# Objective: Recover the Web Ring

This objective gives you a ringside seat (well, pcap-side anyway) to a successful attack on a web server, some practice bypassing security in a web browser, and a chance to make your own attack against a web server.

For the pcap attack, we will see how well Wireshark and tshark complement each other as we switch back and forth between the GUI and CLI versions of the famous protocol analyzer.

The web browser attack will use Chrome developer tools, and the web server attack will be with Burp Suite.

A picture containing text, building

Description automatically generated

## Naughty IP

The artifacts (a packet capture and a web log) will be used for the next four objectives. Alabaster also has hints for you when you talk to him. You could solve a lot of these with the web logs, but packet captures are more fun so we will work with them.

Graphical user interface, text

Description automatically generated

A picture containing text, toy

Description automatically generated

Text

Description automatically generated

A green screen with white text

Description automatically generated with medium confidence

Both the [link in the objective and in Alabaster’s](https://storage.googleapis.com/hhc22_player_assets/boriaArtifacts.zip) discussion are the same, and give you the artifacts. This is the link from [Top Talkers](https://protocoholic.com/2018/05/24/wireshark-how-to-identify-top-talkers-in-network/) hint.

### Question

Use the Top Talkers hint and Wireshark to find the IP address of the attacker and enter it into the objective. For bonus points, do the same task in tshark.

### Answer

Open the packet capture file in Wireshark and select Statistics, Endpoints  
Graphical user interface, text, application

Description automatically generated with medium confidence

Select the IP tab and click on the Bytes column until is sorted with the biggest numbers at the top.  
Graphical user interface, application, table, Excel

Description automatically generated

The number two address is 18.222.86.32, which is the attacker according to the hint.

The trick with tshark is to read the pcap file and then follow the same sort of steps you would in Wireshark. In this case, we looked at statistics for endpoints. A search of the [tshark man pages](https://www.wireshark.org/docs/man-pages/tshark.html) for endpoints gives us this.  
Graphical user interface, text, application, email

Description automatically generated  
   
<snip>  
Text

Description automatically generated

Same answer as before!

## Credential Mining

In the next phase, the attacker attempts to login to the site [with brute force](https://owasp.org/www-community/attacks/Brute_force_attack). Alabaster has advice for us, to look for requests to /login.html. We also know the attacker’s IP from the previous question.

Graphical user interface

Description automatically generated

Text

Description automatically generated

### Question

Use Wireshark to find a display filter that will focus on requests to /login.html. What is the first username tried?

### Answer

We know that the attacker’s IP address is 18.222.86.32, and that we are looking for some sort of HTTP request. I use those two filters often so it is easy to construct a display filter for them.  
ip.addr==18.222.86.32 && http.request  
Graphical user interface, application, table, Excel

Description automatically generated

The string of POSTs to /login.html probably marks the start of the brute force attack. We can investigate further by looking specifically at /login.html. Find login.html in the request body, right-click, select Prepare as a Filter -> Selected, and we get a filter, http.request.uri == "/login.html".

Graphical user interface, text, application, Word, email

Description automatically generated

That give us this filter:  
http.request.uri == "/login.html" && ip.addr == 18.222.86.32

Lots of interesting data there.  
Graphical user interface, text, application, email

Description automatically generated

The first POST has the username alice and password phillip, and that is the answer to the objective.

For fun, we can extract the usernames and passwords attempted in the attack using tshark.  
tshark -r victim.pcap -Y 'ip.addr==18.222.86.32 && http.request.method==POST && http.request.uri == "/login.html"' -T fields -e 'urlencoded-form.value' | head

The option -Y is the display filter we used to select the packets in Wireshark, with the addition of http.request.method==POST to exclude GET requests. The option -T fields tells tshark we want to output fields, and the -e option urlencoded-form.value specifies the field. I found that field name from Wireshark as before, using Prepare as a Filter -> Selected. Note that our choices are only urlencoded-form, urlencoded-form.key, and urlencoded-form.value. The fields do not distinguish between password and username, so we will get both from tshark.  
Text

Description automatically generated

Text

Description automatically generated

Now we can easily count the usernames and passwords that the attacker used (no one asked, but…)  
Text

Description automatically generated

The attacker tried 9 usernames with 101 unique passwords (field 1 is the username, 2 is the password.)

## 404 FTW

In this attack, the attacker is using a tool like [Nikto](https://github.com/sullo/nikto) or [dirbuster](https://github.com/KajanM/DirBuster) to try paths from a wordlist in the URI. It appears that this attack worked and led to compromise of the server.

A picture containing timeline

Description automatically generated  
<https://owasp.org/www-community/attacks/Forced_browsing>

Text

Description automatically generated

### Question

What is the URL path that the attacker found?

### Answer

Find an HTTP Status Code in a server reply and use the Prepare a Filter -> Selected method to create a display filter. Be sure to check for a response code of 404.  
Table

Description automatically generated

It appears the attack started with packet 23355 at time 175.372379. Any 200 OK after that, directed to the attacker’s IP 18.222.86.32, is a successful result for the attacker. The next search is for 200 OK responses to the attacker. There are a lot of OK responses before the attack, so I ruled them out with frame.number > 23355 (you could just scroll past the earlier frames, though.)  
Graphical user interface, application, table, Excel

Description automatically generated

The first OK after 23355 is 26774, shown here.  
Graphical user interface, text, application, email

Description automatically generated

Note that Wireshark kindly inserts the Request URI, <http://www.toteslegit.us/proc>. The answer for the objective is /proc, as they only want the URL path.

If you like tshark, this will build a list of all the response codes and what was requested. Then you can scroll through it to see where the attack started.  
tshark -r victim.pcap -Y 'ip.addr == 18.222.86.32 && http.response.code' -T fields -e frame.number -e http.response.code -e http.response\_for.uri  
Graphical user interface, text

Description automatically generated

You can scroll through the output and see that the attack started at frame 23355  
Text

Description automatically generated

Then you can use the same display filter as we used in Wireshark. The first 200 OK that appears is the answer we seek.  
tshark -r victim.pcap -Y 'frame.number > 23355 && ip.addr==18.222.86.32 && http.response.code==200' -T fields -e frame.number -e http.response.code -e http.response\_for.uri | head  
Text

Description automatically generated

## IMDS, XXE, and Other Abbreviations

The attackers now use the reconnaissance from the previous attacks to compromise the AWS credentials for this server. Pay attention to the XXE Injection in this attack since you will see it again.

Text

Description automatically generated<https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing>

Text

Description automatically generated  
A picture containing graphical user interface

Description automatically generated

Here is the [IMDS link](https://www.sans.org/blog/cloud-instance-metadata-services-imds-/) from Alabaster’s hint. It is also a good article about AWS attacks in general, well worth the read.

### Question 1

Use the IP address from the hint about IMDS and search for it in the body of the request. The ip.addr display filter will not be enough as it only searches the IP header. You can use Edit -> Find Packet but be careful to search the packet bytes. Another way is to put frame contains “searchTermHere” into the display filter.

### Answer 1

The address in the article is 169.254.169.254. It is an [APIPA address](https://study-ccna.com/apipa-automatic-private-ip-addressing/), so it is only available from within the AWS infrastructure and not from the public Internet.

Graphical user interface

Description automatically generated

This is interesting because it captures traffic between the server and its backend infrastructure as well as the attack packets. Note that attack traffic is all Content-Type: application/xml, POST traffic.

### Question 2

Filter on the attacker IP and Content-Type: application/xml. Find the URL the attacker forced the server to fetch (and disclosed AWS credentials.)

### Answer 2

In Wireshark, you can use the display filter  
ip.addr==18.222.86.32 && http.content\_type == "application/xml"  
To find the packets we want. Then you can Follow -> HTTP stream on each packet to see what the attacker did. Or you can use tshark to see them all at once. My ‘Prepare as a Filter -> Selected’ technique did not work well here. Using -e xml showed nothing but the characters, xml. I had to back up one step and use File Data to find my filter, http.file\_data  
Graphical user interface, text, application, email

Description automatically generated

tshark -r victim.pcap -Y 'ip.addr==18.222.86.32 && http.content\_type == "application/xml"' -T fields -e http.file\_data  
Text

Description automatically generated

We can get rid of noise with | cut -d '"' -f 6 (column delimiter “, field 6)Text

Description automatically generated

After a bit of reconnaissance (icanhazip.com returned the IP address of the server), the attacker walked his way down the AWS path until he found the EC2 credentials. This is from Follow -> HTTP Stream on the last packet.  
Text, letter

Description automatically generated

Oops! Keys exfiltrated! Paste this into the objective to get credit:  
<http://169.254.169.254/latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance>

This is the payload for a successful XXE attack—remember it for later.  
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [

<!ELEMENT foo ANY >

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

<foo>&xxe;</foo>

## Boria Mine Door

This challenge is an exercise in using the web developer tools available in major browsers. I chose to use the tool in Chrome. You will learn about some basic methods to protect a site from attacker’s input and some methods to bypass protections.

Graphical user interface, text

Description automatically generated

A picture containing timeline

Description automatically generated

Text

Description automatically generated

If you succeeded in helping Alabaster Snowball with his intrusion and artifacts, he has a lot of hints to offer for the Mine Door.

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

A green field with white text

Description automatically generated with medium confidence

<https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html>

Text

Description automatically generated with medium confidence

### Question Pin 1

The hints are explicit; we should check the HTML and JavaScript code and consider how to pick the locks. For Pin 1, look at pin1.html and pin1.js. Also, right click on the image for pin 1 and select Inspect; it will take you straight to the code for pin 1 in the DOM tree. (I am using Chrome dev tools.)

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Sometimes the developer leaves helpful comments.

What do you need to enter to unlock the first pin?

### Answer Pin 1

In this case, a comment does give the answer. Copy @&@&&W&&W&&&& from the comment and paste it into the window.

Text

Description automatically generated

Text

Description automatically generated

### Question Pin 2

Look at the code for Pin 2 and see if you can find useful information. When you are looking for things in a header, like Content Security Policy, look in the header just above the pin you are interested in and not in the headers at the top of the document. People found many ways to draw lines in the windows for the pins. I saw a hint from another player about using [SVG drawings](https://www.w3schools.com/graphics/svg_intro.asp) and I chose to use them for the rest of the challenge.

What kind of input is likely to slip through uninspected?

### Answer Pin 2

There is a nice hint in the Pin 2 comments. It isn’t a clear giveaway like Pin 1, but it does help. It is nice to know that HTML will be unobstructed.

Graphical user interface, text, application

Description automatically generated

I started creating SVG for this challenge with an online editor but found some simple constructs that allowed me to create my SVG by hand, and more quickly. Compare this SVG  
<svg viewBox="0 0 300 300"><line x1="0" y1="100" x2="300" y2="230" stroke="white" stroke-width="40"/></svg>

with one from an online editor:  
<svg id="eCvtJ2qdyUW1" xmlns="<http://www.w3.org/2000/svg>" xmlns:xlink="<http://www.w3.org/1999/xlink>" viewBox="0 0 300 300" shape-rendering="geometricPrecision" text-rendering="geometricPrecision"><line x1="-150" y1="-70.629552" x2="150" y2="50" transform="translate(150 181.451613)" fill="#fff" stroke="#fff" stroke-width="10"/></svg>

Both work. I prefer the simple version.  
<svg viewBox="0 0 300 300">

<line x1="0" y1="100" x2="300" y2="230" stroke="white" stroke-width="40"/>

</svg>

The simple SVG code sets a view box with (0, 0) in the upper left corner and (300, 300) in the bottom right. Note that this does not exactly match the size of the black image from the web page. The bottom y coordinate of the black image seems to be about 250.

### Question Pin 3

What input sanitization may be missing from Pin 3? What is a simple SVG that will connect the two sides? Note that the pins are sensitive to color.

### Answer Pin 3

The code for Pin 3 indicates that that JavaScript may not be filtered.  
Graphical user interface, text, application

Description automatically generated

This line works for Pin 3. Apparently SVG works in both HTML and JavaScript.

<svg viewBox="0 0 300 300" >

<line x1="0" y1="140" x2="300" y2="30" stroke="blue" stroke-width="40"/>

</svg>

### Question Pin 4

Examine the code for Pin 4. The helpful comments are gone now, and some developer-supplied input filtering is in place. It uses the JavaScript method someString.replace(). There is an easy way around the filter they put in place. What is a line of input that will bypass filtering and work as expected?

### Answer Pin 4

For Pin 4, the Content-Security-Policy is gone, and the developer has written their own code for filtering. It removes quotes and angle brackets using the JavaScript [string replace method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/replace). Graphical user interface, text, application

Description automatically generated

This comment from the documentation is especially useful.  


This developer did not use the g flag, so the method only replaces the first character it finds, and subsequent characters pass through unaltered. So, we can add <>” to the beginning of our SVG and evade the filter.

<>"<svg viewBox="0 0 300 300">

<line x1="0" y1="60" x2="300" y2="60" stroke="white" stroke-width="40"/>

<line x1="0" y1="200" x2="300" y2="200" stroke="blue" stroke-width="40"/>

</svg>

### Pin 5 Question

This time you will find that the developer remembered the g flag, or global flag. All occurrences of the filtered characters will be removed. It is difficult to avoid. However, if you follow the source and find the event that triggers a call to the filtering, perhaps you can find a clever way to evade it.

### Pin 5 Answer

Here is the DOM code for Pin 5, taken from the Elements column of Chrome dev tools.  
Graphical user interface, text, application

Description automatically generated

The sanitizeInput() function does the filtering and is difficult to evade because it uses the g, or global, flag to replace all occurrences of the characters. However, the HTML element that calls the sanitizeInput() function is onblur. The onblur event occurs when the object loses the focus; in our case when the mouse pointer leaves the text field to click on the Go button. See [this link](https://www.geeksforgeeks.org/html-dom-onblur-event/) for a nice demonstration.

Since the filtering occurs when the mouse moves out of the text field, we can avoid that by pressing Enter instead of using the mouse to click Go. This method would have worked for Pin 4 as well.

Do not click Go, instead press Enter after pasting this.  
<svg viewBox="0 0 300 300">

<line x1="0" y1="200" x2="300" y2="60" stroke="red" stroke-width="40"/>

<line x1="50" y1="250" x2="300" y2="140" stroke="blue" stroke-width="40"/>

</svg>

### Pin 6 Question

Is there any client-side filtering in place for this pin? Can you draw the lines with SVG?

### Pin 6 Answer

I could not find any client-side filtering. If there is server-side filtering, it does not object to SVG, which seems to bypass a lot of the protections the challenge designer implemented.

Graphical user interface, text, application

Description automatically generated

This code will draw the three lines we need.  
<svg viewBox="0 0 300 300">  
<line x1="0" y1="50" x2="300" y2="50" stroke="#00ff00" stroke-width="40"/>  
<line x1="0" y1="120" x2="300" y2="160" stroke="red" stroke-width="40"/>  
<line x1="0" y1="180" x2="250" y2="270" stroke="blue" stroke-width="40"/></svg>

## Glamtariel’s Fountain

You have watched an XXE Injection attack through packet captures. Now it is time to execute one. You will use XXE Injection to retrieve the golden ring from Glamtariel’s Fountain.

Text

Description automatically generated

A picture containing text, toy

Description automatically generated

Hal Tandybuck gives you several hints after you unlock all six pins in the Boria Mine door.

Table

Description automatically generated with medium confidence  
<https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing>

Words from Glamtariel in upper case may be helpful hints. I think they are too vague and will give you additional clues.

Text

Description automatically generated

A picture containing text

Description automatically generated

You will need to intercept, change, and resend many packets to the fountain website to complete the XXE attack. Two of the tools available that do this are Burp Suite and the Edit/Resend feature in Firefox. Since we will need to make many changes to the traffic in this challenge, we will use Burp. We will use the Firefox function in the challenge, Exploit a Smart Contract.

Burp Suite is pre-installed on Kali Linux, but the Burp browser was not working properly on the version I had. It is good to install your own tools anyway. The Burp Suite Community Edition is [available here](https://portswigger.net/burp/communitydownload), for both Windows and Linux. I chose Linux, well, because.

To install the Linux version, execute burpsuite\_community\_linux\_v2022\_12\_4.sh and then follow the prompts in the GUI wizard.  
.Graphical user interface, text, application

Description automatically generated

The installer leaves a folder on your Desktop called BurpSuiteCommunity. Execute the file BurpSuiteCommunity to get going.Graphical user interface

Description automatically generated with low confidence  
Graphical user interface, text, application, email

Description automatically generated

If you need help, it is [available here](https://portswigger.net/burp/documentation/desktop/getting-started).

Click through the Next and Start prompts to get to the Burp dashboard. Once you are there, click on the Target tab.  
Graphical user interface, text, application

Description automatically generated

Burp now comes with its own Chromium browser. This is handy since you no longer need to configure a browser to proxy through Burp and to accept Burp’s Certificate Authority (CA) certificate. Click the Open Browser button. Now you should have windows for Burp and the browser, and if you open the Kringlecon site you will see it fill the Target tab in Burp.  
A picture containing graphical user interface

Description automatically generated

Now we are ready to get started.

### Question 1: Catch a Drop

Find your way to the fountain in HHC and click on it. A new tab for Glamtariel’s Fountain will open. Talk to Glamtariel and the fountain by dragging the icons in the top right corner with your mouse and dropping them on Glamtariel or the fountain.

Diagram

Description automatically generated with low confidence

Find the traffic that your drag and drop causes in the Target tab of Burp.

What is the payload and Content-Type in the POST request that the drag and drop triggers?

### Answer 1: Catch a Drop

The browser sends a POST request to /dropped of the fountain site.  
Graphical user interface, application

Description automatically generated

The POST request has a header, Content-Type: application/json, and the payload is indeed JSON,  
{"imgDrop":"img2","who":"princess","reqType":"json"}.  
It may seem strange that we are going to use an attack, XXE Injection, that requires XML when the site uses JSON, but hang in there. Maybe they forgot to turn XML off. The payload tells us which icon was dropped (img2), who received it (princess), and that the request type is JSON.

### Question 2: Play the Game

You must drop each icon on each person (princess, fountain), receive a new set of icons, and repeat before the site will allow you to start the XXE attack. Pay attention to the name of the file that presents the frozen eye. Why do you think it may mean you are ready to start XXE?

### Answer 2: Play the Game

After dragging and dropping for a while, we see the frozen eye.  
Text

Description automatically generated

We can recover the name of the image from either Burp or the browser. It is  
<https://glamtarielsfountain.com/static/images/stage2ring-eyecu_2022.png>

It appears we have started stage 2. Continue to drop until the icons change to four rings.

Text

Description automatically generated

### Question 3: Convert to XML

Before you try to send an XXE attack to the server, see if you can get the site to accept a normal transaction, but translated into XML. Translate this to XML:  
{  
 "imgDrop":"img1",  
 "who":"princess",  
 "reqType":"json"  
}

If you are not familiar with XML, you may find [this converter site](https://www.convertjson.com/json-to-xml.htm) helpful. Compare your XML to the XXE attack at the end of the objective, IMDS, XXE, and other Abbreviations. Where would you insert your XXE?

### Answer 3: Convert to XML

Our simple JSON converts to this in XML:

<?xml version="1.0" encoding="UTF-8" ?>

<root>

<imgDrop>img1</imgDrop>

<who>princess</who>

<reqType>xml</reqType>

</root>

I manually changed reqType to xml. The root and /root can be named anything, but root is common.

### Question 4: Insert XML

Burp has two functions that can replace the content the browser sends to the server with our own: proxy and repeater. We will use proxy first.  
Graphical user interface, text, application

Description automatically generated

When we turn Intercept on, packets will be held in Burp and not sent on to the website until we click the Forward button. While the packets are held, we can change them however we want, and then forward them when we are done. Browsers like to send heartbeat traffic on their own so we may intercept extraneous traffic. Just click Forward on those packets until the packet of interest appears.

Send the translated XML without XXE to the website. Don’t forget to change Content-Type, and do not change Accept, since the princess always replies in JSON. Is your XML accepted?

### Answer 4: Insert XML

First, turn on Intercept in Burp proxy. Then generate a POST to /drop by dragging an icon to the princess. The POST should appear.

Oops, caught a heartbeat. Let it go by pressing Forward.  
Graphical user interface, text, application

Description automatically generated

Here is the original packet, caught in the proxy.  
Graphical user interface, text, application, email

Description automatically generated

Now, change Content-Type to application/xml, and change the JSON payload to our XML version.  
Graphical user interface, text, application, email

Description automatically generated

Now click Forward and see if the server accepts it. Once you’ve clicked Forward, turn Intercept off so the normal traffic can resume. If you forget to turn Intercept Off, it will appear that your browser is broken (a disadvantage of proxy.) If the XML is accepted, you will see this when you return to Target.  
Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generated

If you see anything else, the XML was probably refused. If the site talks about TAMPER, you will probably have to use the Reset button on the site and do all the draggy/droppy stuff again. Sorry.

### Question 5: Guess the PATH

Quite often, penetration testers do not have a view of the source code or file structure of the systems they attack. They are working in the dark and must do a lot of guessing. I think the writers of this challenge are simulating that for us. The words that Glamtariel and the fountain speak in bold letters are supposed to be enough for us to guess the path to our target.

*"I don't know why anyone would ever ask me to TAMPER with the cookie recipe. I know just how Kringle likes them.^Glamtariel likes to keep Kringle happy so that he and the elves will visit often."*

*"Kringle really likes the cookies here so I always make them the same way.^Kringle really dislikes it if anyone tries to TAMPER with the cookie recipe Glamtariel uses."*

*"I don't get away as much as I used to. I think I have one last trip in me which I've probably put off for far too long.^The elves do a great job making PATHs which are easy to follow once you see them."*

*"I helped the elves to create the PATH here to make sure that only those invited can find their way here.^I wish the elves visited more often."*

*"Did you know that I speak in many TYPEs of languages? For simplicity, I usually only communicate with this one though.^I pretty much stick to just one TYPE of language, it's a lot easier to share things that way."*

*"Wow!, what a beautiful silver ring! I don't have one of these. I keep a list of all my rings in my RINGLIST file. Wait a minute! Uh, promise me you won't tell anyone.^I never heard Glamtariel mention a RINGLIST file before. If only there were a way to get a peek at that."*

*"I like to keep track of all my rings using a SIMPLE FORMAT, although I usually don't like to discuss such things.^Glamtariel can be pretty tight lipped about some things."*

Many people I helped in the contest (including myself) found it difficult to guess the path based on these hints. Additionally, files that would have normally been accessible to test the XXE were not accessible. For example, static/js/ajax.js (visible in Target) was not accessible, even with the correct root. I wrote these additional clues that helped some players guess the correct path without further spoilers.

*Be careful you don't trip over a ROOT on the PATH or you will be disAPPointed.*

*I like to keep my things with my IMAGES.*

*RINGLIST and SIMPLE FORMAT go together.*

Be aware that web apps are often kept in containers or jails. The usual /var/www/html is not visible to them.

What is the path to the file we seek?

### Answer 5: Guess the PATH

The hint starting with “Be careful” has the characters ROOT, PATH, and APP, in upper case. This could be interpreted as “the root of the path is app.” So, we have <file:///app>.

The hint, “I like to keep my things with my images,” tells us the middle of the path. If you look on the Target tab in Burp, you will see the images are kept in static/images. <file:///app/static/images>.

Finally, RINGLIST and SIMPLE FORMAT give a clue about the name of the file. We are expecting a text file, so ringlist.txt. The path is <file:///app/static/images/ringlist.txt>. Easy, yes?

### Question 6: XXE with Repeater

Comparing our XML to the XXE attack in *IMDS, XXE, and Other Abbreviations* makes XXE seem a good possibility.

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///app/static/images/ringlist.txt" >]>

<root>

<imgDrop>&xxe;</imgDrop>

<who>princess</who>

<reqType>xml</reqType>

</root>

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE foo [

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

<foo>  
 &xxe;  
</foo>

The reference to the XXE, &xxe; could be inserted into any of the normal fields. In this case it is in imgDrop, in place of the normal content.

I found [this reference](https://portswigger.net/web-security/xxe) on XXE to be very helpful. The XML external entity (XXE) syntax allows the XML to reference content from other locations like websites and internal file systems. The location of the external content is specified in the header as a DOCTYPE. The XML inserts the content using a reference to the name before SYSTEM, xxe in this case. So, &xxe; is inserted into one of the normal XML fields.

Burp Proxy is nice, in that the replies return to the browser so we can view them there if we wish. Since we are in the middle of the transaction and not sending duplicate packets, cookie changes will not bother us, if they occur. Proxy requires a lot of clicking, however. Burp Repeater is nice, since we can select a packet, make changes, send, and see the code results immediately with just one click. If we make errors, just fix the errors, and click send again. Nice!

To use Repeater, select a drag/drop POST packet in the Target list, right-click, and choose Send to Repeater. Then make your changes and click Send.  
Graphical user interface, application

Description automatically generated

Send a packet to the server with Content-Type: application/xml and the XXE in the payload.

### Answer 6: XXE with Repeater

The changed packet should look like this.  
Graphical user interface, text, application, email

Description automatically generated

The response should be:  
Graphical user interface, text, application, email

Description automatically generated

Note that Glamtariel sent us a link to an image, static/images/pholder-morethantopsupersecret63842.png. You can recover that from the browser.  
Graphical user interface

Description automatically generated with low confidence

### Question 7: More Hoops

Use XXE to retrieve the files in the folder, x\_phial\_pholder\_2022. Obvious filenames to try are bluering.txt and redring.txt. They probably live within //app/static/images.

### Answer 7: More Hoops

The bluering.txt file does not help much. Graphical user interface

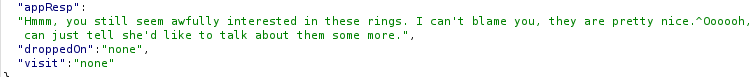
Description automatically generated with medium confidence

A picture containing text

Description automatically generated

The redring.txt file is not helpful either.  
Graphical user interface

Description automatically generated with low confidence



Maybe silvering.txt?  
A picture containing text

Description automatically generated

A picture containing text

Description automatically generated

Let’s look at the image static/images/x\_phial\_pholder\_2022/redring-supersupersecret928164.png.  
A picture containing graphical user interface

Description automatically generated

Note the text (defacement) inside the ring.   


It says, goldring\_to\_be\_deleted.txt.

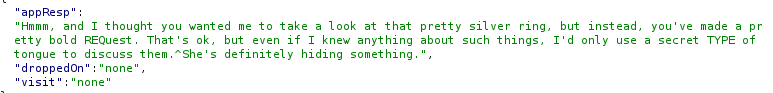
### Question 8: Yet Another Hoop

Try to retrieve goldring\_to\_be\_deleted.txt.

### Answer 8: Yet Another Hoop

Look for the new file.  
A picture containing text

Description automatically generated



Another vague clue.

### Question 9: The Last Hoop

Players had a lot of trouble with the bold REQuest, secret TYPE clue. I wrote this clue to help them.

*I like your eXXEllent language ability. Let's move to REQ TYPE for secrecy.*

### Answer 9: The Last Hoop

My intent with the last clue was to tell the player to move their &xxe; out of imgDrop and into the reqType field. We cannot leave the imgDrop field empty, so let us put the filename for a generic silver ring, img1, into it.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

At last. static/images/x\_phial\_pholder\_2022/goldring-morethansupertopsecret76394734.png Graphical user interface, application, icon

Description automatically generated

Put this into the objective:  
goldring-morethansupertopsecret76394734.png

Whew!

A picture containing text, device, meter, gauge

Description automatically generated