UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE DOMENIUL: CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI SPECIALIZAREA: TEHNOLOGIA INFORMAȚIEI

**SIMULAREA ATACURILOR WI-FI**

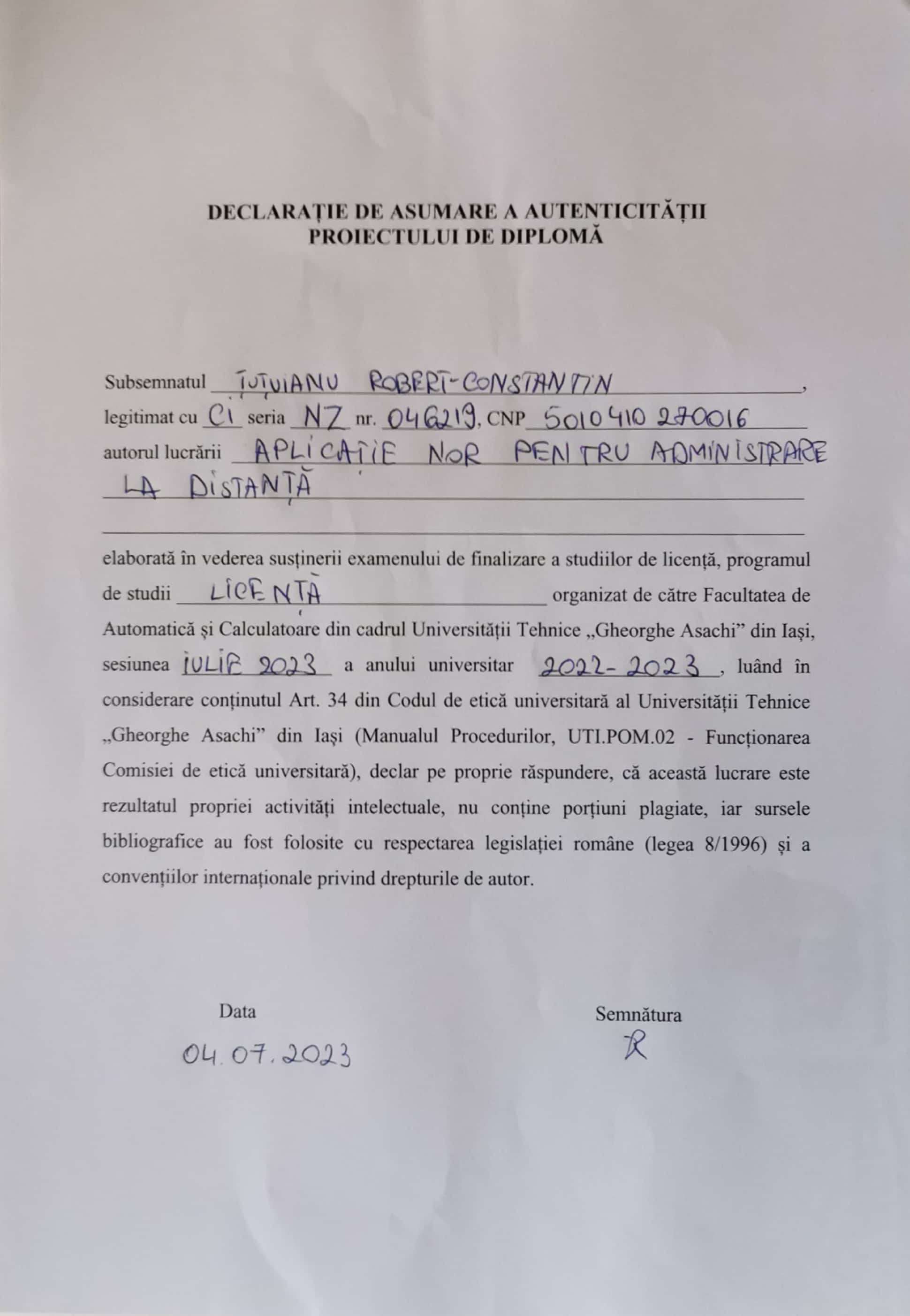
**UTILIZAND ESP32**

LUCRARE DE DIZERTAȚIE

Coordonator științific, absolvent,

conf. dr. ing. Mihai Horia Zaharia ing.dipl. Țuțuianu Robert Constantin

Iași, 2025

****

Cuprins

[1. Introducere 1](#_Toc554191665)

[2. Documentarea problemei 2](#_Toc976728025)

[2.1 Aplicatii software și hardware pentru testarea securității Wi-Fi 2](#_Toc1489103763)

[2.1.1 Aircrack-ng 2](#_Toc492220428)

[2.1.2 Wireshark 2](#_Toc1599578674)

[2.1.3 Wifite 3](#_Toc1991078513)

[2.1.4 ESP8266 Deauther 3](#_Toc97341601)

[2.1.5 Alte soluții 3](#_Toc1876198843)

[2.2 Poziționarea proiectului actual 4](#_Toc447416159)

[3. Abordare teoretică 5](#_Toc523343132)

[3.1 Utilizarea aplicatiei 5](#_Toc2093432715)

[3.2 Scenarii de Utilizare 5](#_Toc155079736)

[3.2.1 Utilizator neconectat la ESP32 6](#_Toc2120499200)

[3.2.2 Utilizator conectat la ESP32 fără a accesa pagina web 6](#_Toc11054632)

[3.2.3 Utilizator accesează pagina web ESP32 6](#_Toc1286436986)

[3.2.4 Activarea și oprirea atacului AP Flood 7](#_Toc540192721)

[3.2.5 Scanarea AP-urilor și atacul de tip Deauthentication 8](#_Toc1201526057)

[3.2.6 Afisarea credentialelor 9](#_Toc278632551)

[3.2.7 Victimă conectată la ESP32 și redirecționată prin DNS Hijacking 10](#_Toc678475663)

[4. Metode folosite 11](#_Toc521218763)

[4.1. Cum funcționează AP-urile și cum trimit ele mesaje către dispozitive 11](#_Toc1182570925)

[4.2. Management frames 12](#_Toc2129470210)

[4.3. Beacon frames 13](#_Toc1060253158)

[4.4. AP Flood prin beacon-uri 14](#_Toc1273040295)

[4.4.1 Realizarea in practica 14](#_Toc439080163)

[4.4.2 Structura funcțională a atacului în proiect 15](#_Toc1385877530)

[4.4.3 Mecanismul de execuție 16](#_Toc1528933113)

[4.4.4 Observații asupra limitelor 16](#_Toc2070866138)

[4.5. Deauthentication frames 17](#_Toc45326034)

[4.6. Atacurile de deautentificare 17](#_Toc682136481)

[4.6.1 Realizarea in practica 18](#_Toc461026387)

[4.6.2 Efectul atacului 19](#_Toc599517699)

[4.8. Mesajele DNS 20](#_Toc1658864507)

[4.9. DNS Hijacking 22](#_Toc1950130150)

[4.9.1 Realizarea in practica 23](#_Toc1091738127)

[4.10 Arhitectura aplicației server-side pe ESP32 24](#_Toc1110959533)

[4.10.1 Serverul Web Principal 25](#_Toc1793725016)

[4.10.2 Serverul Facebook Fals 26](#_Toc1189079728)

[4.10.3 Serverul DNS 27](#_Toc839674728)

[5. Rezultate obținute și analiza datelor 28](#_Toc93657527)

[5.1. Testarea stabilitatii in timp 28](#_Toc1712422861)

[5.2. Testarea consumului de memorie si CPU 29](#_Toc785623795)

[5.3. Testarea validarii datelor introduse de utilizator 29](#_Toc248737263)

[5.4. Testarea comportamentului aplicației la execuții succesive și simultane 31](#_Toc1685438413)

[5.5. Testarea adaptării interfeței web la variații de densitate grafică și dispozitive 32](#_Toc796678427)

[6. Discuţii şi concluzii 33](#_Toc1535174249)

[7. Bibliografie 34](#_Toc78147487)

# Introducere

In contextul unei societati tot mai dependente de mobilitate și conectivitate wireless, rețelele Wi-Fi au devenit nelipsite în viata noastra. Totuși, multe dintre aceste rețele rămân vulnerabile în fața unor atacuri simple, dar eficiente, din cauza unor brese ddise securitate în standardele de securitate sau a lipsei de configurare cu securitatea in minte.

Utilizatorii obișnuiți, de cele mai multe ori, nu conștientizează pericolele la care se expun atunci când se conectează la o rețea publică, deschisă sau necunoscută. În ultimii ani, atacurile cibernetice care au ca tinta rețelele Wi-Fi au cunoscut o creștere semnificativă, atât ca număr, cât și ca diversitate. De la simple atacuri care obliga deviceul sa se deautentifice de la reteaua la care este conectat, până la atacuri mai sofisticate, cum ar fi Evil Twin (crearea unui punct de acces fals care imită unul legitim)[9] sau DNS spoofing, vulnerabilitățile exploatate țintesc în special rețelele personale și cele publice neprotejate. Un exemplu celebru este atacul KRACK (Key Reinstallation Attack), care a expus lumii brese de securitate critice în protocolul WPA2 și a afectat milioane de dispozitive[1]. Aceste atacuri demonstrează cât de fragilă poate fi o infrastructură Wi-Fi aparent sigură și cât de ușor poate fi exploatată în lipsa unei securizări pe masura.

Lucrarea de fata demonstrează, printr-o abordare practică, vulnerabilități comune ale rețelelor Wi-Fi utilizând un microcontroller ESP32[10]. Acesta funcționează ca un punct de acces local, oferind o interfață de control pentru diverse acțiuni demonstrative, printr-un server web. Utilizatorul se conectează la rețeaua creată de microcontroller, iar prin intermediul paginii web poate interacționa cu funcționalitățile aplicației. Aceasta permite scanarea rețelelor din proximitate, deconectarea clienților conectați la un anumit punct de acces, redirecționarea cererilor DNS către o pagină de tip phishing și generarea masivă de rețele false.

Interacțiunea este intuitivă, imediat după conectare utilizatorul accesează interfața web și selectează modul de atac dorit. Toate operațiile sunt executate local, fără conexiune la internet, ceea ce permite o demonstrație complet controlată și sigură. Scopul principal este evidențierea riscurilor reale la care se expun utilizatorii în lipsa unei infrastructuri wireless securizate, oferind totodată un cadru pentru învățare, testare și conștientizare în domeniul securității wireless.

# Documentarea problemei

Rețelele wireless au cunoscut o evoluție accelerată în ultimele două decenii, devenind un instrument esențial al infrastructurii digitale moderne. Primele standarde Wi-Fi, precum IEEE 802.11b, lansat în 1999, ofereau viteze modeste și metode de criptare precum WEP (Wired Equivalent Privacy), care s-au dovedit rapid ineficiente. În timp, acestea au fost înlocuite cu standardele WPA și ulterior WPA2, care au adus îmbunătățiri semnificative, dar aveau si ele o multime de probleme. În anul 2017, atacul KRACK (Key Reinstallation Attack), detaliat de cercetătorii Vanhoef și Piessens, a demonstrat existența unei vulnerabilități critice în WPA2, afectând milioane de dispozitive la nivel global. (vezi [1], [2], [8])

În prezent, standardul WPA3 oferă un nivel superior de securitate, dar adoptarea lui este lentă. Multe dispozitive continuă să utilizeze protocoale mai vechi sau configurări nesigure, ceea ce lasă rețelele expuse. În paralel, utilizatorii se conectează frecvent la rețele publice sau deschise, fără a înțelege riscurile de securitate asociate precum interceptarea traficului, capturarea de date sensibile sau redirecționarea către site-uri malițioase. Conform Wi-Fi Alliance, WPA3 este disponibil din 2018, dar multe dispozitive nu oferă suport complet pentru acesta [8].

## **2.1 Aplicatii software și hardware pentru testarea securității Wi-Fi**

Pentru analiza și testarea vulnerabilităților rețelelor wireless, comunitatea de securitate a dezvoltat numeroase unelte open-source și comerciale. Mai jos sunt prezentate cele mai relevante, împreună cu avantajele și limitările acestora.

### **2.1.1 Aircrack-ng**

Aircrack-ng este o suită de unelte pentru auditarea rețelelor wireless. Permite capturarea de pachete, efectuarea de atacuri de tip deauthentication și spargerea cheilor WEP/WPA prin metode de brute-force sau dictionary attack[3].

Avantaje:

- Suport extins pentru diverse tipuri de plăci de rețea.

- Poate recupera parole Wi-Fi reale în mod controlat.

- Este integrat în distribuții specializate precum Kali Linux.

Dezavantaje:

- Necesită cunoștințe avansate de rețelistică și Linux.

- Nu are interfață grafică, lucrând exclusiv în linie de comandă.

- Nu este portabil pe platforme embedded precum ESP32.

### **2.1.2 Wireshark**

Wireshark este cel mai utilizat analizator de trafic de rețea, permițând inspecția detaliată a pachetelor capturate, inclusiv cele Wi-Fi (802.11) [4].

Avantaje:

- Analiză protocolară foarte detaliată.

- Interfață grafică intuitivă.

- Utilizare profesională în audituri și depanări de rețea.

Dezavantaje:

- Nu include funcții de atac, doar analiză pasivă.

- Nu este adaptat pentru rularea pe dispozitive embedded.

- Poate fi dificil de utilizat pentru începători.

### **2.1.3 Wifite**

Wifite este o unealtă automatizată care combină mai multe instrumente (Aircrack-ng, Reaver, Bully) și permite lansarea rapidă de atacuri asupra rețelelor Wi-Fi din apropiere[7].

Avantaje:

- Automatizează întregul proces de testare.

- Ideal pentru demonstrații și testări rapide.

- Suportă multiple metode de atac asupra WPA și WEP.

Dezavantaje:

- Este limitat ca opțiuni avansate de configurare.

- Necesită drivere speciale și compatibilitate hardware.

- Nu suportă protocoale moderne precum WPA3.

### **2.1.4 ESP8266 Deauther**

ESP8266 Deauther este un proiect open-source dezvoltat pentru microcontrollerul ESP8266, care permite lansarea de atacuri Wi-Fi (deauthentication, beacon flood, probe request flood) printr-o interfață web ușor de folosit.[6]

Avantaje:

- Extrem de portabil și ieftin (ESP8266 costă 3-5 euro).

- Ușor de folosit pentru demonstrații didactice.

- Interfață web integrată și prietenoasă.

Dezavantaje:

- Limitat la câteva tipuri de atacuri simple.

- Nu permite captură de trafic sau analiză complexă.

- Nu suportă funcționalități precum DNS hijacking.

- ESP8266 are resurse hardware limitate față de ESP32.

### **2.1.5 Alte soluții**

Există și soluții comerciale precum Hak5 WiFi Pineapple, care oferă o gamă larga de funcționalități într-un pachet profesionist. Acestea sunt însă costisitoare, greu de personalizat și nu sunt potrivite pentru utilizare academică sau educațională open-source. [5]

## **2.2 Poziționarea proiectului actual**

Proiectul dezvoltat în cadrul acestei lucrări își propune să ofere o soluție portabilă, ușor de utilizat și versatilă pentru testarea și înțelegerea vulnerabilităților rețelelor wireless. Platforma hardware aleasă, ESP32, oferă resurse superioare față de ESP8266 și suport pentru mai multe interfețe de rețea și operațiuni concurente.

Prin intermediul unui server web integrat, utilizatorul poate:

- vizualiza rețelele Wi-Fi din proximitate;

- selecta o rețea și executa un atac de tip deauthentication;

- activa redirecționarea DNS către o pagină de phishing (ex: Facebook);  
 - genera automat rețele Wi-Fi false (AP flood) pentru simularea unui atac de confuzie.

Spre deosebire de alte soluții, aplicația rulează complet local, fără acces la internet, ceea ce permite testarea într-un mediu izolat și controlat. Prin această abordare, proiectul combină elemente de rețelistică, securitate cibernetică și dezvoltare embedded într-un instrument educațional eficient și accesibil.

# Abordare teoretică

## **3.1 Utilizarea aplicatiei**

Aplicațiile pentru testarea și efectuarea atacurilor asupra rețelelor Wi-Fi joacă un rol esențial în identificarea și înțelegerea vulnerabilităților rețelelor wireless. Acestea ofera numeroase oportunități practice și educaționale în diverse domenii, mai jos fiind prezentate câteva exemple relevante de utilizare a unor astfel de aplicații:  
 **Auditarea securității rețelelor wireless** – Aplicațiile care efectuează atacuri Wi-Fi pot fi utilizate pentru auditarea securității unei rețele și identificarea punctelor slabe. Administratorii de rețea pot astfel simula atacuri reale, cum ar fi cele de care deautentifica deviceurile de la retea sau DNS spoofing, pentru a verifica dacă infrastructura de rețea este configurată corect și securizată corespunzător. De exemplu, o companie poate folosi astfel de instrumente pentru a testa periodic rețelele interne și pentru a remedia rapid eventualele vulnerabilități identificate.  
 **Educație și conștientizare** – Aceste aplicații sunt extrem de utile în mediile educaționale și academice pentru a demonstra vulnerabilitățile reale ale rețelelor wireless. Profesorii pot realiza demonstrații practice care arată cât de ușor pot fi exploatate anumite vulnerabilități, scotand in evidenta necesitatea unor măsuri de securitate mai bune. De exemplu, studenții pot învăța cum funcționează atacurile Wi-Fi, observând efectul unui atac DNS hijacking sau al unui atac de tip Evil Twin într-un laborator.  
 **Cercetare și dezvoltare în securitate informatică** – Cercetătorii din domeniul securității cibernetice utilizează aceste instrumente pentru a realiza experimente și pentru a descoperi noi vulnerabilități sau metode de apărare. De exemplu, cercetătorii pot analiza și documenta comportamentul unor dispozitive sau rețele Wi-Fi atunci când sunt expuse la atacuri simulate, contribuind astfel la dezvoltarea unor standarde și protocoale mai sigure.  
 **Testarea dispozitivelor noi** – Dezvoltatorii pot utiliza astfel de aplicații pentru a testa securitatea dispozitivelor noi înainte ca acestea să ajungă pe piață. De exemplu, înainte de lansarea unui nou router Wi-Fi sau a unui dispozitiv IoT, producătorul poate testa dispozitivul folosind aceste instrumente pentru a se asigura că acesta rezistă atacurilor comune precum AP flood sau deauthentication.

## **3.2 Scenarii de Utilizare**

Actori implicați:  
 Utilizator – Persoană care interacționează intenționat cu ESP32.  
 Victimă (Utilizator neavizat) – Persoană care se conectează la ESP32 crezând că este o rețea legitimă.  
 ESP32 – dispozitivul ce oferă funcționalitățile descrise.

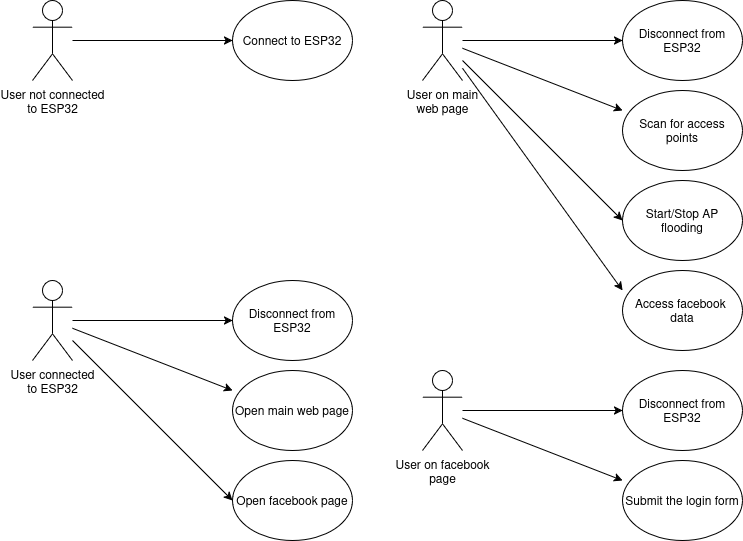


Fig 3.1

### **3.2.1 Utilizator neconectat la ESP32**

Actor: Utilizator  
Descriere: Utilizatorul nu este conectat la ESP32.  
Flux:  
 - utilizatorul caută rețele Wi-Fi.  
 - identifică rețeaua Wi-Fi generată de ESP32.  
 - se conectează la rețea.  
Rezultat: Utilizatorul este conectat la ESP32.

### **3.2.2 Utilizator conectat la ESP32 fără a accesa pagina web**

Actor: Utilizator  
Precondiție: Utilizator conectat deja la rețeaua ESP32.  
Descriere: Utilizatorul nu a accesat pagina web a ESP32.  
Flux:  
 - utilizatorul deschide un browser web.  
 - introduce manual adresa IP a ESP32 (ex: 192.168.4.1).  
Rezultat: Utilizatorul accesează pagina web servită de ESP32.

### **3.2.3 Utilizator accesează pagina web ESP32**

Actor: Utilizator  
Precondiție: Utilizator conectat și pagina web ESP32 accesată.  
Descriere: Utilizatorul accesează funcționalitățile oferite de pagina web.  
Flux principal:  
 - utilizatorul are opțiuni clare prezentate în interfață:  
 - poate să activeze atacul AP Flood.  
 - poate să scaneze AP-urile din apropiere.

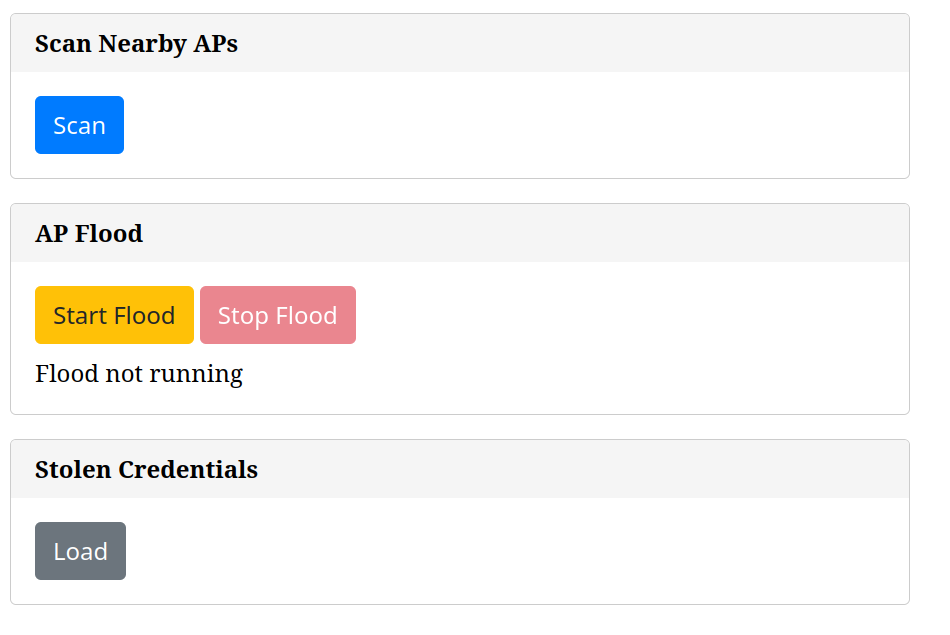


Fig 3.2

### **3.2.4 Activarea și oprirea atacului AP Flood**

Actor: Utilizator  
Precondiție: Utilizator în pagina web ESP32.  
Descriere: Utilizatorul pornește sau oprește atacul AP Flood.  
Flux:  
 - utilizatorul selectează „Start AP Flood”.  
 - ESP32 începe să genereze rețele false în mod continuu.  
 - utilizatorul poate apăsa ulterior „Stop AP Flood”.  
 - ESP32 oprește generarea rețelelor false.  
Rezultat: Atacul AP Flood este controlat complet de utilizator.

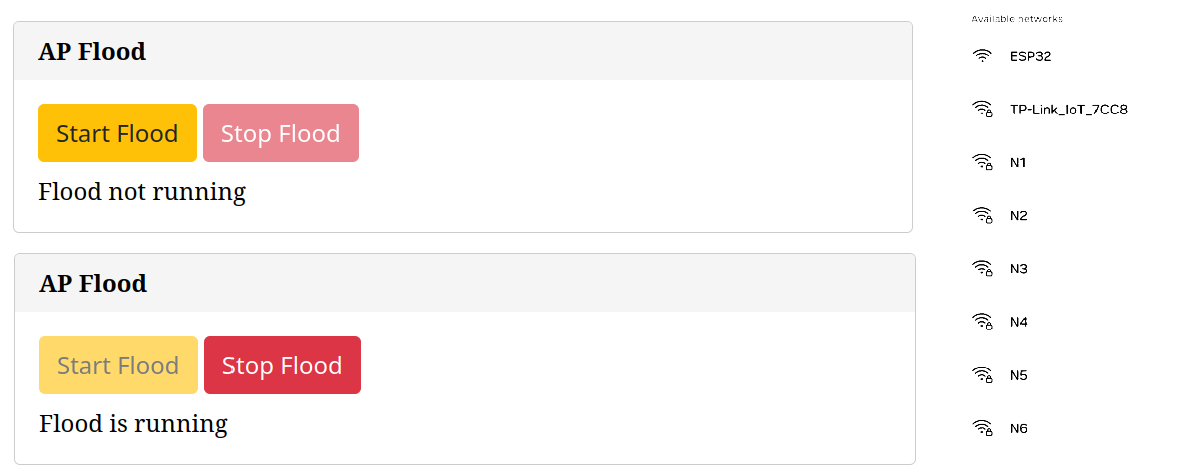


Fig 3.3

### **3.2.5 Scanarea AP-urilor și atacul de tip Deauthentication**

Actor: Utilizator  
Precondiție: Utilizator conectat și a accesat pagina ESP32.  
Descriere: Utilizatorul scanează AP-urile și lansează atacuri de tip Deauth.  
Flux:  
 - utilizatorul selectează opțiunea „Scan AP-uri”.  
 - ESP32 afișează o listă cu AP-urile găsite în apropiere.  
 - utilizatorul selectează un AP țintă din listă.  
 - introduce durata dorită pentru atac.  
 - activează atacul.  
 - ESP32 lansează atacul de tip Deauth împotriva AP-ului țintă pentru perioada selectată.  
 - la final dupa tecerea perioadei de autentificare, utilizatorul este rugat sa se reconecteze la ESP32.  
Rezultat: Dispozitivele conectate la AP-ul țintă sunt deautentificate temporar.

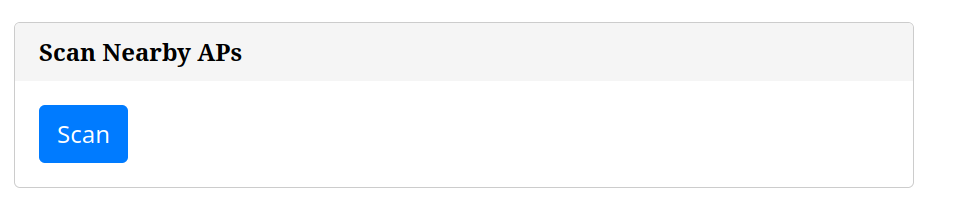


Fig 3.4

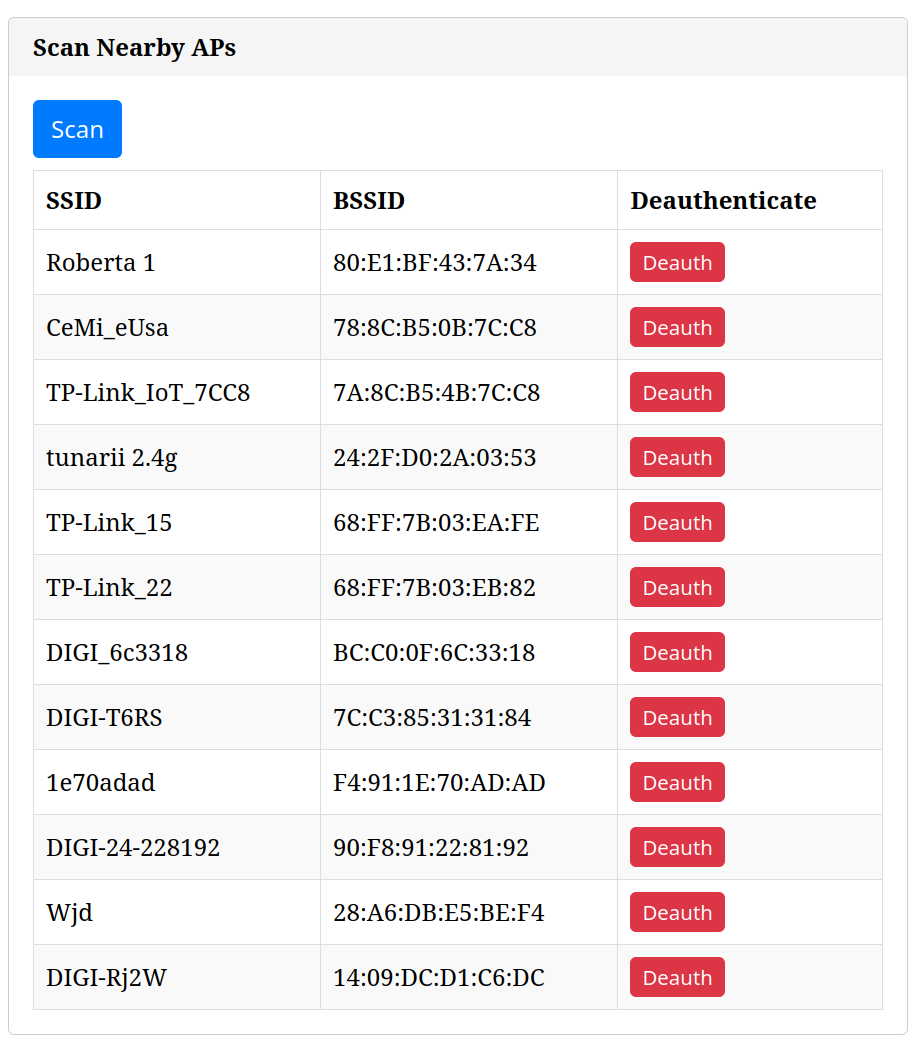


Fig 3.5

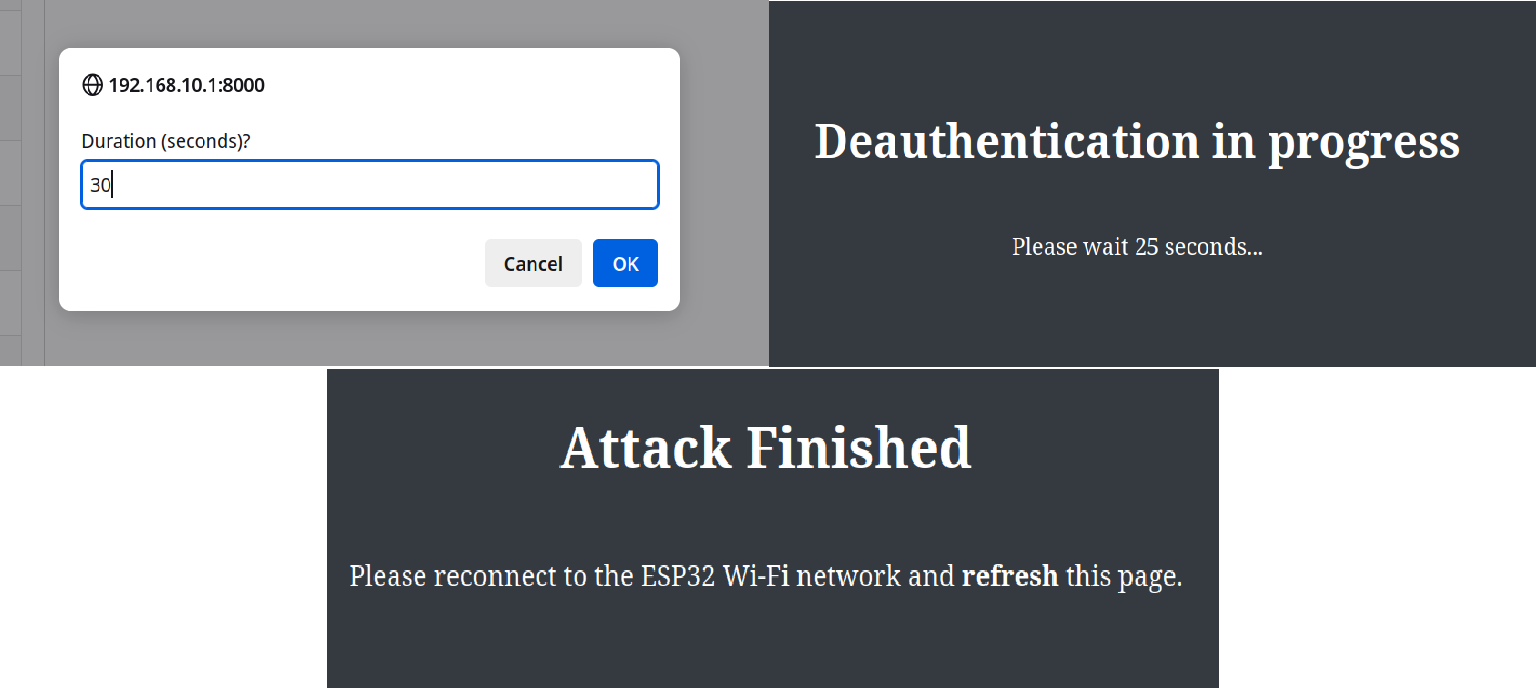


Fig 3.6

### **3.2.6 Afisarea credentialelor**

Actor: Utilizator (intenționat, adică cel care controlează ESP32)  
Precondiție: ESP32 a capturat deja credențiale de la utilizatori neavizați prin pagina falsă de Facebook.  
Descriere: Utilizatorul accesează interfața web a ESP32 și apasă un buton care afișează într-un tabel toate perechile de username și parolă introduse de victime.  
Flux:  
 - utilizatorul accesează pagina principală a aplicației ESP32.  
 - navighează la secțiunea de vizualizare date capturate.  
 - apasă pe butonul „Afișează credențiale”.  
 - ESP32 citește fișierul (sau zona de memorie) unde sunt salvate datele capturate.  
 - interfața web afișează un tabel cu toate username-urile și parolele interceptate.  
Rezultat: Utilizatorul poate vizualiza local datele capturate, fără conexiune la internet.

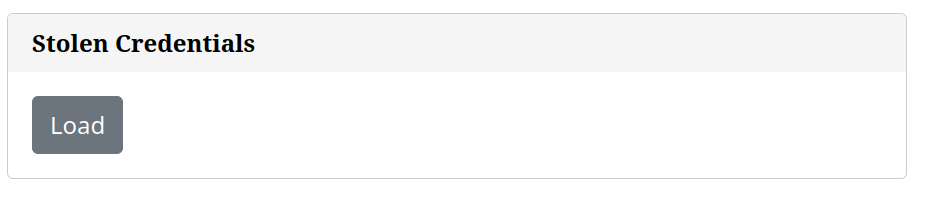


Fig 3.7

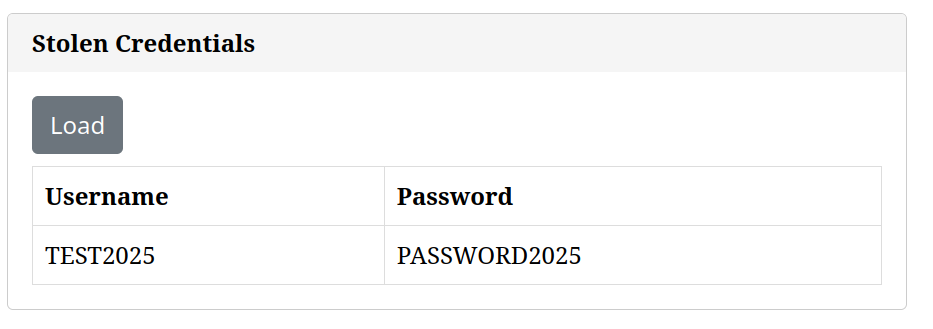


Fig 3.8

### **3.2.7 Victimă conectată la ESP32 și redirecționată prin DNS Hijacking**

Actor: Victimă (Utilizator neavizat)  
Descriere: O victimă se conectează la ESP32 crezând că este o rețea legitimă și accesează un site.  
Flux:  
 - victima caută și identifică rețeaua Wi-Fi falsă creată de ESP32.  
 - victima se conectează la această rețea fără să suspecteze ceva.  
 - victima deschide browserul și încearcă să acceseze un site popular (de exemplu: facebook.com).  
 - prin DNS hijacking, ESP32 redirecționează traficul victimă către o pagină falsă (imitând pagina de autentificare Facebook).  
 - victima introduce datele de autentificare.  
 - ESP32 captează și salvează credențialele introduse de victimă.  
Rezultat: Credințele victimei sunt interceptate și salvate local pe ESP32.

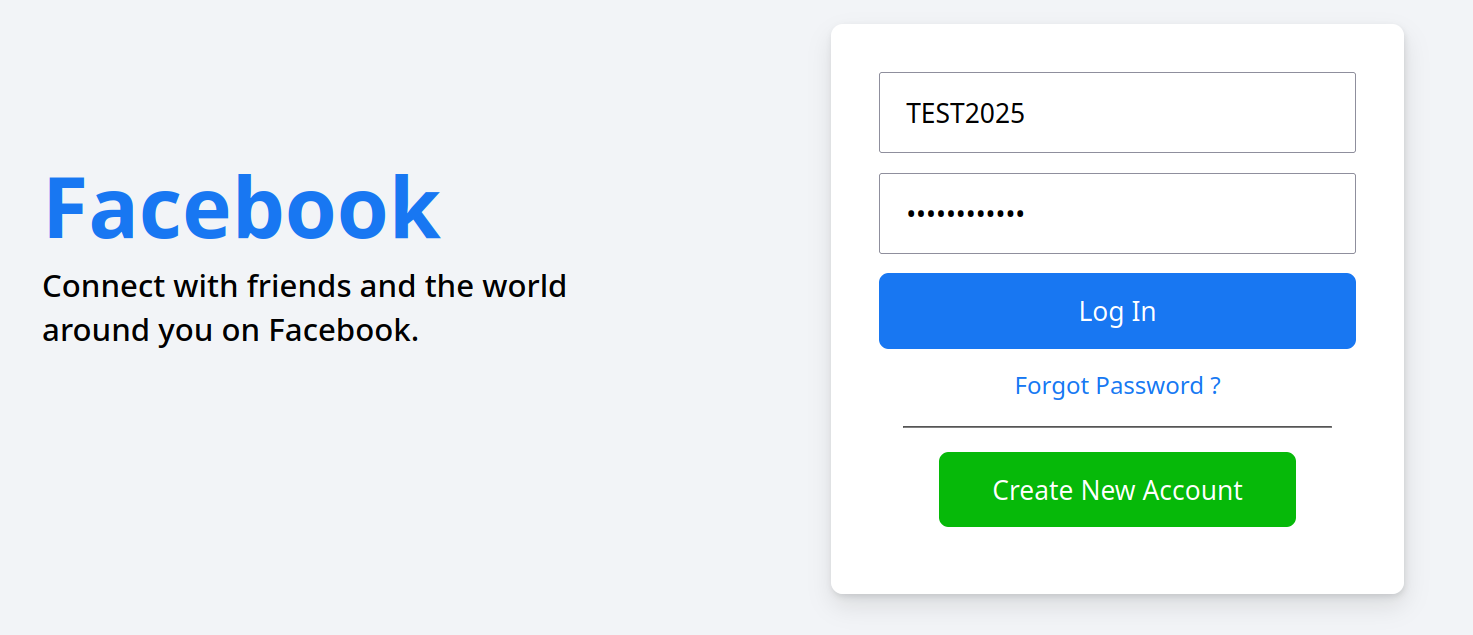


Fig 3.9

# Metode folosite

## **4.1. Cum funcționează AP-urile și cum trimit ele mesaje către dispozitive**

Un punct de acces (Access Point – AP) este componenta care face legătura între rețeaua locală cablată și dispozitivele wireless, permițând comunicația acestora prin unde radio. AP-ul implementează standardul IEEE 802.11, care definește structura fizică și logică a comunicației Wi-Fi, incluzând formatele cadrelor, canalele radio, modurile de autentificare și negocierea criptării [11]. Într-o rețea tipică, AP-ul poate fi un router, dar și un dispozitiv dedicat conectat la un controller de rețea. Comunicarea în rețelele Wi-Fi se face prin cadre (frames) care sunt transmise la nivelul Layer 2 (legătură de date). Aceste cadre sunt împărțite în trei mari categorii: cadre de management, cadre de control și cadre de date. Cadrele de management controlează conexiunile: includ beacon-uri, autentificare, asociere, disasociere și deautentificare. Cadrele de control gestionează transmiterea datelor (ACK, RTS/CTS), iar cele de date conțin informația utilă pentru aplicații.  
 Toate cadrele urmează o structură generală cu antet, adrese MAC, control de durată, secvență, câmpuri opționale și eventual payload. AP-ul emite periodic cadre beacon, transmise broadcast pe canalul curent, prin care își anunță existența și oferă detalii despre rețea. Aceste beacon-uri conțin: SSID (numele rețelei), canalul folosit, metodele de criptare (ex. WPA2), viteze suportate, compatibilități (HT/VHT/HE), și sunt recepționate pasiv de toate dispozitivele aflate în zonă. Când un client dorește conectarea, trimite un probe request sau ascultă beacon-urile, apoi inițiază authentication și association. AP-ul răspunde cu cadre specifice pentru fiecare etapă. Doar după stabilirea asocierii și negocierii criptării poate începe schimbul de cadre de date.  
 Întreg procesul inițial (scanare, autentificare, asociere) are loc în clar, fără criptare, ceea ce îl face vulnerabil la analiză, falsificare sau interferență [12]. De aceea, etapa de conectare este adesea vizată în testele de securitate și în atacurile Wi-Fi, deoarece permite emularea comportamentului unui AP fără parcurgerea autentificării reale [13]. Prin înțelegerea acestor fluxuri de mesaje și rolul cadrelor implicate, pot fi dezvoltate atât aplicații funcționale (routere, sniffer-e), cât și instrumente de testare a securității wireless.

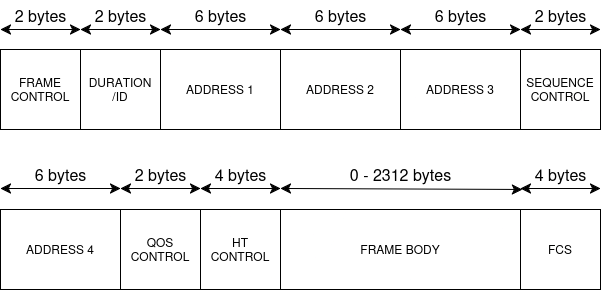


Fig 4.1

## **4.2. Management frames**

În arhitectura rețelelor Wi-Fi, cadrele de management joacă un rol esențial în stabilirea, întreținerea și întreruperea conexiunilor între dispozitivele client și punctele de acces. Aceste cadre sunt definite în standardul IEEE 802.11 și sunt prezente în toate fazele de funcționare ale unei rețele wireless, cu excepția transmiterii efective a datelor [11][12]. Ele conțin informații despre parametrii rețelei, inițiază procesele de autentificare și asociere, permit roaming-ul între AP-uri și gestionează deconectările controlate sau forțate.  
 Structura generală a unui cadru de management include: un câmp Frame Control (care identifică tipul și subtipul), durata transmisiunii, adresele MAC implicate (stație, AP, destinatar), câmpuri de control al secvenței și un payload variabil, compus din Information Elements. Spre deosebire de cadrele de date, cele de management sunt întotdeauna transmise necriptat, fiind interpretate direct de orice dispozitiv care ascultă canalul activ [12].

Printre cele mai importante tipuri se numără:  
 Beacon – transmis de AP pentru a semnaliza existența rețelei;  
 Probe request/response – utilizate pentru căutarea și anunțarea activă a rețelelor;  
 Authentication/Deauthentication – inițiază sau încheie procesul de autentificare;  
 Association/Disassociation – realizează și încheie conexiunea logică MAC.

Un flux tipic implică: clientul trimite un probe request, AP-ul răspunde cu probe response; clientul trimite authentication request, AP-ul îl validează și trimite răspunsul; clientul trimite apoi association request, la care AP-ul răspunde pozitiv dacă acceptă conexiunea. Acești pași sunt obligatorii pentru ca o sesiune de date să poată începe [12].

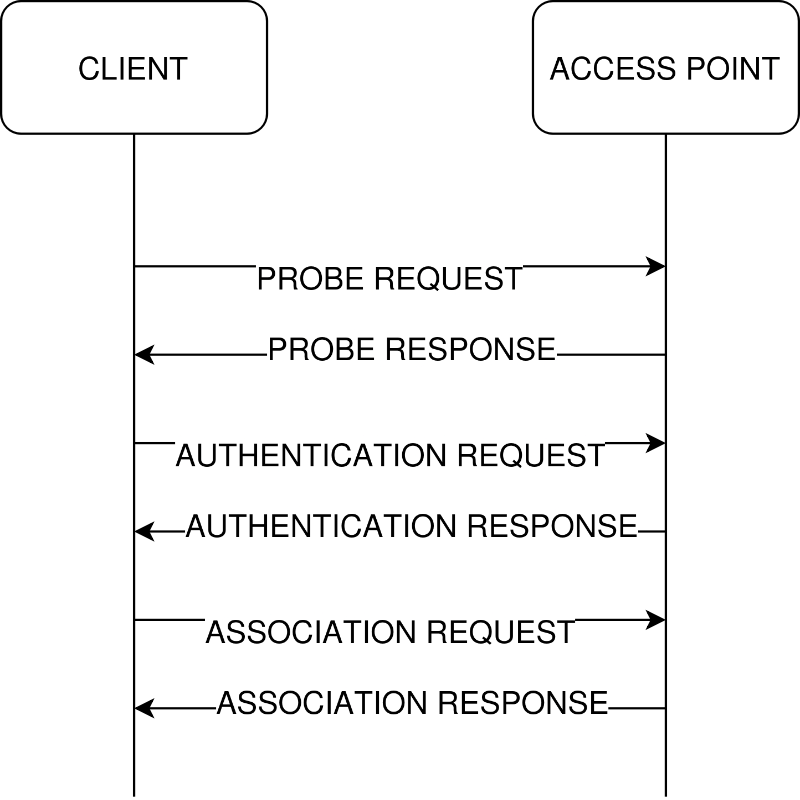


Fig 4.2

Faptul că aceste cadre nu sunt protejate criptografic face posibilă interceptarea, falsificarea sau injectarea lor. Este suficient un atacator în raza rețelei cu un adaptor Wi-Fi capabil de „monitor mode” pentru a trimite cadre deauthentication false și a forța deconectarea dispozitivelor legitime [6][13]. Din acest motiv, aceste cadre sunt vizate în numeroase atacuri wireless (ex: DoS, Evil Twin, Captive Portal). Deși există extensii moderne ale standardului (ex. 802.11w – Protected Management Frames), suportul nu este universal, ceea ce lasă multe rețele și dispozitive vulnerabile [2].

## **4.3. Beacon frames**

Cadrele de tip beacon reprezintă coloana vertebrală a funcționării pasive a unei rețele wireless. Acestea sunt transmise periodic, fără cerere, de către orice punct de acces (AP), pentru a-și anunța prezența și a furniza informații descriptive despre rețea. Transmise la fiecare 100 de milisecunde (sau alt interval configurat), beacon-urile permit dispozitivelor client să afle automat că există o rețea în zonă și să obțină toate detaliile necesare conectării [11][12].  
 Structura unui beacon frame include: antetul cadrului (Frame Control, durată, adrese), un timestamp pentru sincronizarea dispozitivelor, intervalul beacon, BSSID-ul (MAC-ul AP-ului), și o colecție de Information Elements (IEs). Aceste IEs sunt blocuri TLV (type-length-value) care conțin:  
 - SSID-ul rețelei (nume),  
 - tipul și nivelul criptării (WEP/WPA2/WPA3),  
 - vitezele suportate,  
 - canalul radio activ,  
 - extensii pentru roaming sau mesh,  
 - capabilități de rețea (802.11n/ac/ax).

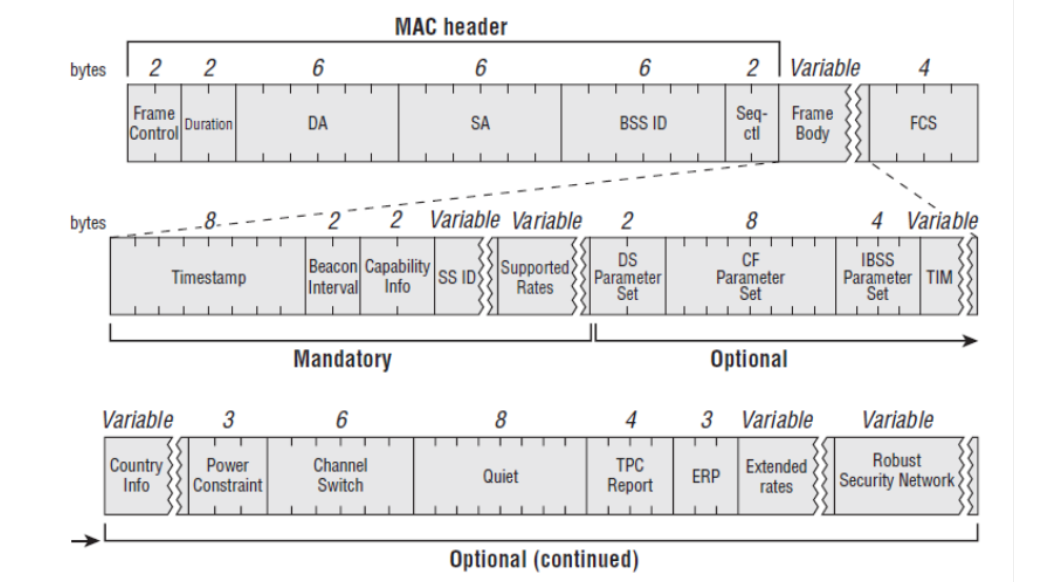


Fig 4.3

Beacon-urile sunt transmise broadcast pe canalul activ, ceea ce înseamnă că toate dispozitivele aflate în raza AP-ului pot recepționa aceste mesaje. Clienții le utilizează pentru a popula lista de rețele vizibile, pentru roaming automat între AP-uri, pentru sincronizarea ceasurilor interne și pentru alegerea celei mai bune rețele în funcție de semnal și suport [12]. Deoarece beacon-urile sunt transmise în clar, fără criptare sau semnătură, acestea pot fi ușor imitate [13]. Un atacator poate crea beacon-uri false cu SSID-uri identice sau aleatorii, păcălind dispozitivele să încerce conectarea la o rețea falsă sau saturând lista de rețele afișate. Această metodă este baza atacurilor de tip AP Flood și Evil Twin, în care se creează zeci sau sute de rețele false pentru a induce confuzie sau pentru a captura date de autentificare [14].

## **4.4. AP Flood prin beacon-uri**

AP Flood-ul este o tehnică de perturbare a funcționării rețelelor wireless care folosește cadre beacon false pentru a simula existența mai multor rețele fictive într-o zonă. Acest atac este posibil din cauza comportamentului dispozitivelor client: în mod normal, acestea recepționează toate cadrele beacon primite pasiv și le interpretează ca rețele valide. Dacă un atacator generează sute sau mii de astfel de beacon-uri, fiecare cu un SSID diferit, dispozitivul va adăuga toate aceste rețele în listă, fără o modalitate de validare a autenticității [14].  
 Dispozitive precum ESP32 permit programarea în așa fel încât să trimită beacon-uri într-o buclă, fiecare având un BSSID (MAC) diferit și un SSID unic sau mimat după rețele reale [10][15]. Aceste beacon-uri sunt transmise broadcast pe canalul dorit, afectând toate dispozitivele aflate în raza de acțiune. Un telefon sau laptop va afișa toate aceste rețele ca fiind disponibile, ceea ce încetinește lista de afișare, poate cauza blocaje în interfața grafică și face dificilă conectarea la rețele reale. Atacul este și mai eficient dacă sunt folosite SSID-uri frecvente sau memorate de dispozitivul victimă. În astfel de cazuri, clientul poate încerca automat conectarea (auto-connect), ceea ce duce la un consum suplimentar de resurse, eșecuri de conectare și întreruperi în rețelele reale. În unele cazuri, firmware-ul dispozitivului poate intra în bucle de scanare sau poate reseta interfața Wi-Fi [15].  
 Beacon-urile false pot fi transmise și pe mai multe canale, crescând raza de acțiune a atacului. Deoarece standardul 802.11 nu include semnături sau criptare pentru beacon-uri, dispozitivele client nu au un mecanism nativ pentru a respinge beacon-urile falsificate [11][12]. Această lipsă de verificare le face vulnerabile chiar și în rețele moderne, în special dacă nu sunt folosite sisteme suplimentare de filtrare sau detecție a anomaliilor [14].

### **4.4.1 Realizarea in practica**

Atacul de tip AP Flood este implementat în proiectul prezent cu ajutorul funcționalităților oferite de platforma ESP32, care permite transmiterea manuală a pachetelor beacon modificate. Codul este structurat astfel încât să creeze un flux continuu de cadre beacon, fiecare reprezentând o rețea wireless falsă, cu SSID și BSSID unic. Scopul este de a simula prezența a zeci sau sute de puncte de acces într-o zonă restrânsă, perturbând funcționarea dispozitivelor Wi-Fi legitime din apropiere.

### **4.4.2 Structura funcțională a atacului în proiect**

Atacul este inițiat de către utilizator printr-un endpoint HTTP specific (ex: /flood) expus de serverul web al ESP32. Serverul web este conceput pentru a primi cereri și a porni diverse acțiuni, inclusiv această funcționalitate de tip DoS. Când cererea către ruta /flood este recepționată, aplicația pornește un task dedicat pe un nucleu al microcontrolerului, care se ocupă exclusiv de generarea beacon-urilor false.

Pentru fiecare beacon generat, sunt completate următoarele câmpuri în pachetul beacon:

**Frame Control** (0x80, 0x00): indică faptul că este un management frame de tip beacon.

**Duration** (0x00, 0x00): câmp standard, fără importanță majoră pentru acest atac.

**Destination address**: adresa MAC de broadcast (FF:FF:FF:FF:FF:FF), indicând că pachetul este destinat tuturor stațiilor.

**Source address & BSSID**: ambele setate identic și variate la fiecare beacon pentru a simula rețele unice. Ultimul octet este modificat în funcție de indexul beaconului.

**Sequence/Fragment number**: incrementat pentru fiecare SSID, permițând identificarea de cadre unice.

**Timestamp**: lăsat 0, va fi suprascris de hardware.

**Beacon interval**: 0x0064 (100ms standard).

**Capability info**: 0x0431 - stabilește capabilitățile AP-ului fals (suport standard).

**SSID**: este completat dinamic, luat dintr-o listă de șiruri de caractere predefinite. Lungimea și conținutul sunt scrise în beacon manual pentru fiecare pachet.

**Supported rates**: câmp fix care indică ratele de transmisie suportate (ex: 1 Mbps până la 36 Mbps).

**DS Parameter set**: indică canalul curent de transmisie (canalul 1 în implementare standard).

**TIM (Traffic Indication Map)**: informații despre buffer multicast pentru clienți în mod power-saving, completate cu valori standard.

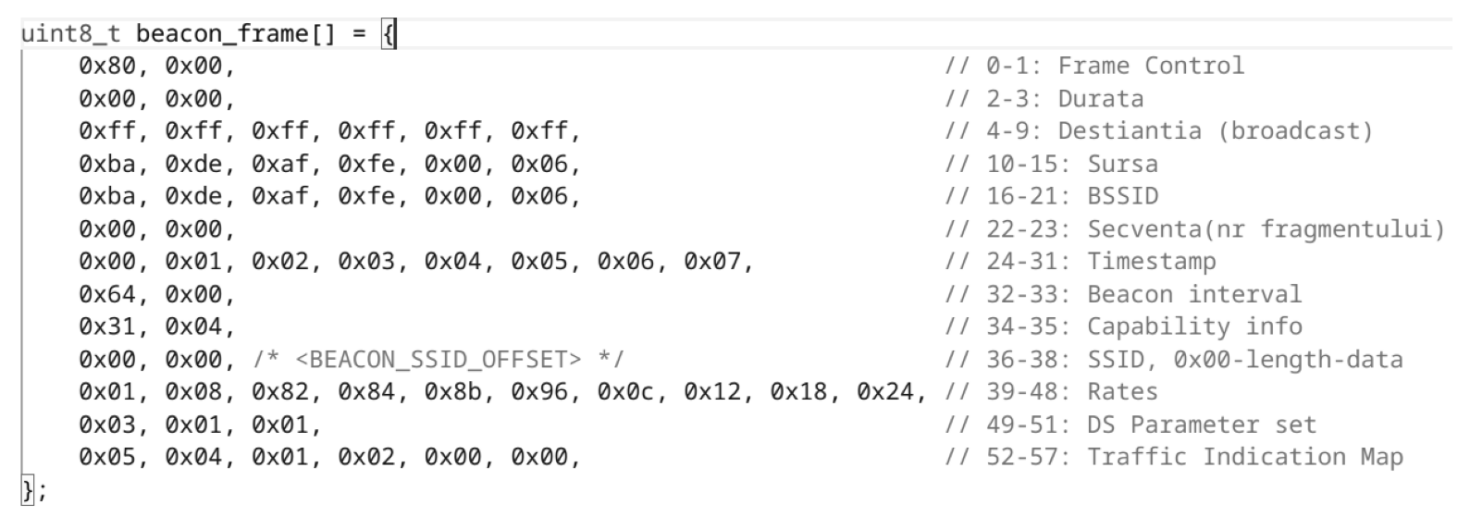


Fig 4.4

### **4.4.3 Mecanismul de execuție**

Task-ul de flood rulează continuu, parcurgând lista de SSID-uri definite. Pentru fiecare, construiește un pachet beacon nou, înlocuind lungimea SSID-ului și conținutul propriu-zis în cadrul pachetului. Se modifică și adresele MAC (ultimul octet), și se calculează secvența corespunzătoare. Pachetul este apoi transmis folosind funcția esp\_wifi\_80211\_tx, care permite injectarea de cadre brute în rețea.  
Rata de transmitere este calculată în funcție de numărul total de SSID-uri și de un delay de 100 ms împărțit la acest număr, astfel încât toate beacon-urile să fie transmise ciclic și constant.

**Impactul asupra rețelelor și clienților**

Dispozitivele client vor interpreta fiecare beacon ca o rețea validă și o vor adăuga în lista de rețele disponibile. Această simulare masivă poate duce la:

- Saturația interfeței grafice de conectare (ex: Android, Windows);  
 - Întârzierea sau blocarea conectării la rețele reale;  
 - Încercări automate de conectare la rețele false dacă numele este memorat;  
 - Scăderea performanței Wi-Fi în aplicații legitime;  
 - Blocaje temporare în driverul rețelei sau scanare repetată eșuată.

### **4.4.4 Observații asupra limitelor**

Deși foarte eficient vizual și funcțional în zone restrânse, atacul are câteva limitări:

- Nu poate forța conectarea clienților la rețele false fără o combinație cu alte atacuri (ex. deauth);  
 - Este limitat la puterea de transmisie a ESP32 și canalul pe care operează;  
 - Poate fi detectat de sisteme de monitorizare a traficului wireless.

## **4.5. Deauthentication frames**

Cadrele de tip Deauthentication sunt cadre de management definite în standardul IEEE 802.11 și au rolul de a încheia oficial procesul de autentificare dintre un client și un punct de acces (AP) [11][12]. Ele sunt parte din mecanismul de control al conexiunilor într-o rețea Wi-Fi și pot fi transmise de ambele părți: atât de client, cât și de AP. Când un astfel de cadru este recepționat, conexiunea este considerată încheiată și orice comunicare ulterioară necesită reluarea autentificării și asocierii.  
Structura unui cadru de deauthentication este simplă. El include:  
 - Antetul de control (Frame Control) care specifică tipul „Management” și subtipul „Deauthentication”,  
 - Trei adrese MAC: sursa (expeditorul), destinația (receptorul) și BSSID (rețeaua),  
 - Câmpul de „Reason Code”, care explică motivul deautentificării (ex: cod 3 – „deautentificat de AP”).

Deauthentication-ul este distinct de disassociation: primul încheie autentificarea, în timp ce al doilea încheie asocierea. Cele două pot apărea împreună, dar standardul le separă logic și funcțional. Spre exemplu, dacă un client se mișcă într-o zonă fără semnal, AP-ul poate trimite un cadru de deauthentication pentru a încheia corect sesiunea, prevenind posibile conflicte de stare.  
 Toate cadrele de tip deauthentication sunt transmise în clar, fără criptare sau protecție criptografică în standardul original [11]. Din acest motiv, ele sunt extrem de vulnerabile la falsificare. Un atacator aflat în apropierea rețelei poate genera manual cadre de acest tip folosind un adaptor Wi-Fi compatibil și software specializat (ex: aireplay-ng) [3]. Dispozitivele client, dacă recepționează aceste cadre false, consideră în mod legitim că au fost deautentificate și încheie conexiunea.  
 Deauthentication-ul este utilizat și în mod legitim, spre exemplu când un AP dorește să elibereze un client inactiv sau când un client se reconectează manual. Totuși, lipsa oricărei forme de autentificare a sursei acestor cadre face ca ele să fie un vector de atac des întâlnit [13]. Standardul IEEE 802.11w încearcă să adreseze această problemă prin introducerea „Protected Management Frames”, dar suportul este limitat [2].  
În mod obișnuit, cadrele deauthentication sunt trimise direct (unicast) către un client specific, dar pot fi trimise și broadcast pentru a afecta toți clienții simultan. Majoritatea dispozitivelor moderne ignoră deauthentication broadcast din cauza abuzurilor frecvente, însă rămân vulnerabile la atacurile direcționate.

## **4.6. Atacurile de deautentificare**

Atacurile de tip Deauthentication se bazează pe trimiterea unor cadre false de deautentificare (deauthentication frames) către dispozitivele client dintr-o rețea Wi-Fi. Aceste cadre sunt o parte legitimă a protocolului IEEE 802.11 și sunt folosite în mod normal pentru a încheia conexiunile dintre client și punctul de acces (AP) [11][12]. Cu toate acestea, lipsa oricărei forme de autentificare a acestor cadre în standardul de bază le face extrem de vulnerabile la spoofing și abuz [13].  
 Într-un atac clasic, atacatorul transmite în mod repetat cadre de deauthentication către un client sau către toți clienții conectați la un AP. Dispozitivele recepționează aceste cadre și consideră că deconectarea este legitimă. Drept urmare, ele încheie conexiunea și, în funcție de sistemul de operare și configurare, pot încerca automat reconectarea. Dacă atacatorul repetă trimiterea acestor cadre la intervale scurte, clientul nu mai reușește să se reconecteze, fiind prins într-un ciclu de deconectare continuă [13].  
Există două variante principale ale atacului:  
 **Broadcast Deauthentication** – cadrele sunt trimise către toți clienții conectați la un AP. Este ușor de executat, dar multe dispozitive moderne ignoră cadrele broadcast nesemnate.  
 **Unicast Deauthentication** – cadrele sunt trimise țintit către adresa MAC a unui client. Este mai eficient, deoarece dispozitivul client consideră sursa ca fiind autentică.

Un atacator are nevoie de:

- un adaptor Wi-Fi compatibil cu modul „monitor” și capabil de injecție de pachete,  
 - software specializat (ex. aireplay-ng, mdk3, Scapy),  
 - identificarea adresei MAC a clientului și BSSID-ul AP-ului (prin scanare cu airodump-ng).

Un exemplu tipic: atacatorul identifică un client activ, trimite cadre de deauthentication, clientul se deconectează și în timpul reconectării, atacatorul poate captura „4-way handshake”-ul pentru a încerca spargerea parolei WPA [1].  
Pentru a combate aceste atacuri, standardul 802.11w introduce Protected Management Frames (PMF), care criptează cadrele de management (inclusiv deauthentication) [2]. Totuși, multe dispozitive nu suportă complet acest standard, iar altele îl dezactivează din motive de compatibilitate. În absența PMF, rețelele rămân vulnerabile la atacuri de deautentificare, mai ales în medii publice sau nesecurizate [2].

### **4.6.1 Realizarea in practica**

În cadrul proiectului atacul este structurat pe mai multe faze distincte, reflectând o abordare sistematică ce imită comportamentul natural al rețelelor Wi-Fi.

#### **Faza 1: Scanarea rețelelor**

Procesul începe prin scanarea activă a mediului wireless pentru identificarea rețelelor disponibile. Această etapă este realizată de ESP32 cu ajutorul funcțiilor oferite de platforma ESP-IDF, în special esp\_wifi\_scan\_start() și esp\_wifi\_scan\_get\_ap\_records(). În esență, dispozitivul ESP32 trimite cereri de scanare (probe request), iar punctele de acces din apropiere răspund cu cadre de tip beacon și probe response, conținând informații esențiale precum SSID, BSSID (adresa MAC a AP-ului), canalul utilizat și nivelul semnalului.  
 Aceste date sunt colectate de către microcontroler și prezentate utilizatorului printr-o interfață web sau altă metodă de selecție, permițând alegerea rețelei țintă asupra căreia se va efectua atacul. Utilizatorul alege un AP, iar informațiile despre acesta, precum SSID-ul, BSSID-ul și canalul, sunt transmise către serverul ESP32 pentru pregătirea atacului.

#### **Faza 2: Configurarea pentru atac**

După primirea informațiilor despre rețeaua țintă, ESP32 se configurează astfel încât să se prezinte ca fiind acel AP. Se setează BSSID-ul (adresa MAC) și SSID-ul propriu pentru a coincide cu cele ale AP-ului selectat. Acest lucru are două efecte importante:

- ESP32 va putea intercepta pachetele trimise broadcast către AP-ul real (fiind pe același canal și cu aceeași identitate);  
 - Clienții care comunică cu AP-ul real pot trimite accidental pachete și către ESP32, mai ales în cazul în care semnalul ESP32 este mai puternic.

**Faza 3: Transmiterea cadrelor de deautentificare**

Atunci când ESP32 detectează trafic din partea clientului către AP-ul real – sau chiar în mod direct și proactiv – acesta transmite cadre de tip Deauthentication către client. Aceste cadre sunt standardizate în IEEE 802.11 și au rolul de a notifica clientul că trebuie să întrerupă conexiunea.

Un cadru de deautentificare include:

**Frame Control**: setat pentru a indica un management frame de tip deauth;  
 **Duration**: câmp uzual setat la 0;  
 **Destination Address** (DA): adresa MAC a clientului;  
 **Source Address** (SA): adresa MAC a AP-ului (fals, în cazul nostru);  
 **BSSID**: adresa MAC a AP-ului;  
 **Reason Code**: motivul de deautentificare, de obicei 0x0001 (stația a fost deautentificată de AP).

ESP32 folosește funcția de transmisie de cadre brute esp\_wifi\_80211\_tx pentru a injecta aceste pachete în rețea. Întrucât standardul Wi-Fi nu prevede un mecanism de autentificare a cadrelor de deautentificare, clientul nu poate valida dacă acestea provin de la AP-ul real. Prin urmare, clientul întrerupe conexiunea în mod imediat.

### **4.6.2 Efectul atacului**

După transmiterea cadrului de deautentificare, clientul va fi forțat să se deconecteze de la rețeaua Wi-Fi, iar în funcție de comportamentul sistemului de operare sau de nivelul semnalului, poate:  
 - Să încerce reconectarea automată (declanșând un nou atac);  
 - Să se conecteze la un AP fals (dacă este disponibil un Evil Twin);  
 - Să intre într-un ciclu de conectare-deconectare, afectând grav utilizabilitatea rețelei.

**4.7. Ce este DNS**

DNS (Domain Name System) este un sistem esențial pentru funcționarea internetului, responsabil pentru traducerea numelor de domenii ușor de înțeles de oameni (ex: [www.example.com](http://www.example.com" \t "/home/robert/Documents\\x/_new)) în adrese IP (ex: 93.184.216.34), pe care calculatoarele le pot utiliza pentru a comunica între ele [16]. Sistemul DNS funcționează ca o agendă telefonică distribuită, oferind rapid adresa unui server atunci când un utilizator introduce un URL într-un browser.  
 Organizarea acestuia este ierarhică. În vârf se află serverele root, care știu unde se află serverele pentru fiecare top-level domain (TLD), precum .com, .org, .ro. Sub acestea urmează serverele TLD care direcționează către serverele autoritare pentru fiecare domeniu (ex: example.com). Aceste servere autoritare cunosc adresele exacte ale subdomeniilor (ex: www, mail, ftp). Când un utilizator accesează un site, dispozitivul său trimite o interogare DNS către un server recursiv (de obicei, serverul DNS al providerului de internet sau unul public ca Google 8.8.8.8). Acesta caută în cache, iar dacă nu are răspunsul, inițiază o serie de interogări recursive pentru a obține IP-ul corect. Răspunsul este apoi returnat dispozitivului, care inițiază conexiunea către acel IP [16].  
Sistemul DNS este rapid, flexibil și extensibil, dar prezintă și puncte slabe: interogările sunt de obicei transmise în text clar (UDP pe portul 53), fără criptare, ceea ce permite interceptarea, modificarea sau manipularea lor de către atacatori [13]. Acest lucru a dus la dezvoltarea unor extensii precum DNSSEC, DoT (DNS over TLS) și DoH (DNS over HTTPS), care oferă integritate și confidențialitate. În lipsa acestor protecții, un atacator poziționat între client și serverul DNS poate modifica răspunsurile, direcționând utilizatorul către servere false [13].

## **4.8. Mesajele DNS**

Protocolul DNS definește două tipuri principale de mesaje: request (interogare) și response (răspuns). Aceste mesaje sunt transmise de obicei prin protocolul UDP, pe portul 53, deși în unele cazuri (zone transfer, DNSSEC) se folosește TCP. Formatul mesajului DNS este standardizat și are o structură bine definită care permite funcționarea ierarhică și rapidă a sistemului [16].

Un mesaj DNS conține următoarele secțiuni:

Header – 12 octeți care includ ID-ul unic al cererii, tipul mesajului (query/response), coduri de eroare, câmpuri de control (ex: dacă este răspuns autoritar, dacă se permite recursivitate).  
 Question – secțiunea care conține numele de domeniu solicitat și tipul de înregistrare (de regulă A pentru IPv4, AAAA pentru IPv6, dar și MX, CNAME, NS etc.).  
 Answer – secțiunea răspuns, care conține datele returnate (ex: IP-ul asociat cu domeniul).  
 Authority – indică serverul autoritar pentru domeniul interogat.  
 Additional – poate include informații adiționale utile (ex: alte IP-uri, TTL, recorduri de tip OPT pentru EDNS) [16].

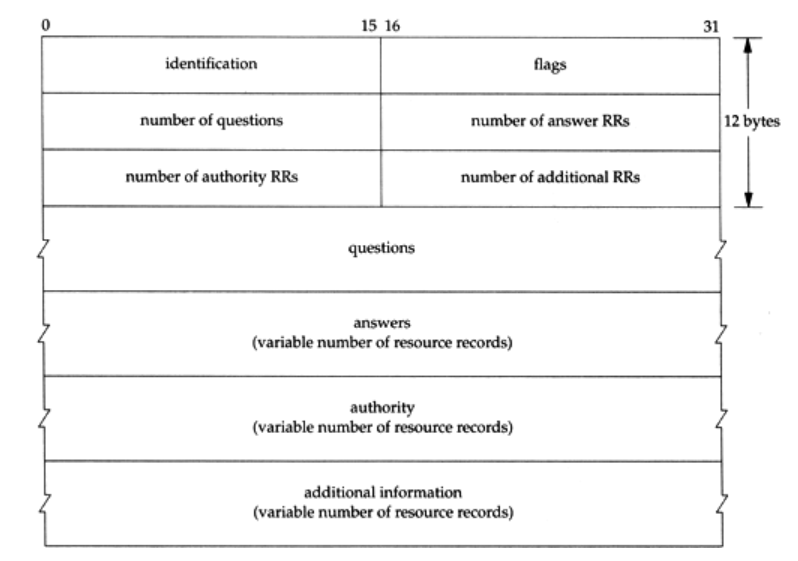


Fig 4.5

Într-o interogare simplă, un client cere „Care este IP-ul pentru www.example.com?”. Serverul DNS răspunde cu un mesaj de tip response în care secțiunea „Answer” conține înregistrarea de tip A: „93.184.216.34”. Dacă serverul nu cunoaște adresa, poate returna un răspuns „non-authoritative” sau poate redirecționa interogarea mai departe [16].

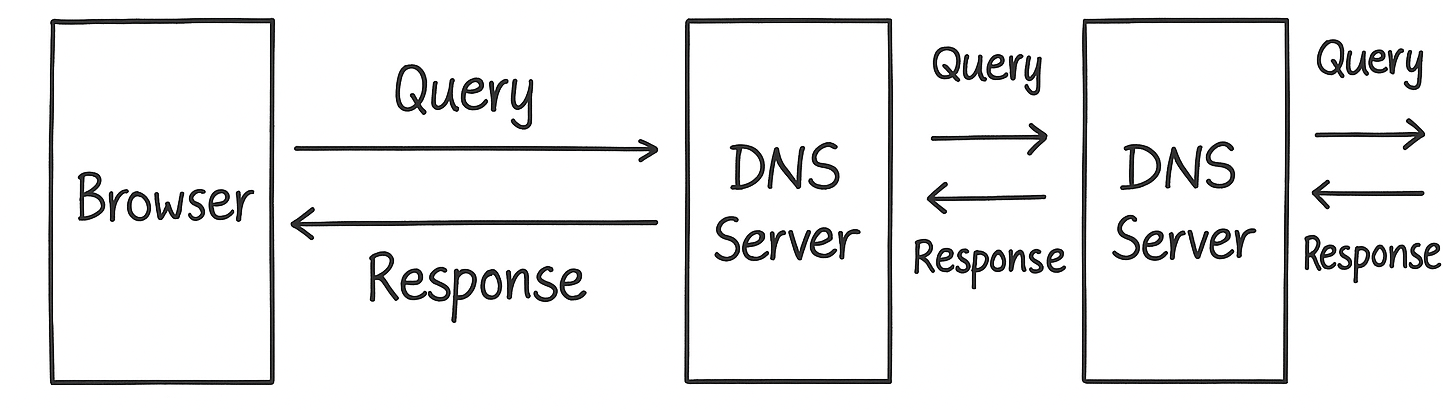


Fig 4.6

Lipsa de criptare a acestor mesaje le face vulnerabile la atacuri de tip spoofing și hijacking. Un atacator poate forja un răspuns fals către un client înainte ca serverul DNS legitim să răspundă, păcălindu-l astfel că IP-ul corect pentru un domeniu este cel al unui server malițios [13]. Această tehnică stă la baza atacului de tip DNS Hijacking. Tehnologiile moderne încearcă să contracareze aceste riscuri prin validare criptografică (DNSSEC) sau criptare completă a transportului (DoT, DoH), dar implementarea nu este universală [13].

## **4.9. DNS Hijacking**

DNS Hijacking este o tehnică prin care un atacator interceptează sau falsifică răspunsurile DNS ale unui utilizator, direcționându-l către adrese IP diferite de cele reale. Atacul exploatează în special lipsa criptării și verificării autenticității mesajelor DNS în scenariile clasice [13][16]. Un dispozitiv compromis sau un atacator aflat în aceeași rețea poate intercepta cererea DNS și trimite un răspuns modificat mai rapid decât serverul legitim, convingând clientul că adresa IP a site-ului solicitat este alta [16].  
Atacul poate avea loc în mai multe moduri:  
 Client-side hijacking – malware modifică configurația DNS a dispozitivului;  
 Router hijacking – atacatorul schimbă setările DNS pe routerul victimei;  
 Man-in-the-Middle – atacatorul aflat în aceeași rețea (ex: într-un Wi-Fi public) interceptează și răspunde la cereri DNS;  
 Evil AP DNS injection – un AP fals configurează prin DHCP un server DNS controlat de atacator.  
 În cazul proiectului bazat pe ESP32, hijacking-ul este realizat local, fără a accesa o rețea externă. ESP32 acționează simultan ca AP și server DNS. Prin DHCP, dispozitivele care se conectează primesc ca DNS server chiar adresa ESP32-ului. Ulterior, când utilizatorul accesează un domeniu (ex: „facebook.com”), cererea este interceptată de ESP32, care returnează propriul IP în locul adresei reale. Browserul utilizatorului deschide automat o pagină web găzduită local pe ESP32 – o clonă a paginii Facebook. Utilizatorul introduce datele, care sunt stocate de atacator [15].  
 Această tehnică este extrem de eficientă în rețele neprotejate și este dificil de detectat de către utilizatori obișnuiți, întrucât domeniul din bara de adresă rămâne aparent corect, iar conexiunea nu este securizată (HTTP în loc de HTTPS). Protejarea împotriva acestui tip de atac necesită validarea criptografică a răspunsurilor DNS (DNSSEC), criptarea traficului DNS (DoH/DoT) și evitarea conectării la rețele necunoscute [13].

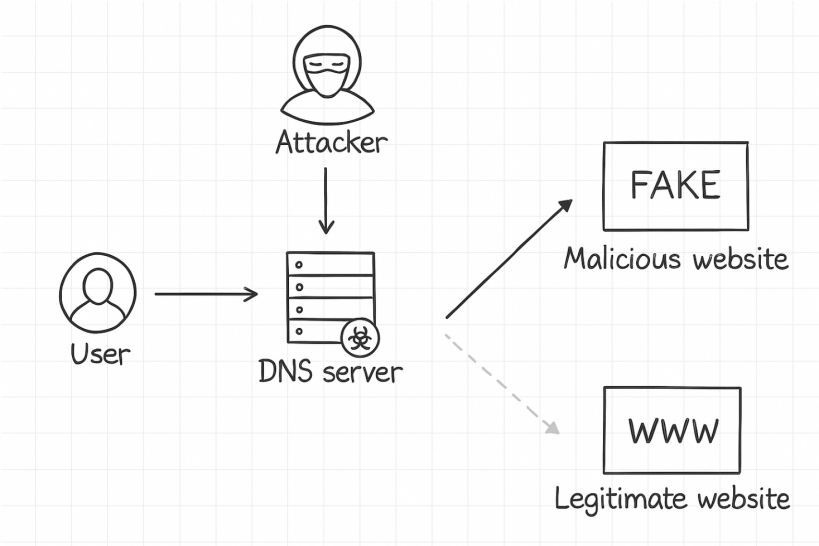
****

Fig 4.7

### **4.9.1 Realizarea in practica**

Atacul de tip DNS Hijacking implementat în proiectul de față are scopul de a redirecționa cererile DNS ale utilizatorului către ESP32, indiferent de domeniul cerut, pentru a servi o pagină falsă. Acest tip de atac este posibil deoarece DNS-ul funcționează pe baza unui protocol nesecurizat (UDP), iar clienții nu validează autenticitatea răspunsurilor DNS. Implementarea este minimalistă, dar eficientă, și constă într-un server DNS scris manual, care rulează pe ESP32 și răspunde în mod intenționat greșit la anumite cereri DNS.

#### **Inițializarea serverului DNS**

Serverul DNS rulează ca un task separat, utilizând un socket UDP legat la portul standard DNS (53). La inițializare, se creează un socket și se face binding la adresa locală INADDR\_ANY, ceea ce înseamnă că serverul va asculta pe toate interfețele disponibile pentru cereri DNS.  
Dacă bindingul este reușit, serverul intră într-o buclă infinită în care ascultă cereri folosind recvfrom(). Pachetele primite sunt analizate și interpretate ca interogări DNS.

**Parsarea numelui de domeniu**

Pentru a interpreta interogarea DNS, funcția get\_query\_name() este folosită pentru a extrage numele domeniului din secțiunea de întrebări a pachetului. Domeniul este extras folosind convenția DNS de tip etichetă: lungimea fiecărui segment urmată de caracterele segmentului. De exemplu, facebook.com apare în pachet ca 0x08 'facebook' 0x03 'com' 0x00. Această funcție reconstruiește domeniul complet și îl salvează într-un buffer pentru a putea fi comparat mai târziu.

**Tratarea cererii DNS**

După ce se obține domeniul, serverul verifică dacă acesta este ținta (de exemplu, facebook.com sau www.facebook.com). Dacă da, construiește un răspuns DNS valid, direcționând domeniul către adresa IP a ESP32 (ex. 192.168.10.1).

- Se setează bitul QR (Query Response) pentru a marca pachetul ca răspuns.  
 - Se completează câmpul ANCOUNT cu valoarea 1 (există un singur răspuns).  
 - Se adaugă în secțiunea de răspuns un record de tip A (adresa IPv4), cu un TTL de 120 secunde și adresa ESP32 codificată pe 4 octeți.

Clientul care a făcut cererea va interpreta acest răspuns ca valid și va încerca să se conecteze la adresa IP furnizată, adică la ESP32, în loc de serverul real.

**Comportament pentru domenii non-țintă**

Dacă domeniul cerut nu este unul din cele vizate, serverul construiește un răspuns NXDOMAIN. În acest caz:

- Bitul QR este setat, iar codul de eroare (RCODE) este 3, indicând că domeniul nu există.  
 - Numărul de răspunsuri este 0 (ANCOUNT = 0).

Această strategie face ca atacul să fie mai ușor de detectat, deoarece doar anumite domenii sunt redirecționate, iar restul generează erori sau par inexistente. Într-o versiune viitoare, microcontrollerul ar putea transmite cererile DNS care nu vizează domeniile-țintă către un server DNS real, ceea ce ar spori realismul și ar simula un comportament autentic de rețea.

## **4.10 Arhitectura aplicației server-side pe ESP32**

Pentru a susține funcționalitățile de scanare, atac și captare de date demonstrativă, aplicația rulează trei servere independente în paralel pe dispozitivul ESP32.

Aceste servere sunt:   
 - un server web principal care expune o interfațăsimplă prin HTTP pe portul 8000, permițând utilizatorului să scaneze rețele, să activeze atacuri de tip deauthentication și flood.  
 - un server dedicat pentru pagina falsă de Facebook, expus pe portul 80, folosit în contextul unui atac DNS hijacking pentru a capta credențialele introduse de utilizator.  
 - un server DNS pe portul standard 53, care răspunde la interogările DNS și direcționează utilizatorii către ESP32 atunci când aceștia solicită domenii vizate (ex: facebook.com).

Toate cele trei servere sunt active simultan și rulează concurent folosind mecanismele asincrone ale sistemului de operare încorporat pe ESP32 (ex: FreeRTOS, task-uri separate).

### **4.10.1** **Serverul Web Principal**

Acest server gestionează comenzile principale transmise de utilizator printr-o interfață simplificată. Este punctul central de interacțiune și oferă acces la toate funcțiile experimentale.

Rutele expuse sunt:

#### 

Fig 4.8

#### **Metoda “GET” la ruta “/”** Returnează pagina principală HTML (interfața de control).

#### **Metoda “GET” la ruta “/scan”**

#### Inițiază un proces de scanare a rețelelor Wi-Fi din proximitate. Răspunsuri posibile:

#### 200 OK – returnează un JSON cu lista rețelelor descoperite:

#### [ { "ssid": "Retea1", "bssid": "AA:BB:CC:DD:EE:01" }, { "ssid": "Retea2", "bssid": "11:22:33:44:55:66" } ]

#### 500 Internal Server Error – eroare în procesul de scanare (ex: Wi-Fi deja ocupat).

#### **Metoda “GET” la ruta “/attack” cu parametrii BSSID si TIMEOUT** Inițiază un atac de deautentificare împotriva rețelei specificate. Parametrii: bssid: adresa MAC a AP-ului țintă. timeout: durata atacului în secunde. Răspunsuri: 200 OK – atacul a fost pornit cu succes. 400 Bad Request – lipsesc parametrii obligatorii. 500 Internal Server Error – eroare internă la lansarea atacului (ex: modul Wi-Fi instabil). Efect: ESP32 imită AP-ul țintă și transmite cadre de deauthentication; în timpul atacului, ESP32 devine indisponibil pentru conexiuni.

#### **Metoda “GET” la “/flood”** Pornește atacul de tip AP Flood (transmiterea de cadre beacon false). Răspunsuri: 200 OK – flood-ul a fost pornit. 500 Internal Server Error – imposibil de inițiat flood-ul.

#### **Metoda “GET” la “/flood/stop”** Oprește atacul flood-ul dacă este activ. Răspunsuri: 200 OK – atacul a fost oprit cu succes. 409 Conflict – atacul nu era activ (nu există instanță de flood pornită).

### **4.10.2 Serverul Facebook Fals**

Acest server este activat automat și rulează pe portul standard HTTP, simulând o pagină legitimă de autentificare Facebook. Este utilizat în contextul unui atac DNS hijacking, unde utilizatorul este direcționat către această pagină în loc de site-ul real.

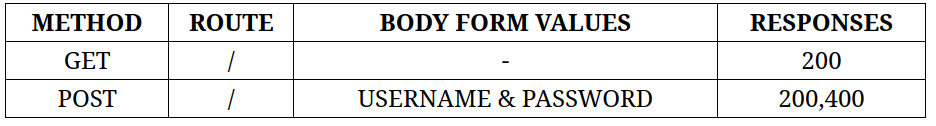


Fig 4.9

#### **Metoda “GET” la “/”** Returnează pagina HTML statică ce imită interfața de autentificare Facebook. - Conține un formular cu câmpuri pentru username și parolă. - Această pagină este aproape identică vizual cu pagina reală, dar este servită de ESP32 fără TLS (HTTP simplu).

#### **Metoda “POST” la “/flood”** Este activat când utilizatorul introduce datele în formular și apasă „Log in”. Serverul procesează formularul și salvează local datele introduse. Nu există nicio validare reală – pagina de răspuns poate afișa un mesaj generic sau redirect în buclă. Informațiile pot fi afișate ulterior în interfață de către utilizator (ex: printr-un buton în / din serverul principal care citește aceste date).

### **4.10.3 Serverul DNS**

Acest server DNS minimalist este activ doar pentru a intercepta cererile DNS ale dispozitivelor conectate la ESP32. După conectare prin DHCP, dispozitivul client primește adresa ESP32-ului ca DNS principal. Astfel, toate interogările DNS sunt direcționate către ESP32.

Comportamentul este următorul:  
Dacă se solicită un domeniu care conține facebook (ex: [www.facebook.com):](http://www.facebook.com):)  
 Serverul returnează IP-ul local al ESP32, obligând clientul să acceseze pagina falsă.  
Pentru orice altă cerere:  
 Fie este blocată (returnând un răspuns vid sau 127.0.0.1),  
 Fie este ignorată, în funcție de implementare.

Această redirecționare permite simularea unui DNS hijacking complet local, fără acces la internet și fără intervenții externe. Victima crede că a fost conectată la internet și deschide site-ul Facebook, dar în realitate interacționează cu ESP32.

# Rezultate obținute și analiza datelor

Acest capitol este dedicat prezentării testelor realizate asupra aplicației dezvoltate pe microcontrollerul ESP32, având ca scop evaluarea stabilității, eficienței și comportamentului funcțional al sistemului în condiții variate de execuție. Testele acoperă aspecte precum stabilitatea codului în timp, consumul de resurse hardware (memorie și procesor), comportamentul interfeței web în scenarii de utilizare succesivă sau concurentă, precum și verificarea corectitudinii datelor procesate. S-au avut în vedere atât execuții repetate, cât și utilizare combinată a funcționalităților, într-un efort de a valida robustețea sistemului și coerența logicii interne.

## **5.1. Testarea stabilitatii in timp**

Testarea stabilității aplicației reprezintă o etapă esențială în validarea fiabilității sistemului embedded dezvoltat pe microcontrollerul ESP32. Fiind o platformă cu resurse hardware limitate și cu funcționalități multiple care rulează în paralel (server HTTP, server DNS, server web de phishing, scanare Wi-Fi, transmitere de cadre), este important de verificat dacă aplicația își poate menține funcționarea corectă pe o perioadă extinsă de timp, fără degradări funcționale, blocaje sau consum progresiv de resurse.  
Pentru acest test, dispozitivul ESP32 a fost alimentat continuu și a fost lăsat să ruleze timp de 8 de ore fără întrerupere. În acest interval:

- serverul DNS a fost activ în permanență.  
 - serverul web de phishing a fost disponibil la portul 80.  
 - serverul principal web (port 8000) a fost accesat periodic pentru scanare și activarea atacurilor.

La fiecare 15 minute, a fost realizată o interogare GET /scan, urmată de o activare alternativă a funcției /flood și /flood/stop, în scopul de a simula utilizare regulată și iterativă. De asemenea, la fiecare oră, a fost declanșat un atac GET /attack?bssid=...&timeout=..., pe o durată limitată.  
În urma testului:

- ESP32 nu a suferit nicio resetare automată.  
 - utilizatorii au putut accesa interfața web pe întreaga durată a testului.  
 - nu au apărut erori fatale sau pierderi de memorie detectabile vizual (prin serial log).

Aplicația s-a comportat stabil în regim de funcționare continuă, cu execuții repetate ale funcțiilor critice, confirmând robustețea implementării și gestionarea corectă a resurselor la nivel de cod. Stabilitatea este cu atât mai notabilă cu cât toate serverele rulează concurent pe același dispozitiv, într-un mediu lipsit de sistem de operare complet (doar RTOS).

## **5.2. Testarea consumului de memorie si CPU**

Evaluarea consumului de resurse hardware este un pas esențial în validarea oricărei aplicații embedded. În proiectul actual, aplicația rulează pe ESP32 fără a utiliza alocare dinamică a memoriei (heap), bazându-se exclusiv pe memorie globală și memorie locală pe stivă. Această decizie de arhitectură a fost luată pentru a asigura stabilitate pe termen lung, simplitate în managementul memoriei și un comportament predictibil în execuție.  
Toate resursele necesare funcționării aplicației (buffer-e pentru pachete, structuri de stocare, răspunsuri JSON, parametri temporari) sunt prealocate static sau definite local în cadrul funcțiilor. În acest mod, nu există riscul apariției erorilor cauzate de fragmentarea heap-ului sau de lipsa memoriei în timp de execuție. Această abordare este ideală pentru aplicații care rulează pe termen lung sau care trebuie să fie reziliente în fața traficului de rețea imprevizibil.  
 Pentru analiza consumului de CPU, au fost măsurate timpii de execuție ai principalelor rute și ai task-urilor asociate fiecărei funcționalități, folosind funcția esp\_timer\_get\_time(), care returnează timpul în microsecunde de la pornirea sistemului. Măsurătorile au fost realizate pe mai multe cicluri de rulare, iar valorile prezentate reprezintă medii aproximative.

|  |  |
| --- | --- |
| Ruta | Timp mediu de executie |
| / | 12 ms |
| /flood | 2.3 ms |
| /flood/stop | 7 ms |
| /scan | ~ 1000 ms |
| /attack | N s (conform durata introdusa de utilizator) |
| /facebook | 9 ms |
| Servirea paginii de phishing | 11 ms |
| Server DNS | 3 ms |

Este de remarcat că toate rutele și componentele serverului, cu excepția celor care presupun operațiuni directe de rețea (scanarea punctelor de acces și atacurile), răspund într-un interval foarte scurt, între 2 și 20 milisecunde. Acest comportament asigură o experiență fluidă pentru utilizator și permite rularea în paralel a tuturor funcționalităților fără afectarea performanței generale.  
În cazul rutei /scan, întârzierea este inevitabilă, deoarece procesul de scanare activă necesită ascultarea pe toate canalele și agregarea rezultatelor, ceea ce durează aproximativ o secundă. În cazul rutei /attack, durata este stabilită de utilizator prin parametrul timeout, iar ESP32 va transmite cadrele de deautentificare pe întreaga durată specificată.

## **5.3. Testarea validarii datelor introduse de utilizator**

În cadrul aplicației dezvoltate, validarea datelor introduse de utilizator joacă un rol important în prevenirea comportamentului neașteptat și în asigurarea robusteței generale a sistemului. Deși aplicația rulează într-un mediu embedded restrâns și nu presupune interacțiune complexă de tip formular cu back-end sofisticat, există totuși mai multe puncte de intrare în sistem unde datele introduse de utilizator trebuie validate corect: prin rute(/attack), în pagina de phishing și în interfața DNS.

**Validarea parmetrilor in cererile http**

Principala zonă în care utilizatorul introduce date direct este în apelul către ruta:

GET /attack?bssid=<MAC>&timeout=<secunde>

În acest caz, aplicația verifică:  
 - prezența ambilor parametri (bssid și timeout);  
 - formatul adresei BSSID – se verifică dacă stringul conține exact 17 caractere și are separatori “:”;

- valoarea numerică a timeout-ului – se verifică dacă este pozitivă.

Dacă unul dintre parametri lipsește sau este invalid, serverul răspunde cu:

400 Bad Request – în cazul parametrilor absenți sau greșit formatați;

500 Internal Server Error – în cazul unor erori interne de execuție.

Această validare este implementată local, fără biblioteci externe, prin parsing manual al stringului din URL. Fiind o aplicație embedded, resursele sunt limitate, deci s-a optat pentru parsare eficientă și validare strictă, fără toleranță pentru valori ambigue.

**Validarea în formularul de phishing**

Pagina de phishing imită un formular de autentificare Facebook, unde utilizatorul introduce un username și o parolă. La apăsarea butonului de trimitere (POST /), aplicația:  
 - verifică dacă ambele câmpuri sunt completate (nu sunt stringuri vide);  
 - salvează local perechea în format JSON, într-o structură de date predefinită;  
 - afișează un mesaj de confirmare în interfața web (sau redirecționează pagina).

Fiind o simulare, nu există validare semantică (ex: dacă username-ul seamănă cu un email real), însă se testează capacitatea aplicației de a procesa și reține date multiple în ordine cronologică.

**Validarea cererilor DNS**

Serverul DNS de pe ESP32 analizează numele de domeniu primit și compară stringul cu termeni cheie (ex: facebook). Dacă stringul este nevalid sau este un domeniu care nu trebuie interceptat, cererea este ignorată sau returnează o adresă nulă. Acest comportament asigură că aplicația răspunde doar la cererile relevante și nu afectează domenii neintenționate.

## **5.4. Testarea comportamentului aplicației la execuții succesive și simultane**

În validarea oricărei aplicații embedded expuse la interacțiune externă, un aspect critic este comportamentul acesteia în condiții de execuții repetate și în prezența cererilor concurente. În cazul proiectului de față, aplicația rulează pe microcontrollerul ESP32 și expune funcționalități multiple prin trei servere: unul web (port 8000), unul DNS (port 53) și unul web clasic pentru phishing (port 80). Deși toate aceste componente rulează simultan la nivel de sistem, serverul web este implementat în mod single-threaded, ceea ce înseamnă că gestionează cererile pe rând, într-o buclă principală de procesare.

#### **Execuții succesive**

Testarea execuțiilor repetate a implicat rularea în buclă a principalelor funcții ale aplicației, cu apeluri repetate la intervale scurte:

- rutele /scan, /flood, /flood/stop și /attack au fost apelate secvențial timp de 15 minute;  
 - fiecare execuție a fost urmată de o verificare a funcționării altor componente (DNS, phishing);  
 - contextul rețelei Wi-Fi a fost resetat și reinițializat după fiecare scanare sau atac.

Aplicația a gestionat toate cererile în mod corect, fără blocaje sau răspunsuri invalide. Chiar și în lipsa unui mecanism de concurență reală în serverul web, execuția secvențială a fost suficientă pentru a răspunde prompt, dat fiind că majoritatea cererilor sunt scurte și nu blochează procesorul pentru perioade lungi.

#### **Execuții simultane simulate**

Deși serverul este single-threaded, testarea a inclus scenarii în care mai mulți utilizatori au trimis cereri aproape simultan, fie de pe dispozitive diferite, fie din file multiple ale aceluiași browser.   
Exemple de teste:

- un utilizator activează /flood, în timp ce altul deschide pagina HTML de control;  
 - /scan este apelat imediat după începerea unui /attack;  
 - un client trimite date în formularul de phishing, în timp ce altul accesează logurile.

Datorită naturii single-threaded a serverului, cererile sunt puse în coadă și procesate în ordinea sosirii. Chiar dacă nu sunt procesate în paralel, aplicația asigură integritate și consistență, deoarece:

- fiecare cerere este procesată complet înainte de a începe următoarea;  
 - nu există acces concurent la resursele globale partajate;  
 - buffer-ele de răspuns și variabilele sunt fie locale, fie gestionate în mod exclusiv de contextul fiecărei cereri.

Această abordare reduce complexitatea codului și elimină nevoia de sincronizare sau mutexuri. Singura limitare observată este că în cazul cererilor de durată mai mare (ex: /scan sau un /attack activ), cererile următoare trebuie să aștepte finalizarea, ceea ce poate introduce un mic delay.

## **5.5. Testarea adaptării interfeței web la variații de densitate grafică și dispozitive**

Aplicația dezvoltată pe ESP32 expune o interfață web simplă, accesibilă de pe orice dispozitiv conectat la rețeaua generată de microcontroller. Această interfață este esențială pentru controlul funcțiilor disponibile (scanare, flood, deauthentication, vizualizare date capturate) și trebuie să funcționeze corect pe o gamă variată de dispozitive cu dimensiuni de ecran și densități grafice diferite.

Testele au fost efectuate folosind două platforme principale:  
 - Smartphone Samsung S24 Ultra, cu ecran de înaltă rezoluție și densitate mare (~500+ PPI);  
 - PC Desktop, cu Windows 11, folosind browserele Microsoft Edge, Mozilla Firefox si Google Chrome, pe un monitor Full HD (1920x1080 px).

Interfața web a fost testată accesând ESP32 ca punct de acces direct, fără conexiune la internet, folosind adresa IP locală 192.168.10.1.

Observații:  
 - Interfața nu utilizează tag-ul viewport specific în HTML pentru redimensionare automată în funcție de rezoluția ecranului. Cu toate acestea, structura HTML/CSS este concepută suficient de îngust pentru a nu crea probleme de afișare nici pe ecrane mici, nici pe cele mari.   
 - Pe Samsung S24 Ultra, interfața este perfect funcțională. Butoanele, textele și formularele sunt vizibile și ușor de utilizat, fără a necesita zoom sau scroll orizontal. Fiind un layout vertical și compact, elementele se adaptează implicit la lățimea ecranului.  
 - Pe PC, în ambele browsere testate (Edge și Firefox), layout-ul se menține centrat și lizibil. Fonturile sunt proporționale, iar elementele interactive funcționează corect.  
 - Butonul de submit din pagina de phishing, precum și tabelele cu loguri, sunt complet vizibile pe ambele platforme, fără suprapuneri sau erori de randare.  
 - Codul JavaScript este compatibil cu toate browserele moderne testate (Edge, Firefox, Chrome), fiind bazat pe API-uri standard, fără dependențe externe sau funcții avansate.

# Discuţii şi concluzii

Lucrarea de față a avut ca scop realizarea unei platforme demonstrative bazate pe ESP32, capabilă să simuleze și să explice funcționarea mai multor tipuri de atacuri asupra rețelelor Wi-Fi. Printre acestea s-au numărat: atacul de tip Deauthentication, AP Flood și DNS Hijacking. Proiectul a fost conceput cu un caracter didactic și demonstrativ, în scopul înțelegerii vulnerabilităților reale ale protocoalelor wireless utilizate pe scară largă.  
Obiectivele propuse la începutul lucrării au fost, în mare parte, atinse. S-a reușit implementarea cu succes a unui server web minimalist pe ESP32, capabil să răspundă la cereri HTTP și să declanșeze funcționalități asociate fiecărui atac. De asemenea, au fost construite cadre 802.11 manual (beacon-uri, deauth etc.) și transmise în eter folosind interfața radio a ESP32 în mod promiscuu. Atacul DNS Hijacking a fost implementat prin construirea unui server DNS personalizat care redirecționează selectiv traficul DNS al utilizatorului către IP-ul ESP32.  
 Printre limitările întâmpinate se numără imposibilitatea realizării unei redirecționări complete și transparente a traficului DNS. În forma actuală, răspunsurile DNS malițioase funcționează doar în contextul în care ESP32 este DNS-ul setat implicit prin DHCP. Din acest motiv, pentru direcții viitoare, se propune implementarea unui sistem de DNS forwarding combinat cu NAT/WiFi passthrough pentru ca utilizatorul să poată naviga aparent normal, în timp ce traficul DNS este interceptat și modificat în fundal. Această extindere ar contribui semnificativ la realismul simulării atacului și ar îmbunătăți caracterul demonstrativ al lucrării.  
 Rezultatele obținute au demonstrat clar vulnerabilitățile protocoalelor Wi-Fi atunci când nu sunt folosite mecanisme moderne de protecție (ex. WPA3, 802.11w, DNS-over-HTTPS). Interfața de control prin HTTP și modularitatea codului facilitează extinderea lucrării prin adăugarea de noi tipuri de atacuri sau integrarea cu platforme externe de analiză.  
 Contribuțiile personale constau în proiectarea arhitecturii software, scrierea codului pentru manipularea pachetelor 802.11, implementarea manuală a unui server DNS, gestionarea interfeței HTTP și integrarea tuturor componentelor într-un sistem coerent și ușor de demonstrat. De asemenea, a fost proiectată o interfață care permite utilizatorului să controleze atacurile dintr-un browser, fără cunoștințe avansate de rețea. Aplicațiile potențiale ale acestei lucrări includ demonstrații în laboratoare de securitate cibernetică, formarea tehnicienilor în rețelistică, precum și validarea unor soluții de detecție la nivelul rețelei wireless. Direcțiile viitoare pot include suportul pentru deautificarea dispozitivelor in retele Wi-Fi 5G și integrarea unei funcții de logging extinse, salvarea datelor într-un fișier local sau trimiterea acestora către un server extern pentru analiză centralizată.

# Bibliografie

[1] M. Vanhoef and F. Piessens, “Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2,” Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017. URL: <https://doi.org/10.1145/3133956.3134027>

[2]IEEE Standards Association, “IEEE Std 802.11™-2016,” IEEE, 2016. URL: <https://standards.ieee.org/ieee/802.11/1733/>

[3] Aircrack-ng Project. “Aircrack-ng suite,” 2023. URL: <https://www.aircrack-ng.org/>

[4] Wireshark Foundation. “Wireshark - Go Deep,” 2023. URL: <https://www.wireshark.org/>

[5] D. Ryan. “WiFi Pineapple - Wireless auditing,” Hak5, 2023. URL: <https://www.hak5.org/>

[6]Spacehuhn Technologies. “ESP8266 Deauther,” GitHub repository, 2023. URL: <https://github.com/SpacehuhnTech/esp8266_deauther>

[7] Reaver Project. “Wifite 2,” GitHub repository, 2023. URL: <https://github.com/derv82/wifite2>

[8] IEEE, “WPA3 specification,” Wi-Fi Alliance, 2018. URL: <https://www.wi-fi.org/discover-wi-fi/security>

[9] Grayson, M., McNab, J., & Turner, S. (2013). Wi-Fi Security: WPA and 802.11i. Cisco Press.

[10] Espressif Systems. (2023). ESP32 Technical Reference Manual. <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>  
[11] IEEE Std 802.11™-2020 – IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, URL: <https://ieeexplore.ieee.org/document/9363693>  
[12] Matthew Gast, 802.11 Wireless Networks: The Definitive Guide, 2nd ed., O'Reilly Media, 2005.  
[13] Andrea Bittau, Mark Handley, Joshua Lackey, The Final Nail in WEP’s Coffin, IEEE Symposium on Security and Privacy, 2006.

[14] A. Zill, “Beacon Flooding Attacks in Wireless Networks,” IEEE Wireless Communications, vol. 21, no. 3, 2014.

[15] Spacehuhn Technologies, ESP8266 Deauther, GitHub, 2023. <https://github.com/SpacehuhnTech/esp8266_deauther>

[16] Mockapetris, P. Domain names - concepts and facilities. RFC 1034, IETF, 1987. <https://tools.ietf.org/html/rfc1034>