe en nom de fichier puis opérer l'inclusion de ce dernier.
 *
 * Par exemple, si le code exécuté contient une ligne de ce genre :
 *
 * ```php
 * $pdo = \Dal\Dao::getInstance() ;
 * ```
 *
 * La fonction sera appelée et le fichier `classes/Dal/Dao.php` sera inclu,
 * la classe sera alors disponible et le code pourra être exécuté normalement.
 *
 */
spl_autoload_register(function($classname) {

    $filename = dirname(__DIR__) . DIRECTORY_SEPARATOR
              . "classes"        . DIRECTORY_SEPARATOR
              . str_replace("\\", DIRECTORY_SEPARATOR, $classname)
              . ".php" ;

    if (is_file($filename)) {
        require $filename ;
    }

}) ;

```

```php
<?php // html/index.php

require_once "../autoload.php" ;

$page = new \View\Page("Liste des articles") ;

foreach (\Model\Post::readLast(10) as $post) {
    echo "<h2>{$post->title}</h2>" ;
    echo "<p>Par <em>{$post->author}</em>... " ;
    echo "<a href=\"/posts/show.php?id={$post->id}\">en savoir plus</a></p>" ;
}
```

```php
<?php // html/posts/add.php

require_once "../autoload.php" ;

if (\Model\User::get() != null && isset($_POST["title"])) {
    $post = \Model\Post::create(
                $_POST["title"],
                \Model\User::get(),
                $_POST["content"]
                ) ;
    \View\Page::redirect("/posts/show.php?id={$post->id}");
}

$page = new \View\Page("Publication") ;

?>
<form method="post" action="">
    <input type="hidden" name="page" value="add" />

    <p>Titre <input type="text" name="title" /></p>
    <p>Contenu <textarea name="content" ></textarea></p>

    <input type="submit"/>
</form>
```

```php
<?php // html/posts/del.php

require_once "../autoload.php" ;

$post = \Model\Post::read($_GET["id"]) ;

if (\Model\User::get() != $post->author && isset($_POST["confirm"])) {
    $post->delete() ;
    \View\Page::redirect("/post/index.php");
}

$page = new \View\Page("Publication") ;

?>
<form method="post" action="">
    <input type="hidden" name="page" value="add" />

    <p>Titre <input type="text" name="title" /></p>
    <p>Contenu <textarea name="content" ></textarea></p>

    <input type="submit"/>
</form>
```

```php
<?php // html/posts/index.php
require_once "../autoload.php" ;

$page = new \View\Page("Liste des articles") ;

foreach (\Model\Post::readAll() as $post) {
    echo "<h2>{$post->title}</h2>" ;
    echo "<p>Par <em>{$post->author}</em>... " ;
    echo "<a href=\"/posts/show.php?id={$post->id}\">en savoir plus</a></p>" ;
}
```

```php
<?php // html/posts/show.php

require_once "../autoload.php" ;

$post = \Model\Post::read($_GET["id"]) ;

$page = new \View\Page($post->title) ;
echo "<p>Par <em>{$post->author}</em></p>" ;
echo $post->content ;

if (\Model\User::get() != null) {
    echo "<p><a href=\"/posts/del.php?id={$post->id}\">Supprimer</a></p>" ;
}
```

```php
<?php // html/users/login.php

require_once "../autoload.php" ;


if (isset($_POST["username"])) {
    \Model\User::login($_POST["username"], $_POST["password"]) ;
    \View\Page::redirect($_SERVER['HTTP_REFERER']);
}

$page = new \View\Page("Connexion") ;

?><form method="post" action="">
    <p>Nom Utilisateur <input type="text" name="username"/></p>
    <p>Mot de passe <input type="password" name="password"/></p>
    <input type="submit"/>
</form>
```

```php
<?php // html/users/logout.php

require_once "../autoload.php" ;

if (isset($_POST["confirm"])) {
    \Model\User::logout() ;
    \View\Page::redirect($_SERVER['HTTP_REFERER']);
}

$page = new \View\Page("Connexion") ;

?><form method="post" action="">
    <p>Êtes vous sûr de vouloir vous déconnecter ?</p>
    <input type="submit" name="confirm"/>
</form>
```

```php
<?php // html/users/register.php

require_once "../autoload.php" ;


if (isset($_POST["username"])) {
    \Model\User::register($_POST["username"], $_POST["password"]) ;
    \View\Page::redirect($_SERVER['HTTP_REFERER']);
}

$page = new \View\Page("Enregistrement") ;

?><form method="post" action="">
    <p>Nom Utilisateur <input type="text" name="username"/></p>
    <p>Mot de passe <input type="password" name="password"/></p>
    <input type="submit"/>
</form>
```

```sql
--
-- mysql/setup.sql
--
-- Ce fichier SQL permet de créer la base de donnée utilisée par le blog.
--

drop   database blog ;
create database blog ;
use    blog ;

create table Post (
    id      int(11) auto_increment primary key,
    title   varchar(64),
    author  varchar(64),
    content text
) ;

create table User (
    id    int(11) auto_increment primary key,
    login varchar(64),
    pass  blob
) ;

INSERT INTO User ( login, pass ) VALUES ( "admin", "" );
```

```c
/**
 * \file vernam.c
 * \brief Vernam Cipher
 * \author speed-e-dev
 * \version 0.2
 *
 * Implementation du chiffrement de Vernam. Ce chiffrement a été prouvé
 * mathématiquement comme sûr en 1940 par Shannon.
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void vernam_encipher(char * data, char* mask){
    int i,j;
    for(i=0; i< strlen(data); i++){
        printf("%c", ((data[i] + mask[i%strlen(mask)]) % 26 ) + 'A');
    }
    printf("\n");
}

int main(int argc, char ** argv){

    if(argc < 3) {return 1;}

    int data_length = 10;
    char * data = (char*)malloc(data_length);
    strcpy(data, argv[1]);

    char * mask = (char*)malloc(data_length);
    strcpy(mask, argv[2]);

    vernam_encipher(data, mask);

    free (data);
    free (data);
    return 0;
}
```