

UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENȚĂ

**Sistem de autentificare continuă bazat pe comportamentul
utilizatorilor**

COORDONATOR ȘTIINȚIFIC

Conf. Dr. Olimid Ruxandra

ABSOLVENT

Coica Oana-Alexandra

București, iulie 2020

CUPRINS

1. INTRODUCERE	3
1.1 Contextul proiectului	3
1.2 Motivația lucrării și studiul situației actuale	3
1.3 Structura lucrării	6
2. OBIECTIVE ȘI CONTRIBUȚII	9
2.1 Scopul proiectului	9
2.2 Diferențiatorul sistemului	9
2.3 Contribuții personale.....	10
3. ANALIZĂ ȘI FUNDAMENTARE TEORETICĂ.....	11
3.1 Noțiuni introductive	11
3.2 Cerințe funcționale și non-funcționale	12
3.3 Cazuri de utilizare	13
3.4 Perspectiva tehnologică	15
3.5 Arhitectura conceptuală	18
4. PROIECTARE DE DETALIU ȘI IMPLEMENTARE	21
4.1 Colectarea datelor	21
4.2 Extragerea caracteristicilor	22
4.3 Clasificarea și decizia	24
4.4 Actualizarea șablonului.....	30
4.5 Structura bazei de date	30
5. TESTARE ȘI VALIDARE	33
5.1 Analiza arhitecturii sistemului	33
5.2 Analiza performanței sistemului	35
6. UTILIZAREA SISTEMULUI.....	39
6.1 Manual de utilizare a componentei client	39
6.2 Manual de utilizare a componentei server	46
7. CONCLUZII ȘI DEZVOLTĂRI ULTERIOARE	49
REFERINȚE	51
Anexa 1 – Secvență de cod pentru colectarea datelor brute	56
Anexa 2 – Secvență de cod pentru extragerea caracteristicilor (amprentelor)	57
Anexa 3 – Glosar	60
Anexa 4 – Lista figurilor și tabelelor	61

1. INTRODUCERE

1.1 Contextul proiectului

Tehnologia a devenit indispensabilă. Aplicațiile devin parte integrală a vieților noastre, fiind unelte esențiale pentru desfășurarea sarcinilor zilnice, iar activitățile online, de la comerțul electronic până la accesarea anumitor date confidențiale, au forțat dezvoltarea sistemelor sofisticate de autentificare, utilizatorii fiind nevoiți să-și confirme identitatea în universul digital.

Cheia de a deschide aproape orice ușă din lumea virtuală este, în mod tradițional, parola. Din perspectivă pur teoretică, parola poate asigura o securitate extrem de puternică, fiind necesară stocarea acesteia în text clar numai în mintea utilizatorului. În practică însă, un utilizator mediu are, în 2020, în jur de 207 seturi de date de autentificare de memorat, lucru mult prea dificil pentru mintea umană, astfel că majoritatea oamenilor ajung fie să reutilizeze aceeași parolă peste tot, fie să-și noteze și/sau stocheze parolele în locuri nesigure. În plus, există o mulțime de puncte vulnerabile în sistemele de securitate bazate pe parolă [1].

1.2 Motivația lucrării și studiul situației actuale

Motivația principală a lucrării o reprezintă așa-numita „problemă a parolelor”: faptul că utilizatorii, în ciuda recomandărilor, aleg parole ușoare și le reutilizează, distribuie sau neglijează necesitatea de a le schimba frecvent. Totodată, chiar și în cazul utilizării unor parole sigure (care să nu reprezinte un cuvânt din dicționar și care să nu includă date personale, cu dimensiune cât mai mare, cu toate tipurile de caractere disponibile, majuscule, minuscule, cifre și caractere speciale [2]) și renunțării la comoditate, sistemele de securitate bazate pe parolă sunt vulnerabile la multiple tipuri de atacuri, atât online (testând presupusele parole în interacțiuni directe cu un server), cât și offline (interacționând cu o bază de date cu *hash*-uri, de exemplu). Din cauza reutilizării de parole, este suficient ca un serviciu să fie compromis pentru ca alte servicii să aibă de suferit și ele.

O parte din atacurile utilizate împotriva parolelor sunt atentate criptografice sofisticate [3], fie tehnice ori non-tehnice, în timp ce altele sunt mai curând simple atacuri de forță brută (*brute force attacks*). În categoria celor tehnice intră de la atacuri de tip *wire sniffing*, ce presupun interceptarea și înregistrarea traficului de rețea brut, din care pot fi extrase datele de autentificare în cazul protocoalelor care trimit parolele necriptate, și până la atacuri de *phishing* sau *trojan horse*. În cazul atacurilor non-tehnice, acestea sunt și ele de mai multe tipuri: de la

simpla ghicire a parolei sau a răspunsurilor întrebărilor de siguranță ce permit resetarea acestora și până la atentate de *social engineering* (manipularea utilizatorului țintă în a dezvălui datele necesare autentificării) în cazul celor non-tehnice.

Soluția ce a fost propusă inițial pentru această problemă, autentificarea cu doi factori (*two-factor authentication*, 2FA), presupune acordarea accesului utilizatorului numai după ce acesta prezintă două tipuri de autentificatori dintre cei trei posibili:

- 1) cunoaștere - ceva ce știe (o parolă, un secret);
- 2) posesie - ceva ce deține (un *smart card*, un *token*, un dispozitiv);
- 3) inerentă - ceva ce este (indici biometrici).

Cea mai des întâlnită formă de *two-factor authentication* este cea compusă din primii 2 factori: utilizatorul autentificându-se la server după dovedirea faptului că are acces la un dispozitiv auxiliar, în plus față de cunoașterea propriei parole, formând astfel o apărare mai sigură în fața atacurilor online, dar și o a doua linie de apărare în cazul compromiterii parolei [4]. Însă, în ciuda încrederii inițiale într-un astfel de sistem și în ciuda faptului că este recomandată utilizarea acestei modalități de autentificare atunci când este disponibilă, nici *two-factor authentication* nu este invincibil în fața anumitor atacuri, în special atunci când dovada accesului la un dispozitiv implică introducerea unui cod primit prin SMS, care ar putea fi interceptat de o terță malițioasă (atac cunoscut sub denumirea de deturnare de SMS, sau *SMS hijacking*), prin escrocheriile de tip portare în altă rețea (*port out scams*), de pildă [5]. Totodată, autentificarea în doi factori nu oferă protecție în cazul atacurilor avansate de inginerie socială (*social engineering*).

Metodele de autentificare biometrice înlocuiesc treptat tipicele parole și chei de acces, fiind considerate cea mai eficientă realizare în domeniul identității și management-ului accesului. Ampretele, trăsăturile noastre faciale sau modul în care ne mișcăm - toate pot fi transformate în indici biometrici unici, ce pot fi utilizați pentru a ne autentifica.

În general, o trăsătură biometrică trebuie să urmărească anumite cerințe [6]:

- universalitatea presupune ca fiecare utilizator să posede caracteristica biometrică;
- unicitatea se referă la cât de diferite sunt caracteristicile biometrice de la persoană la persoană;
- permanența măsoară modul în care schimbările caracteristicilor de-a lungul timpului afectează performanța sistemului;
- măsurabilitatea exprimă cât de dificilă este colectarea și analizarea caracteristicii;
- acceptabilitatea măsoară cât de ușor acceptă un utilizator tehnologia;

- transparența se referă la posibilitatea de a colecta caracteristica în timp ce utilizatorul se ocupă de alte activități;
- minimalitatea își dorește ca măsurarea să nu necesite hardware suplimentar;
- eludarea este legată de cât de ușoară este falsificarea sau imitarea caracteristicilor biometrice;
- performanța indică viteza, acuratețea și robustețea.

Sistemele de autentificare biometrice folosesc caracteristicile fizice și comportamentale ale utilizatorilor pentru a le verifica identitatea, stocând datele biometrice într-o bază de date sigură și comparând factorii biometrici pentru a confirma autentificarea.

Indicii biometrici nu pot fi ghiciți, imitați sau falsificați în același mod în care se poate întâmpla cu o parolă obișnuită. În mod tradițional, oamenii se gândesc, atunci când vine vorba de autentificare biometrică, la recunoașterea amprente, cea facială, la scanarea irisului sau a vocii. Chiar dacă și acestea sunt metode importante în verificarea identității, există și indici bazați pe comportamentul utilizatorilor, fiind posibilă autentificarea continuă, ce presupune asigurarea că utilizatorii (care au trecut de portalul de autentificare inițial) se comportă conform unor valori de referință (unui așa-numit *baseline* [6]). Utilizatorii care nu se comportă ca ei înșiși pot fi deconectați până când identitatea este reverificată.

Printre indicii mai puțin populari ce pot fi utilizați pentru autentificare se află recunoașterea semnăturii, a modului de a scrie, merge sau mișca - toate depinzând de șabloane comportamentale, astfel procedeul de verificare a identității devenind intuitiv pentru utilizator, dar greu de imitat de cineva din extern. Din nefericire însă, toate aceste metode biometrice necesită integrarea cu hardware avansat, fiabil și stabil, un inconvenient major din perspectiva utilizatorului final, care pune pe primul loc practicabilitatea și costurile de implementare și susținere a unui astfel de sistem [7].

În ciuda acestor inconveniente, biometria bazată pe comportamentul utilizatorilor are un rol important în viitorul măsurilor de securitate, iar biometria apăsării tastei, numită și dinamica tastării, reprezintă o practică nouă ce ar putea avea un impact pozitiv în încercarea de a proteja identitatea digitală. Aceasta presupune monitorizarea și analizarea modului în care utilizatorul tastează, identificarea șablonului unic de comportament specific și legarea acestui șablon de utilizatorul în sine, pentru a-l putea autentifica mai târziu pe baza confirmării acestei identități. În plus, această metodă biometrică nu necesită hardware suplimentar, fiind suficientă o simplă tastatură. Totuși, analiza ritmului de a scrie aduce cu sine apariția anumitor preocupări

legate de intimitatea utilizatorului, care ar putea sta în calea creșterii popularității acestei metode de autentificare [8].

1.3 Structura lucrării

În Capitolul 1, a fost făcută o scurtă introducere în contextul proiectului, marcându-se necesitatea îmbunătățirii modalității de autentificare a utilizatorilor, a fost expusă motivația și studiul situației actuale, fiind descrisă problema parolelor, au fost enumerate câteva tipuri de atacuri în fața cărora sistemele de autentificare bazate pe parole sunt deseori vulnerabile, a fost definită ideea de autentificare în doi factori. Totodată, au fost descrise câteva metode biometrice de autentificare, împreună cu o parte din dezavantajele specifice acestora, și a fost făcută o comparație între acestea, ajungând la concluzia că printre cele mai sigure și poate prea puțin studiate metode de autentificare se află biometria tiparului de testare, într-un final fiind prezentată o structură pe capitole a lucrării.

Capitolul 2 descrie scopul principal al proiectului, precum și obiectivele secundare, este făcută o succintă evaluare a unor sisteme similare existente deja pe piață (*KeyTrac*, *Keystroke DNA*, *TypingDNA*, *BehavioSec*) și a fost prezentat diferențiatorul sistemului – prezența modulului de autentificare continuă, precum și evidențiată contribuția personală.

Capitolul 3 prezintă câteva noțiuni introductive (vulnerabilitatea principală a sistemelor de autentificare statică – deturnarea de sesiune –, informații despre ce presupun autentificarea continuă și cea transparentă și beneficiile pe care acestea le aduc, precum și avantajele și dezavantajele utilizării biometriei de tastare în confirmarea identității utilizatorilor), au fost enumerate cerințele funcționale și non-funcționale ale sistemului, dar și câteva cazuri de utilizare (complementar al altor sisteme de securitate ce controlează drepturile de acces, modificare sau distribuire a informațiilor; ca mecanism de protecție împotriva obiceiurilor prejudiciabile ale utilizatorilor, precum lăsarea stației nesupravegheate, dar și împotriva unui eventual furt; ca unealtă în evaluările la distanță ale studenților; pentru descurajarea partajării unui singur cont; ca înveliș de protecție asupra datelor în cazul unor atacuri reușite asupra parolei utilizatorului), a fost expusă perspectiva tehnologică (noțiuni principale care stau la baza dezvoltării acestui sistem de autentificare continuă, precum și câteva informații despre tehnologiile utilizate și motivele alegerii lor) și descrisă arhitectura conceptuală a sistemului.

În Capitolul 4, este realizată divizarea proiectării sistemului în patru etape: colectarea datelor brute, extragerea caracteristicilor care formează amprenta utilizatorului, clasificarea (determinarea dacă avem de-a face cu un impostor sau cu un comportament normal) și

actualizarea de șablon, detaliindu-se procedurile urmate, ilustrându-se rezultatele experimentelor și motivându-se deciziile luate în fiecare fază a implementării. Totodată, este prezentată structura bazei de date asociate sistemului.

În Capitolul 5, sunt descrise testele și analizele arhitecturale și cele de performanță prin care sistemul a trecut pentru etapa de validare, precum și modul de răspuns al acestuia.

În Capitolul 6, este ilustrat un manual de întrebuințare a sistemului, împărțit în două secțiuni - prima fiind adresată utilizatorului final, conținând informații despre modul de interacționare posibil cu componenta client, iar cea de-a doua, destinată administratorului sistemului, prezentând informații despre componenta server.

În Capitolul 7, ultimul din această lucrare, sunt sintetizate concluziile cercetării preliminare, proiectării sistemului și testării acestuia, fiind analizate rezultatele obținute și enumerate câteva direcții de dezvoltare pentru viitor.

2. OBIECTIVE ȘI CONTRIBUȚII

2.1 Scopul proiectului

Scopul principal al proiectului este elaborarea și implementarea unui sistem de autentificare continuă, bazat pe comportamentul utilizatorului. Deoarece elementul cel mai important al unui astfel de sistem este conceptul din spatele lui, interfața cu utilizatorul va fi considerată mai degrabă o caracteristică suplimentară, nepunându-se accentul pe aceasta.

În proiectarea sistemului se va ține cont de cerințe precum securitate, scalabilitate, utilizabilitate, disponibilitate, și se vor lua decizii în conformitate cu acestea.

Obiective secundare, dar foarte importante în atingerea scopului principal, vor fi următoarele:

- studiul biometriei de tastare ca modalitate de autentificare;
- experimentarea cu mai mulți algoritmi de învățare automată și determinarea celui ce oferă cea mai mare acuratețe la confirmarea identității utilizatorilor;
- testarea și îmbunătățirea sistemului;
- studiul punctelor vulnerabile într-un sistem biometric tradițional și eliminarea, atenuarea sau oferirea de soluții pentru moderarea acestora în sistemul propus.

2.2 Diferențiatorul sistemului

Sistemele de autentificare bazate pe dinamica tastării nu reprezintă metode populare de management al identității și accesului, dar se prefigurează o creștere a popularității acestora, datorită costurilor reduse, transparenței și modului non-invaziv [9].

Pe măsură ce tehnologia continuă să se dezvolte, iar utilizatorii devin din ce în ce mai conștienți de slăbiciunea simplelor parole, se remarcă un interes crescut și un progres la nivelul pieței de dezvoltare software, cu câteva companii, variate în mărime, scop și abilități, ce își intensifică eforturile de promovare și îmbunătățire în sensul elaborării de sisteme de autentificare avansate, bazate pe comportamentul utilizatorilor.

Unul dintre cei mai vechi jucători din acest segment de piață este KeyTrac [10], oferind serviciul de autentificare pe baza modului de a scrie, în două variante posibile: ca metodă de întărire a parolelor (*password hardening*), și ca metodă de identificare pe baza unui text liber (*anytext identification*). Totuși, serviciul este disponibil doar pentru aplicații web.

Tot pentru aplicații web sunt elaborate și produsele oferite de Keystroke DNA [11], cel mai nou jucător în acest domeniu, și TypingDNA [12], echipă românească, aflată încă în stadiul

de start-up. Toate produsele menționate anterior nu prezintă însă o funcție extrem de importantă în contextul actual - autentificarea continuă.

În contrast cu autentificarea biometrică statică, aptă să verifice identitatea utilizatorului la prima interacțiune a acestuia cu sistemul, într-o anumită sesiune, autentificarea continuă, numită uneori și autentificare dinamică sau reautentificare, vizează îndeplinirea unei cereri diferite în securitatea informației, scopul fiind acela de a se asigura că identitatea care interacționează cu sistemul este același utilizator care a trecut de procedura inițială de autentificare [13].

BehavioSec [14] este probabil cel mai cunoscut dezvoltator în biometricele comportamentale. Deși oferă și servicii de autentificare continuă, sistemul oferit de BehavioSec este elaborat îndeosebi pentru telefoanele mobile, printre caracteristicile luate în considerare fiind în special gesturile făcute pe ecranul tactil (deci implicit și felul în care tastează), modul în care utilizatorul ține dispozitivul, dar și mișcările mouse-ului.

2.3 Contribuții personale

Contribuția personală în domeniul de studiu este reprezentată de:

- explorarea situației actuale referitoare la „problema parolelor”, expusă în *Secțiunea 1.2*, înțelegerea și sintetizarea metodelor de autentificare tradiționale, continue, transparente, biometrice, a avantajelor și dezavantajelor corespunzătoare acestora;
- studierea tipurilor de atacuri și puncte vulnerabile la nivelul unor mecanisme sau sisteme de autentificare;
- dezvoltarea unui sistem de autentificare continuă, transparentă, bazat pe comportamentul utilizatorului, o unealtă software ce ar putea fi folosită la implementarea altor aplicații, conform cazurilor de utilizare din *Secțiunea 3.3*, dar nelimitându-se la acestea;
- experimentarea cu tehnologii noi, cu algoritmi de *machine learning* nestudiați în cadrul programului de licență și testarea biometriei de tastare ca metodă de verificare continuă a identității utilizatorilor.

3. ANALIZĂ ȘI FUNDAMENTARE TEORETICĂ

3.1 Noțiuni introductive

Tehnicile tradiționale de autentificare presupun verificarea statică și singulară a utilizatorului, de obicei la prima interacțiune cu sistemul în timpul acelei sesiuni. Strategia este însă vulnerabilă, întrucât sistemul nu suspectează o eventuală înlocuire a operatorului legitim cu unul neautorizat, iar utilizatorii au supărătorul obicei de a-și lăsa stația de lucru nesupravegheată fără a se deconecta de la sistemele sensibile [15]. Astfel, un impostor ar putea oricând obține accesul la date confidențiale sau permisiuni la care nu are dreptul, procedeu cunoscut sub denumirea de deturnare a sesiunii (*session hijacking*).

Este așadar necesară implementarea unui modul de autentificare continuă (*continuous authentication*), adică de verificare repetată a identității utilizatorului. Cu ajutorul acestei funcționalități, comportamentul operatorului este monitorizat pe parcursul întregii sesiuni, fiind posibilă diferențierea dintre o manifestare normală și una suspectă, caz în care are loc încheierea sesiunii sau generarea unei alerte. Autentificarea continuă aduce însă cu sine câteva provocări, precum nevoia de întârzieri cât mai mici la confirmarea identității, acuratețe cât mai mare și rezistență la tentative de falsificare [6].

Un alt concept important este acela de autentificare transparentă (*transparent authentication*), ce presupune colectarea trăsăturilor comportamentale ale utilizatorului, fără ca acesta să fie nevoit să acționeze într-un mod deosebit sau să depună vreun efort (în afara cazului înregistrării și autentificării inițiale), reducând astfel frustrarea operatorului legitim [16]. Așadar, transparența se referă la culegerea datelor despre utilizator în fundal, pe măsură ce acesta își folosește dispozitivul în mod obișnuit, conștient sau nu de această colectare a informațiilor, în timp ce continuitatea este legată de frecvența cu care se realizează verificarea identității utilizatorului.

Așa cum a fost exprimat și în prima secțiune a Capitolului 1, în cazul comparării metodelor de autentificare biometrice, sistemele bazate pe modul de a scrie al utilizatorilor (*keystroke dynamics*) se bucură de o valoare suplimentară datorită ușurinței de utilizare și a caracterului non-intrusiv, informația în legătură cu șabloanele modului de tastare putând fi obținută fără știința utilizatorului [17], tehnica fiind astfel compatibilă cu scopul de a crea un sistem de autentificare continuă și cât mai transparentă. În plus, alte avantaje ale utilizării biometriei de tastare sunt costul de implementare redus și falsificarea aproape imposibilă.

Totuși, există și dezavantaje, precum acuratețea mai mică în comparație cu alte tipuri de indici biometrici, accidentarea, oboseala, emoțiile puternice sau simpla neatenție generând schimbări în ritmul de scriere [18]. În plus, schimbarea dispozitivului aduce cu sine schimbarea tastaturii, care poate contribui la modificarea șablonului de tastare. Totodată, unul dintre dezavantaje este necesitatea de a monitoriza și înregistra fiecare apăsare de tastă efectuată de utilizator, lucru considerat ilegal în anumite state (mai ales în cazul în care utilizatorul nu este prevenit în legătură cu astfel de măsuri).

Asociațiile pentru drepturile omului sunt preocupate de implicațiile etice și sociale ale utilizării tehnologiilor biometrice. Astfel, este necesară proiectarea de algoritmi mai puțin intrusivi, mai ușor de utilizat și totuși preciși [19].

3.2 Cerințe funcționale și non-funcționale

Cerințele funcționale prezintă utilitatea sau serviciile oferite de către sistem și marchează comportamentul produsului, modul în care acesta reacționează la un anumit *input* sau la o anumită situație.

Din punct de vedere funcțional, sistemul are un scop clar: trebuie să permită autentificarea utilizatorului legitim, dar să blocheze accesul unui intrus, în urma analizei continue a comportamentului.

Cerințele non-funcționale reprezintă o listă de așteptări implicite, proprietăți/ atribute calitative, restricții sau constrângeri, de obicei aplicate asupra întregului sistem, și nu doar unor servicii individuale.

Din punct de vedere non-funcțional, sistemul de autentificare trebuie să asigure cel puțin următoarele cerințe:

- securitate - sistemul nu va permite accesul neautorizat la informație, iar datele biometrice vor fi salvate criptat în baza de date de pe server, deci furtul de către alte persoane sau aplicații nu vor compromite identitatea digitală a utilizatorului.
- scalabilitate - deoarece nu putem anticipa numărul de utilizatori ai sistemului, trebuie ca funcționalitățile să nu depindă de acest aspect, făcând posibilă migrarea în viitor la o bază de date distribuită.
- utilizabilitate - sistemul trebuie să asigure o interfață prietenoasă și intuitivă, iar utilizatorul trebuie să fie atenționat prin mesaje și notificări ori de câte ori are loc o schimbare în cerințe/ nevoi/ comportament.

- disponibilitate - sistemul trebuie să rămână stabil în fața erorilor, iar timpul în care serviciile sistemului devin indisponibile trebuie să fie minim, astfel că arhitectura trebuie să fie compatibilă cu o posibilă utilizare a tehnicilor de *load balancing*.

- extensibilitate - sistemul trebuie să fie flexibil unor dezvoltări ulterioare și să permită adăugarea/ actualizarea unor componente, atunci când cerințele se modifică, și trebuie să fie relativ ușor de integrat într-o nouă aplicație.

3.3 Cazuri de utilizare

În primul rând, un sistem de autentificare continuă, bazat pe comportamentul utilizatorilor ar putea fi implementat în complementaritatea soluțiilor de tip DLPD (detectie și prevenție a scurgerilor de informații, *Data Leak Prevention and Detection*, uneori numite doar DLP – *Data Loss Prevention*) dar și produselor software din tip DCAP (auditare și protecție centrate pe date, *Data-Centric Audit and Protection*).

Pierderea informațiilor confidențiale (fișe medicale, date financiare sau de cercetare, dosare cu informații cu caracter personal despre studenți, angajați, clienți, distribuitori, secrete comerciale, planuri de vânzări sau marketing, invenții patentabile etc.) ar putea provoca prejudicii serioase imaginii și stabilității pe termen lung ale unei organizații, și implicit și pierderi financiare, sistemele de tip DLP fiind menite să protejeze împotriva acestei categorii de riscuri, monitorizând, identificând și prevenind expunerea, atât intenționată, cât și accidentală, de date sensibile, prin impunerea de politici de utilizare a datelor și prin inspecția traficului, blocând astfel transferul de informații sensibile către părți nedemne de încredere sau nefiabile [20]. Similar, produsele de tip DCAP le oferă administratorilor un control granular asupra drepturilor de citire, modificare sau ștergere a fișierelor companiei.

Sistemele DLP și DCAP, în ciuda faptului că sunt foarte eficiente pentru a micșora numărul de breșe de securitate, devin deseori inutile în scenariile din lumea reală, în care utilizatorii își uită stațiile de lucru nesupravegheate, folosesc dispozitive comune cu ale altora, uitând sau neglijând deconectarea la finalul sesiunii sau chiar devin victime ale furtului de dispozitive – în lucrarea [21] sunt prezentate statistici referitoare la furturile de laptop-uri ale angajaților din două universități, observate pe parcursul a doi ani: în 46% din cazuri, laptop-urile au fost furate după ce au fost lăsate pentru scurt timp în locuri publice precum sălile de ședințe, în 19% din situații, furtul a fost produs atunci când angajatul nu a închis ușa de la birou, iar în 30% din cazuri, hoțul a avut acces la dispozitiv intrând prin efracție, în timp ce, conform studiilor efectuate de Institutul Ponemon, valoarea medie a unui laptop pierdut este

de 49246\$, 80% din cost luând în considerare posibilitatea unei breșe de securitate, pe locul al doilea al factorilor fiind pierderea proprietății intelectuale [22].

Este evident astfel cât de folositor pentru apărarea stratificată ar fi un control de securitate în plus, precum autentificarea continuă – chiar și în eventualitatea cazului în care utilizatorul și-ar pierde sau lăsa dispozitivul nesupravegheat (fie conectat la aplicații senzitive, fie neconectat, dar cu o parolă slabă sau stocată într-un loc nesigur), impostorul ar putea fi observat de modulul de inteligență artificială într-un timp destul de scurt, accesul la datele sensibile fiind restricționat.

În al doilea rând, sistemul și-ar putea găsi ușor locul în instituțiile de învățământ, centrele de evaluare, sau ca parte a aplicațiilor de *e-learning*, care oferă posibilitatea evaluării la distanță. Prin înrolarea celui examinat, ce ar putea avea loc însoțită de alte controale (precum cele de recunoaștere facială), comportamentul acestuia este învățat, controalele suplimentare putând fi întrerupte. Dacă pe parcursul examinării, apar schimbări abrupte în comportament, acestea ar putea indica posibilitatea ca un alt utilizator să susțină testul în locul celui legitim, profesorul/ organizatorul putând fi anunțat de aceste suspiciuni pentru a efectua verificări auxiliare sau pentru a bloca automat accesul la proba de evaluare.

O altă consecință fericită a implementării sistemului de autentificare continuă bazat pe comportamentul utilizatorului ar fi descurajarea deținerii unui cont de mai multe persoane, în cazurile în care acest lucru este nerecomandat sau interzis, aceasta fiind o aplicabilitate importantă din punct de vedere legal, deoarece este esențial ca entitățile contractante – chiar și atunci când este vorba de un contract virtual, semnat prin confirmarea citirii termenilor și condițiilor – să fie cât mai bine definite).

Poate cel mai important rol pe care l-ar avea un astfel de sistem însă este acela de a reprezenta un înveliș de securitate suplimentar împotriva fraudelor de tip preluare de cont (*account takeover*). După cum a fost menționat și în *Capitolul 1*, cea mai utilizată metodă de autentificare, aceea cu parolă, nu asigură pe atât de multă securitate pe cât am crede. Atacurile prin forță brută (*brute force attacks*) – ce presupun încercarea tuturor combinațiilor de caractere până în momentul autentificării –, atacurile de tip dicționar (*dictionary attacks*) – care implică utilizarea unei liste de cuvinte sau a unor înșiruiți comune de caractere și încercarea acestora sau a combinațiilor dintre ele într-un mod similar cu cel al atacurilor *brute-force* –, atacurile *birthday* sau coliziunile de hash (*hash collisions*) – care presupun construcția unei parole care produce același hash ca parola legitimă – nu mai pot provoca aceleași efecte dezastruoase,

comportamentul utilizatorului fiind extrem de greu de falsificat pentru a putea trece de faza a doua a autentificării.

3.4 Perspectiva tehnologică

În continuare, voi prezenta câteva concepte care stau la baza acestui sistem și voi descrie totodată tehnologiile utilizate pentru realizarea proiectului.

Pentru dezvoltarea unui sistem de autentificare continuă, este importantă înțelegerea anumitor noțiuni precum: arhitectură client-server, învățare automată (*machine learning*) și detecție de anomalii (*anomaly detection*), dar și motivația alegerii limbajului Python pentru implementare.

Structura client-server, ilustrată în mod simplist în *Figura 3.1*, reprezintă unul dintre cele mai vechi modele arhitecturale și este bazată pe împărțirea de funcții între două tipuri de componente [23], distribuite, ce comunică între ele prin intermediul unei rețele: clientul (componenta care solicită un anumit serviciu și trimite cereri către un sever) și serverul (componenta care reprezintă furnizorul de servicii pentru clienții de la care primește cereri).

Funcțiile principale îndeplinite de componenta client sunt: gestionarea interfeței cu utilizatorul, verificarea și acceptarea sintaxei în cadrul datelor introduse de utilizator, procesarea logicii aplicației, generarea și transmiterea cererilor pentru server, răspunderea la solicitările suplimentare venite din partea server-ului. Funcțiile principale îndeplinite de componenta server sunt următoarele: acceptarea și procesarea cererilor venite de la client, verificarea autorizării și constrângerilor de integritate, răspunderea la cererile clientului [23].

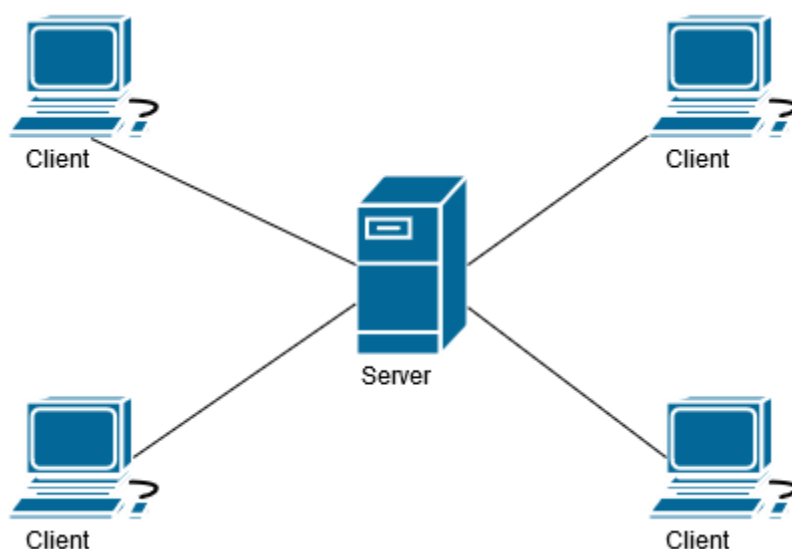


Figura 3.1: Topologie de tip client-server

Printre avantajele utilizării arhitecturii client-server se numără securitatea (datele fiind stocate pe server, sunt mai ușor de protejat decât dacă ar fi răspândite pe un număr mare de stații client), performanța (procesarea este distribuită între client și server), interoperabilitatea sistemului (arhitectura permite ca tipuri diferite de clienți, dispozitive de rețea și servere să lucreze împreună), scalabilitatea (natura modulară a modelului face ca un sistem bazat pe această arhitectură să fie extrem de receptiv la noi tehnologii și software, și hardware, fiind posibilă înlocuirea unei componente fără a fi afectat restul sistemului), accesul centralizat la date (ce oferă flexibilitate și simplitate) și ușurința la mentenanță (clientul nu cunoaște detalii despre server, astfel că activități precum cele de upgrade nu afectează funcționarea clientului). Totuși, modelul client-server cunoaște și câteva limitări, precum complexitatea și utilizarea excesivă a lățimii de bandă a rețelei pentru comunicare, dar și faptul că server-ul devine punct unic de încredere – este suficient ca acesta să fie compromis pentru ca tot sistemul să se prăbușească [23] [24].

În sens larg, sistemele de inteligență artificială pot fi definite ca fiind motoare de decizii luate de computere ce pot atinge o inteligență similară cu cea a omului. Învățarea automată este o ramură a inteligenței artificiale care se ocupă de unul dintre aspectele acestei misiuni, referindu-se la algoritmi și procese capabile să „învețe”, în sensul capacității de a generaliza experiențe (date) din trecut pentru a „intui” rezultate viitoare [25].

Utilizările învățării automate în domeniul securității informatice pot fi distribuite în două clase mari: recunoașterea tiparelor (*pattern recognition*) și detecția anomaliilor (*anomaly detection*), linia care diferențiază cele două clase nefiind însă întotdeauna foarte clară, deși cele două categorii au obiective bine stabilite.

Când vine vorba de recunoașterea tiparelor, se încearcă descoperirea unor caracteristici ascunse într-o anumită cantitate de date, parametri ce pot fi utilizați în algoritmi, la recunoașterea altor forme de date cu aceleași caracteristici.

În cazul detecției de anomalii, în locul învățării unor tipare specifice unui subset de date, obiectivul este acela de a stabili niște valori de referință, un *baseline*, ce descrie majoritatea datelor, astfel încât deviațiile să fie detectate ca anomalii [25].

Deoarece învățarea automată implică adeseori foarte multă matematică, lizibilitatea și ușurința de a transpune ideile în cod sunt foarte importante, făcând astfel din Python limbajul preferat în acest domeniu.

În plus, procesul prin care se trece pentru a dezvolta o abordare de învățare automată nu este compatibil cu o metodologie de tip dezvoltare în cascadă (*waterfall*), ci este similar mai degrabă cu o abordare de încercare și eroare (*trial and error*), analizând diferite tipuri de date, diferiți algoritmi, tocmai acest caracter exploratoriu potrivindu-se atât de bine cu limbajul Python [26].

Mediul de dezvoltare utilizat va fi PyCharm [27], datorită familiarității cu acesta, popularității, suportului pentru librăriile științifice și opțiunilor de refactorizare a codului.

Pentru proiectarea interfeței grafice, vom utiliza Qt [28] (și librăria PyQt5 [29], ce-l asociază cu limbajul de programare Python), un *framework* puternic și flexibil, folosit pentru elaborarea aplicațiilor sofisticate *cross-platform* [30], mult superior librăriei standard Tkinter incluse în pachetul de instalare Python.

Pentru baza de date, am ales să folosesc MySQL (și implicit, mysql.connector pentru asocierea cu Python) [31], acesta fiind cel mai popular sistem de gestiune *open-source* (în 2015, peste 80% din site-urile web din lume foloseau MySQL), asigurând accesul la o bază de date rapidă, stabilă și de dimensiuni reduse atunci când este nevoie [32].

Printre librăriile utilizate vor fi și socket [33] – pentru comunicarea dintre componenta server și componenta client, threading [34] – pentru construirea mai multor fire de execuție (astfel încât interfața grafică să nu înghețe în momentul în care în fundal se efectuează activități consumatoare de timp, de pildă), numpy [35] – pentru lucrul cu vectori multidimensionali, scikit-learn [36] – pentru analiza datelor și algoritmi de învățare automată (de la preprocesare, clasificare și *clustering* până la tehnici de validare).

Pentru securizarea conversației client-server va fi utilizat protocolul TLS v1.3, folosindu-se pentru implementare librăria ssl [37]. Pentru generarea certificatelor de securitate se va folosi OpenSSL [38].

Totodată, utilizați vor mai fi și algoritmi bcrypt [39] – funcție de hashing, derivată din Blowfish, având incorporată și sarea (sau salt-ul) pentru protecția împotriva atacurilor de tip *rainbow table*, SHA-256 (*Secure Hash Algorithm*) din librăria hashlib [40] – de asemenea funcție de hashing, dar și algoritmul AES (*Advanced Encryption Standard*) din MySQL – cea mai populară metodă de criptare .

Pentru toate diagramele și arhitecturile din această lucrare va fi utilizată aplicația *open-source* diagrams.net (sau draw.io) [41], datorită simplității și flexibilității, iar pentru partea de planificare și organizare am ales să folosesc Microsoft Planner, pentru interfața intuitivă, ușurința de setare a termenelor limită și graficele de monitorizare a activităților rămase [42].

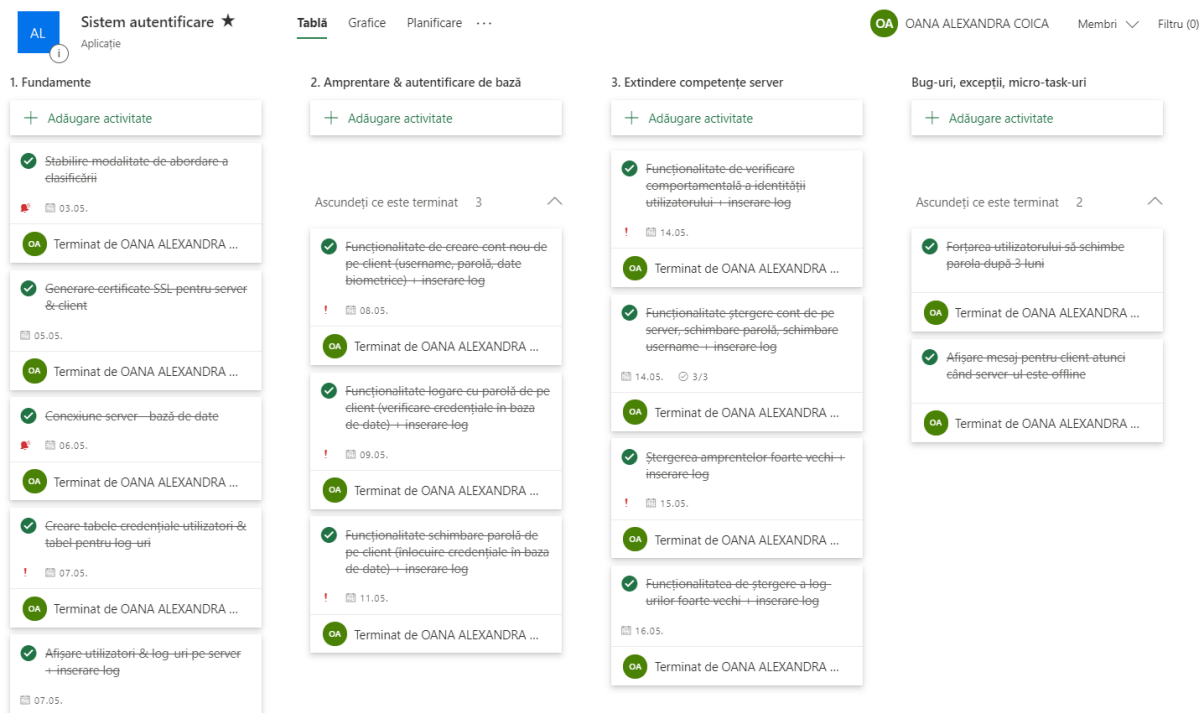


Figura 3.2: Captură task-uri din Microsoft Planner asociate aplicației

3.5 Arhitectura conceptuală

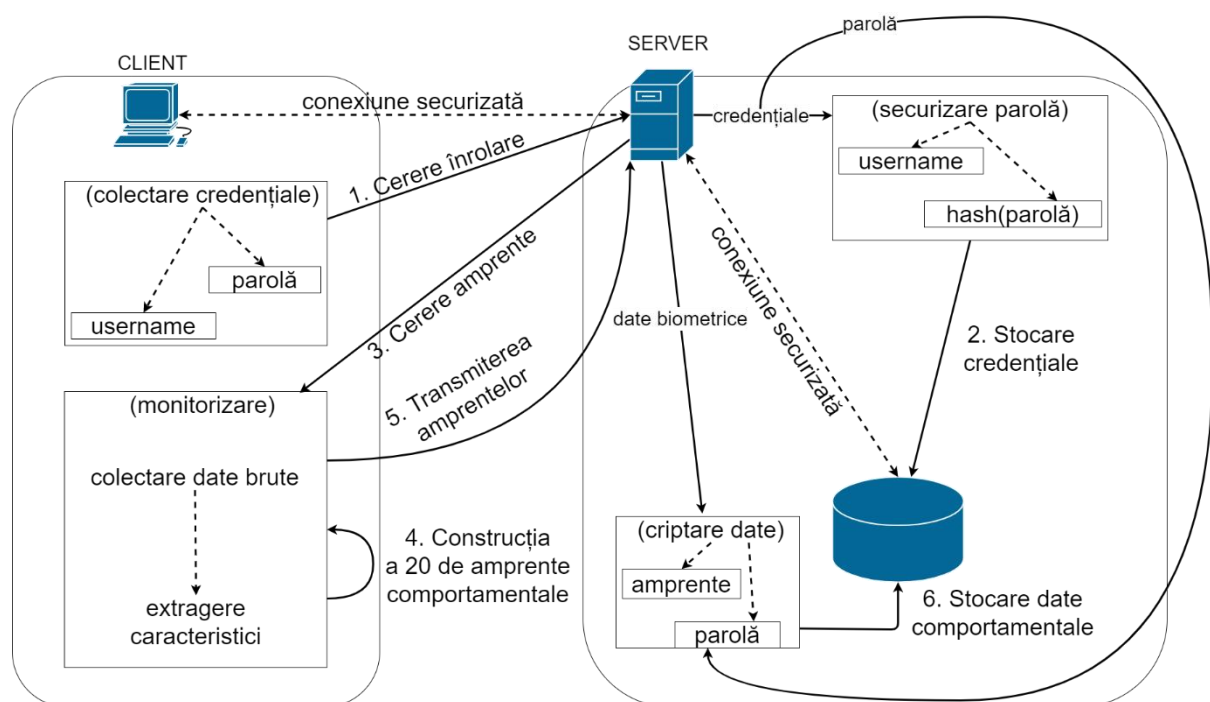


Figura 3.3: Arhitectura conceptuală a sistemului pentru înrolarea unui utilizator

Pentru simplificarea reprezentărilor arhitecturale, interacțiunile dintre componentele sistemului au fost separate în două secțiuni, fiecareia fiindu-i asociată o diagramă conceptuală: procedura de înrolare a unui utilizator nou este ilustrată în *Figura 3.3*, iar procedura de autentificare a unui utilizator înregistrat, în *Figura 3.4*. Tot pentru o consultare mai ușoară, au fost luate în calcul pentru ilustrare numai cazurile ideale.

În scenariul înrolării, în cazul în care numele de utilizator ales are mai puțin de cinci caractere și/sau parola mai puțin de zece, cererea de înregistrare nu este transmisă către server, utilizatorul fiind notificat în legătură cu restricțiile nerespectate. Totodată, în cazul în care există deja un utilizator al sistemului cu un nume de utilizator identic cu cel solicitat la înrolare, cererea este respinsă cu o notificare sugestivă.

În cazul în care pașii 4-6 ai înrolării nu sunt finalizați (fie din cauza închiderii forțate a aplicației, fie din cauza unei eventuale pierderi de conexiune cu baza de date), aceștia vor fi reluați la momentul re-conectării utilizatorului (care în cazul în care în baza de date nu se regăsesc cel puțin 20 de amprente biometrice, se poate realiza doar cu credențialele simple de acces, fără verificarea identității pe baza comportamentului).

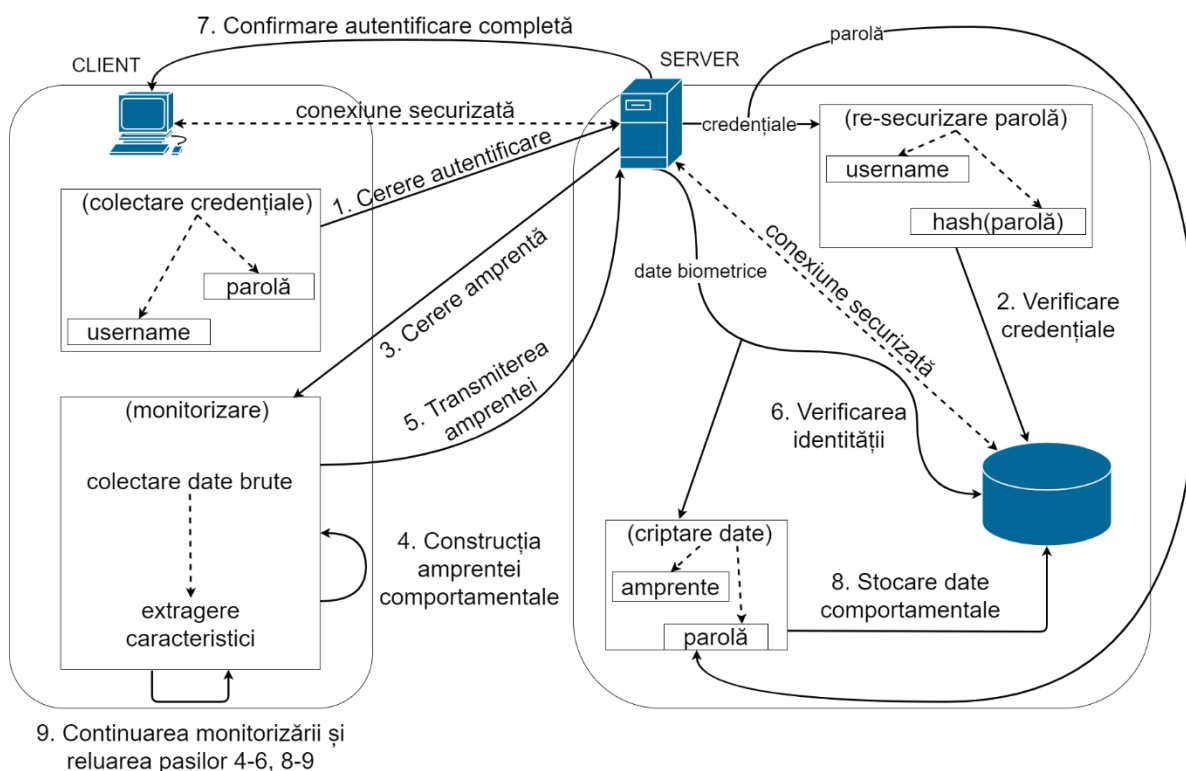


Figura 3.4: Arhitectura conceptuală a sistemului pentru autentificarea unui utilizator

Și în scenariul autentificării, similar cu situația posibilă și în cadrul înregistrării, în cazul în care câmpul/câmpurile numelui de utilizator și/sau parolei este/sunt necompletate, cererea de autentificare nu este transmisă către server, fiind respinsă local cu notificare. Și în cazul în care câmpurile sunt completate, dar în mod eronat (dacă în urma confruntării informațiilor stocate în baza de date cu cele introduse de utilizator nu există un acord), cererea este respinsă tot cu notificare.

Așa cum deja am menționat, există cazuri în care nu se intră în faza a doua a autentificării – verificarea comportamentului (pasul 6), însă monitorizarea (pasul 4), transmiterea (pasul 5) și stocarea datelor comportamentale (8) au loc chiar și în cazul în care numărul de amprente este considerat a fi insuficient – tocmai pentru culegerea acestora.

În cazul în care utilizatorul ajunge însă în faza verificării, similar cu fenomenul respingerii credențialelor neconforme (în faza întâi a autentificării), și amprente pot fi declarate a fi nelegitime dacă nu sunt recunoscute ca aparținând utilizatorului înregistrat. În această situație, clientul este deconectat complet, cu notificare (pentru reautentificare fiind necesară parcurgerea tuturor pașilor), amprenta presupusului impostor nu este stocată și este inserat un log accesibil de pe server ce cuprinde informații în legătură cu circumstanțele incidentului: numele corespunzător contului implicat, *hostname*-ul (o etichetă asociată stației, utilizată de obicei pentru identificare acesteia), adresa IP, data și ora, putându-se calcula chiar și durata în care utilizatorul respins a fost conectat ilicit la sistem.

4. PROIECTARE DE DETALIU ȘI IMPLEMENTARE

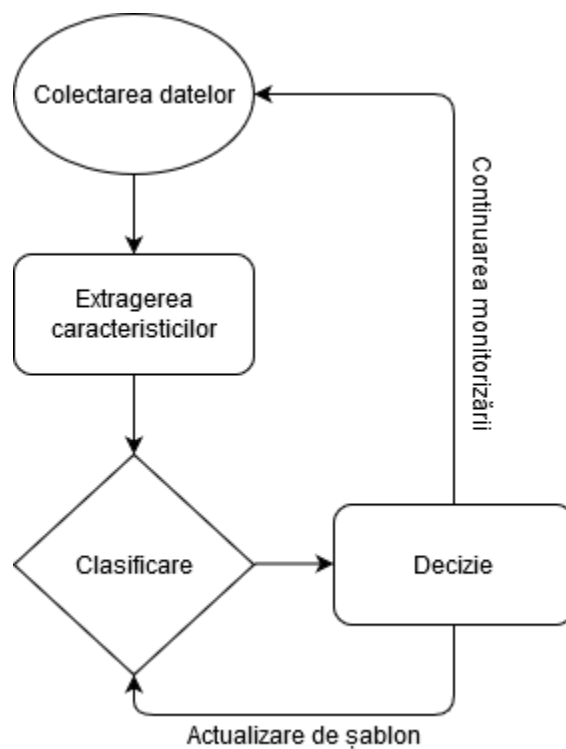


Figura 4.1: Componentele unui sistem de autentificare continuă bazat pe caracteristici comportamentale

Proiectarea unui sistem de autentificare bazat pe comportamentul utilizatorului implică divizarea în etape precum colectarea datelor, extragerea caracteristicilor, clasificarea, decizia și actualizarea șablonului, așadar vom analiza în detaliu fiecare componentă în cele ce urmează.

4.1 Colectarea datelor

Colectarea datelor este componenta fundamentală a oricărui sistem de autentificare continuă, constând în preluarea datelor brute despre comportamentul utilizatorului. Această etapă are loc, pe de o parte, la momentul înregistrării, fiind necesară colectarea datelor utilizatorului pentru a se realiza construirea unui șablon corespunzător identității înrolate, dar monitorizarea și capturarea sunt active și la momentul verificării identității - acest lucru însemnând, în cazul autentificării continue, urmărire constantă.

Datele specifice utilizatorului sunt colectate cu ajutorul unei tastaturi clasice, nefiind necesară achiziționarea unor senzori sofisticăți (chiar dacă în anumite studii, au fost testate și alte surse de informație, precum posibilitatea utilizării unor senzori de presiune, datele noi fiind utile în diferențierea mai eficientă a identităților [43]), și a unui *keylogger* cât se poate de

simplu, realizat cu ajutorul librărilor *pynput* (pentru monitorizarea dispozitivelor de input - tastatura, în acest caz) și *time* (pentru obținerea amprente de timp). Secvența de cod corespunzătoare este prezentă în *Anexa 1*.

Datele brute culese de *keylogger* constau într-o listă de evenimente ordonate cronologic. Un eveniment este alcătuit din trei segmente:

- acțiunea întreprinsă de utilizator asupra unei anumite taste (*pressed*, atunci când tasta este apăsată și *released*, atunci când tasta este eliberată);
- codul specific tastei;
- amprenta de timp a evenimentului.

O precizare foarte importantă este aceea că datele brute culese de *keylogger* nu părăsesc în niciun moment stația utilizatorului. Acestea sunt utilizate numai de componenta client a sistemului, care extrage amprenta și o trimite mai departe către server și baza de date.

4.2 Extragerea caracteristicilor

Extragerea caracteristicilor este o problemă a reducerii dimensiunilor șabloanelor rezultate din datele colectate, această etapă fiind deseori numită preprocesare. Luarea în considerare a unui subset limitat din mulțimea unităților de măsurare a datelor de intrare ar putea fi o decizie generată fie de faptul că subsetul de caracteristici ales este suficient pentru confirmarea identității utilizatorului, fie pentru că analizarea unor trăsături în plus ar crește complexitatea computațională a problemei fără a aduce un beneficiu real [44].

Dintre caracteristicile care pot fi extrase din datele de intrare brute, cele mai populare sunt durata și latența. Durata D specifică unei anumite taste n reprezintă intervalul de timp în care tasta n este apăsată, adică $D_n = R_n - P_n$, unde R_n și P_n reprezintă amprentele de timp specifice eliberării, respectiv apăsării tastei n .

Latența măsoară durata de timp dintre apăsarea sau eliberarea a două taste succesive. Putem obține, așadar, diferite tipuri de latențe: $LPP_i = P_{i+1} - P_i$ (diferența dintre amprentele de timp corespunzătoare apăsării a două taste consecutive); $LRR_i = R_{i+1} - R_i$ (diferența dintre amprentele de timp corespunzătoare eliberării a două taste consecutive); $LPR_i = P_{i+1} - R_i$ (diferența dintre amprentele de timp corespunzătoare apăsării unei taste și eliberării celei de dinaintea ei), cea mai comună fiind prima dintre cele enumerate [43].

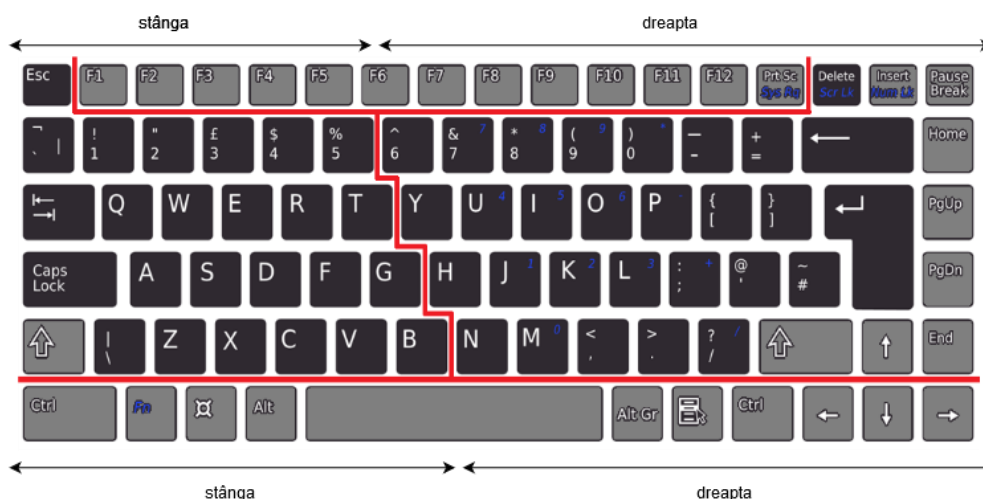


Figura 4.2: Împărțirea logică a tastaturii pentru extragerea caracteristicilor

Alte caracteristici colectabile ar putea fi: valorile mediilor și deviațiilor standard pentru fiecare din tipurile de date extrase, media numărului de taste apășate într-un anumit interval de timp, numărul de cuvinte în unitatea de timp, numărul mediu de greșeli de scriere și altele, dar aceste tipuri de caracteristici ar putea genera rezultate inconsistente, întrucât ar putea varia extrem de mult pentru același utilizator în funcție de natura textului scris.

Pentru această etapă, am proiectat un extractor de caracteristici și am împărțit tastatura în două secțiuni, corespunzătoare fiecărei mâini a utilizatorului, așa cum se poate observa în Figura 4.2.

Caracteristicile extrase au fost următoarele: media duratelor pentru tastele din partea stângă, media duratelor pentru tastele din partea dreaptă, media duratelor pentru tasta *Spațiu* (*Space*), media latențelor $LPP_i = P_{i+1} - P_i$, $LRR_i = R_{i+1} - R_i$ și $LPR_i = P_{i+1} - R_i$ (explicate anterior) pentru trecerile de la o tastă din partea stângă la o altă tastă din partea stângă, pentru trecerea de la o tastă din partea stângă la o tastă din partea dreaptă, pentru trecerea de la o tastă din partea dreaptă la o tastă din partea stângă, respectiv pentru trecerea de la o tastă din partea dreaptă la o altă tastă din partea dreaptă, obținând astfel trei durate și doisprezece latențe. Datorită modului în care au fost alese, probabilitatea de a avea caracteristici reprezentate de valori nule în „amprente” utilizatorului este foarte mică, nefiind nevoie să apelăm la metode de curățare sau imputare statistică.

O parte din evenimentele captate de keylogger – de exemplu, cele referitoare la tastele funcționale, cele de navigare și câteva dintre cele speciale (Ctrl, Alt, Shift, Windows, Esc etc.), precum și apășările continue de tastă (mai multe evenimente de tip *pressed*, fără un eveniment

de tip *released* asociat), au fost filtrate, fiind considerate irelevante pentru determinarea modului de a tasta al utilizatorului. Secvența de cod corespunzătoare extragerii caracteristicilor (ușor simplificată) poate fi regăsită în *Anexa 2*.

Caracteristicile extrase în această etapă sunt acele informații care vor părăsi stația utilizatorului, fiind transportate către server, în mod sigur, cu ajutorul protocolului TLSv1.3, această amprentă a utilizatorului urmând a fi stocată în mod criptat în baza de date, în tabelul destinat, și accesată și folosită doar pentru procesul de validare a identității utilizatorului corespunzător, nu pentru a diferenția toți utilizatorii sistemului. Altfel spus, nu vom trata problema ca pe una de clasificare multi-clasă (în momentul verificării unui utilizator nu vom încerca să „ghicim” cui îi aparține comportamentul analizat – dintre toți utilizatorii pe care îi avem înregistrați), ci ca pe o problemă de detecție a anomaliilor (verificăm dacă utilizatorul este legitim, în cazul în care comportamentul său este în parametri normali, sau impostor, în caz contrar).

4.3 Clasificarea și decizia

După extragerea și decriptarea amprentelor utilizatorului din baza de date, acestea intră în etapa de preprocesare. Algoritmii de învățare automată nu reacționează foarte bine atunci există diferențe mari între valorile pe care fiecare atribut le poate lua [45]. De exemplu, dacă una dintre caracteristicile amprentei utilizatorului ar fi vârsta, iar o alta ar fi numărul de degete utilizate pentru a tasta, deoarece valorile pe care le-ar putea lua vârsta ar putea fi cu mult mai mari decât valorile numărului de degete utilizate, vârsta ar avea un impact mai mare în decizia finală de impostor versus utilizator legitim.

Cu ajutorul scalării datelor, putem reduce impactul caracteristicilor cu valori mari extrase poate pe o scală diferită și putem ajuta caracteristicile cu valori mai mici să contribuie la fel de mult în stabilirea comportamentului obișnuit al utilizatorului [46]. Printre metodele comune de a diminua dezechilibrul se află standardizarea Z-score (*Z-score standardization*) și scalarea MinMax (*MinMax Scaling*), cunoscută și sub numele de normalizare [47].

Standardizarea este un mod de a scala datele, transformând vectorii de caracteristici astfel încât media acestora să fie egală cu 0 și deviația standard, σ , egală cu 1. Valoarea rezultată reprezintă numărul de deviații standard dintre valoarea originală și media valorilor atributului respectiv (numit și *z-score* în domeniul statisticii) [48]:

$$x_i' = \frac{x_i - \bar{x}}{\sigma}$$

Cea de-a doua tehnică foarte populară, normalizarea, utilizează punctele de minim și maxim ale fiecărui atribut pentru a reduce valorile într-un anumit interval, de cele mai multe ori $[0, 1]$. Mai exact, se efectuează următorul calcul:

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Deoarece dinamica tastării nu este printre cel mai studiate subdomenii ale biometriei, numărul de seturi de date disponibile publicului este foarte mic. Totodată, cea mai parte din colecțiile de date fie sunt bazate pe text fix (scrierea în mod repetat a aceleiași parole, propoziții), neexaminând comportamentul utilizatorului decât pentru perioade foarte scurte de timp, fiind asociate mai degrabă studiilor care abordează modul de a tasta numai ca factor secundar la autentificare, nu pentru o autentificare continuă, fie pun la dispoziție caracteristicile deja extrase din textul scris de către utilizator.

O abordare nouă, cu un extractor de caracteristici original, necesită un set de date mai autentic, nealterat, simplu, care să surprindă doar momentele de timp ale apăsării și eliberării unei anumite taste. Cea mai apropiată astfel de colecție a fost cea utilizată în lucrarea [49], un studiu amplu cu privire la 136 de milioane de apăsări de tastă, culese de la aproape 200 de mii de utilizatori, conținând printre altele, ID-ul unic al participantului, momentul apăsării tastei, momentul eliberării tastei, codul tastei, dar și metadata despre utilizator (precum vârsta, genul, informații dacă participantul a luat vreodată cursuri de dactilografie, modul de amplasare al tastelor – QWERTY, AZERTY, QWERTZ –, limba nativă, numărul de degete utilizate pentru a tasta, numărul mediu de ore petrecut zilnic în fața unei tastaturi, tipul de tastatură, rata erorilor, media cuvintelor pe minut, media intervalelor de trecere de la o tastă la alta, numărul de corecții de erori per caracter, durata medie și raportul de apăsări de tastă simultane) [50].

Din această bază de date impresionantă am utilizat doar seturile corespunzătoare utilizatorilor care au tastat suficient de mult pentru a-mi putea permite extragerea a cel puțin 20 de amprente, acesta fiind și numărul de înregistrări care vor fi culese ca parte din înrolarea în sistem – rezultând astfel 250 de mini-seturi de date.

Totuși, pentru a evalua capacitățile modelului pe termen lung, dat fiind faptul că sistemul continuă să monitorizeze și să stocheze amprente utilizatorilor și după momentul înregistrării propriu-zise, a fost nevoie și de niște seturi de date mai consistente. Am cules astfel între 80 și 150 de amprente de la câte șase voluntari. Numărul de utilizatori pe care s-au făcut astfel validările și testele nu este unul impresionant, întrucât de obicei, astfel de studii ar trebui realizate pe sute de mii de persoane pentru ca rezultatele să fie semnificative din punct de vedere statistic, dar din cauza deținerii unei cantități atât de mici de date și încrederii necesare înmânării amprentelor biometrice, chiar și studiile cu dimensiuni mai reduse ar putea fi folositoare în acest stadiu.

Printre algoritmi testati în această lucrare se află DBSCAN – algoritmul de clusterizare spațială bazată pe densitate pentru aplicațiile cu variații incontrolabile (*Density-based spatial clustering of applications with noise*) –, K-means, Elliptic Envelope, Mașini cu suport vectorial One-Class (OC-SVM sau One-Class *Support Vector Machines*) și Local Outlier Factor (LOF).

Potențialul fiecărui algoritm a fost măsurat, pentru fiecare voluntar, cu ajutorul mai multor indicatori: acuratețea la antrenare (*training accuracy*, notată TA), rata de acceptare falsă (*false acceptance rate*, notat FAR) și rata de respingere falsă (*false rejection rate FRR*), în timp ce pentru mini-seturile de date au fost utilizate doar primele două dintre măsuri (din cauza numărul mic de date și datorită faptului că acuratețea la antrenare influențează destul de mult indicatorul FRR final). Toți acești indicatori au la bază alte patru componente: numărul de pozitive adevărate (*true positive*, notat TP), numărul de negative adevărate (*true negative*, notat TN), numărul de pozitive false (*false positive*, notat FP) și numărul de negative false (*false negative*, notat FN).

Numărul de pozitive adevărate reprezintă situațiile în care modelul oferă un răspuns afirmativ adevărat, în timp ce numărul de negative adevărate situațiile în care modelul oferă un răspuns negativ adevărat. Similar, numărul de pozitive false reprezintă situațiile în care modelul oferă un răspuns afirmativ fals, iar numărul de negative false situațiile în care modelul oferă un răspuns negativ fals.

Tipic, acuratețea, notată cu A , în cazul clasificării, reprezintă numărul de situații în care modelul oferă un răspuns adevărat, raportat la numărul total de răspunsuri, fiind măsurată în felul următor:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

Acuratețea la antrenare reprezintă acuratețea modelului pe exemplele pe care este antrenat, mai exact, în cazul nostru, pe exemplele care constituie amprente utilizatorului din baza de date, folosite pentru a determina comportamentul său de referință. Pentru calcularea acestei valori este folosită, de obicei, metoda împărțirii setului de date în set de antrenare și set de test, cel din urmă fiind utilizat pentru a evalua performanța modelului, comparând rezultatele așteptate cu cele oferite de către algoritm. În cazul seturilor de date mari, acuratețea oferită de această metodă este apropiată de acuratețea reală, dar atunci când este vorba de seturi de date mai mici, rezultatele nu sunt foarte reușite, unul dintre motive fiind acela că setul de test devine mai puțin reprezentativ pentru setul de date original, rezultând estimări neconforme cu realitatea [51].

O metodă mult mai bună de a calcula acuratețea este tehnica numită validare încrucișată cu k pliuri (*K-Fold Cross-Validation*) [52]. Alegem valoarea k și împărțim setul de date în k subseturi. Antrenăm modelul pe $k-1$ subseturi și testăm pe cel neutilizat, obținând o acuratețe pentru pliul respectiv, repetând acest procedeu de k ori, până fiecare subset a devenit la un moment dat subset de testare. Agregând valorile obținute pentru fiecare pli, obținem acuratețea de antrenare finală – mult mai apropiată de acuratețea reală.

Ceilalți doi indicatori ai performanței, rata de acceptare falsă (FAR) și rata de respingere falsă (FRR), sunt calculați după următoarele formule:

$$FAR = \frac{FP}{FP + TN}; \quad FRR = \frac{FN}{FN + TP}$$

Prima dintre măsuri se referă la numărul de probe de la utilizatori neautorizați confundate cu probe de la utilizatorul legitim, raportat la numărul de probe de la impostori, în timp ce a doua reprezintă raportul dintre numărul de probe de la utilizatorul legitim respinse și numărul total de probe legitime.

Clusterizarea sau clustering-ul reprezintă procesul de a structura datele, grupând obiectele similare în mulțimi numite clustere astfel încât variația din submulțimea respectivă față de media cluster-ului să fie minimizată.

Printre cele mai populare metode de clustering sunt DBSCAN și K-means, iar felul în care pot fi utilizate pentru detecția de anomalii este similar pentru cei doi algoritmi: în urma determinării punctelor care definesc clusterelor, se calculează distanța de la aceste puncte la proba care trebuie testată, comparându-se distanța minimă obținută cu o toleranță ε , abaterea maximă acceptată între vecinii de cluster. Dacă distanța minimă este mai mare decât ε , proba este numită anomalie [53].

K-means funcționează în felul următor: sunt alese k obiecte din setul de date și sunt numite centroizi. După inițializare, este pornit un ciclu din două etape: fiecărui obiect îi este alocat unui centroid (formându-se astfel clusterelor), iar mai apoi sunt creați noi centroizi calculându-se media dintre toate obiectele alocate fiecărui fost centroid. Ciclul se încheie atunci când diferența dintre centroizii vechi și cei noi devine neglijabilă.

Printre dezavantajele utilizării K-means se numără inflexibilitatea, faptul că alegerea numărului de clusterelor se dovedește a fi, de cele mai multe ori, o sarcină dificilă, dar și faptul că, la final, clusterelor au formă regulată, lucru destul de improbabil în cazurile reale.

În schimb, DBSCAN utilizează informații referitoare la densitate pentru a forma clusterelor, numărul acestora depinzând de distanțele punctelor din setul de antrenare, și de doi parametri: ε epsilon și *MinPts*, folosiți pentru a determina dacă un punct aparține unui cluster sau nu. Orice punct din setul de date este fie un nucleu (dacă numărul de puncte la distanță mai mică decât ε , incluzând punctul respectiv, este mai mare decât *MinPts*), fie un punct accesibil (dacă se află la o distanță mai mică decât ε față de un nucleu, dar nu este nucleu), fie o anomalie (dacă nu este nici nucleu, nici punct accesibil) [54].

Rezultatele obținute de cei doi algoritmi, deși neremarcabile, sunt prezentate pe larg în *Subcapitolul 5.2*.

Un alt algoritm interesant este Elliptic Envelope, care pornind de la premisa că datele au repartiție gaussiană, încearcă să determine o frontieră elipsoidă care să conțină majoritatea datelor, definind astfel structura setului de date și putând să detecteze anomalii ca observații suficient de îndepărtate de forma de bază. Astfel, orice set aflat în afara elipsoidului este clasificat ca fiind o abatere de la normalitate [55]. Rezultatele obținute au fost mai bune în cazul acestei abordări, fiind și acestea prezentate în detaliu în *Subcapitolul 5.2*.

Algoritmul SVM este una dintre cele mai populare metode utilizate în clasificare – problema ce urmărește repartizarea unor obiecte necunoscute în categorii predefinite [56],

utilizând în etapa de antrenare date din toate clasele. Există însă și o variantă a acestei tehnici ce verifică apartenența unui obiect la o singură clasă, permițând absența exemplilor negative. Clasificarea One-Class (OC) presupune antrenarea modelului numai pe exemple pozitive și încercarea diferențierii clasei față de restul spațiului, un clasificator OC luând decizia de a accepta sau respinge un obiect de test în clasa pozitivă [57].

OC-SVM este unul dintre cele mai populare astfel de clasificatoare, fiind aplicat cu succes în probleme cu clase nebalansate și capabil să fixeze o suprafață de decizie închisă în jurul clasei pozitive ce înglobează majoritatea instanțelor din setul de date de antrenare [58]. Acesta a obținut printre cele mai bune rezultate, prezentate în detaliu în *Subcapitolul 5.2*, datorită acestora, algoritmul ajungând să fie utilizat și în sistemul de autentificare final.

Un alt algoritm foarte popular pentru detecția anomaliilor este Local Outlier Factor. Propus pentru prima dată în anii 2000 și inspirat de tehnicile de clustering bazat pe densitate (ca algoritmul DBSCAN, de pildă), a fost cel dintâi algoritm care pe lângă detecția anomaliilor, realizează și cuantificarea nivelului de abatere de la normal [59].

Metode de detecție a anomaliilor sunt de obicei grupate în două clase: globale și locale. Metodele globale iau în calcul întregul set de date de antrenare pentru determinarea factorului de abatere [60]. În cazul algoritmului LOF, acest factor de abatere este considerat a fi local în sensul considerării unei vecinătăți restrânse în jurul fiecărei instanțe, putând astfel determina anomalii ce nu ar putea fi observate în mod normal folosind metodele globale. LOF pornește de la premisa că instanțele anormale sunt mai izolate decât punctele obișnuite, evaluând factorul de abatere în funcție de distanța până la cei mai apropiați k vecini. Și rezultatele obținute de LOF pot fi consultate în *Subcapitolul 5.2*, acolo unde au fost descrise în detaliu.

În urma examinării algoritmilor, s-a ajuns la concluzia că, în ciuda acurateței la antrenare mai mici pentru OC-SVM decât pentru LOF, în privința acceptanței false, OC-SVM este mai potrivit pentru seturile mai mici de date, pentru această problemă, în timp ce pentru seturile de peste 100-120 de amprente, LOF obține rezultate mai bune, mai ales în cazul ratei de respingere falsă. Astfel, pe server, la momentul clasificării utilizator legitim versus impostor, pentru alegerea algoritmului de decizie, va fi luat în considerare numărul de amprente din baza de date – în cazul unui număr mai mic de date, va fi utilizat algoritmul *One-Class Support Vector Machines*, (sacrificând, poate, comoditatea utilizatorului pentru a crește securitatea), iar în caz contrar, se va folosi *Local Outlier Factor*.

4.4 Actualizarea șablonului

Comportamentul utilizatorului este inevitabil susceptibil la modificări în timp, astfel încât sistemul trebuie să fie flexibil și adaptabil la schimbări cauzate atât de factori psihologici cât și de alterări ale aptitudinilor și deprinderilor. Numită și procedură de învățare incrementală, reactualizarea șablonului este o etapă crucială în sistemele de autentificare continuă, care împiedică degradarea performanței, deoarece păstrarea amprentelor vechi și utilizarea exclusivă a acestora pentru antrenarea modelului ar putea cauza creșterea numărului de respingeri ale utilizatorului legitim.

Două dintre metodele populare pentru a combate această provocare sunt fereastra crescătoare (*growing window*) și fereastra glisantă (*moving window*) [61]. Fereastra crescătoare presupune adăugarea noilor amprente la șablonul (deja existent) ce determină comportamentul de referință al utilizatorului. Chiar dacă această metodă garantează toleranța la modificări în timp, faptul că numărul de amprente crește permanent va avea un impact asupra costurilor de stocare, dar și asupra timpului de antrenare și implicit, de decizie, fără ca amprente vechi să aducă vreun beneficiu întrucât, pe măsura trecerii timpului, acestea devin din ce în ce mai puțin reprezentative pentru comportamentul utilizatorului legitim.

Astfel ajungem la cea de-a doua metodă, fereastra glisantă. Aceasta presupune adăugarea amprente noi cu costul înlăturării celei mai vechi amprente deja existente, permițând astfel păstrarea unui număr constant de instanțe, care să reprezinte utilizatorul [61].

Am ales, așadar, datorită avantajelor oferite, metoda ferestrei glisante, cu câteva mențiuni – chiar dacă la antrenarea modelului (care se re-efectuează înainte de fiecare verificare a identității utilizatorului) nu sunt luate în considerare decât cele mai recente 750 de amprente ale utilizatorului, dacă există mai multe, acestea nu sunt șterse automat, ci păstrate pentru evaluările de tip audit IT, însă componenta server a sistemului permite administratorului ștergerea instanțelor vechi din baza de date.

4.5 Structura bazei de date

Tabelele care alcătuiesc baza de date asociată sistemului de autentificare au o arhitectură simplă, prezentată în *Figura 4.3*. Cheile primare sunt evidențiate prin utilizarea simbolului specific „#”.

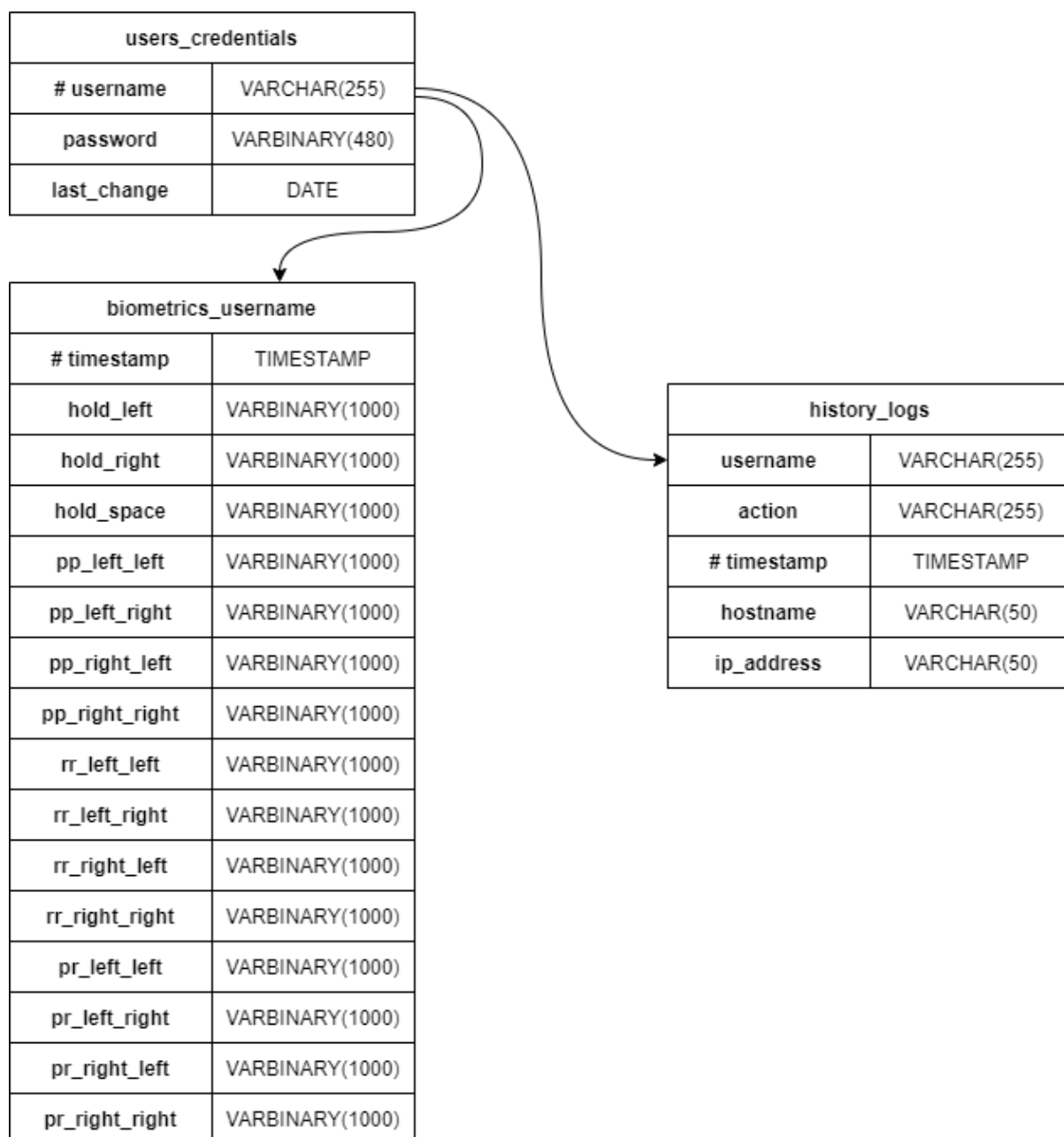


Figura 4.3: Schema bazei de date

În tabelul *users_credentials* sunt stocate credențialele de acces ale fiecărui utilizator (username-ul și parola – acestea din urmă fiindu-i aplicată o funcție de *hashing* cu *salt*, fiind utilizat algoritmul *bcrypt*, după cum a fost menționat în *Secțiunea 3.4*, înainte de a fi inserată în tabel), dar și data ultimei modificări de parolă (pentru a putea atenționa utilizatorul în punctul în care este recomandată reactualizarea acesteia).

Pentru fiecare utilizator înregistrat, este creată o nouă tabelă în care îi sunt salvate amprente (în format criptat, alcătuite din caracteristicile prezentate în *Subcapitolul 4.2*), împreună cu marcajul de timp corespunzător colectării acestor amprente. Stocarea amprentelor

fiecărui utilizator într-o tabelă distinctă este motivată de dorința de eficiență. În cazul unui număr mare de utilizatori, parcurgerea tuturor amprentelor și extragerea celor corespunzătoare unei anumite persoane pentru verificarea identității utilizatorului respectiv s-ar putea dovedi extrem de consumatoare de timp.

Tabelul *history_logs* permite stocarea log-urilor, compuse din numele utilizatorului, acțiunea întreprinsă sau generată de acesta, amprenta de timp a acțiunii, *hostname*-ul și adresa IP a stației de pe care a fost conectat utilizatorul în timpul acțiunii.

5. TESTARE ȘI VALIDARE

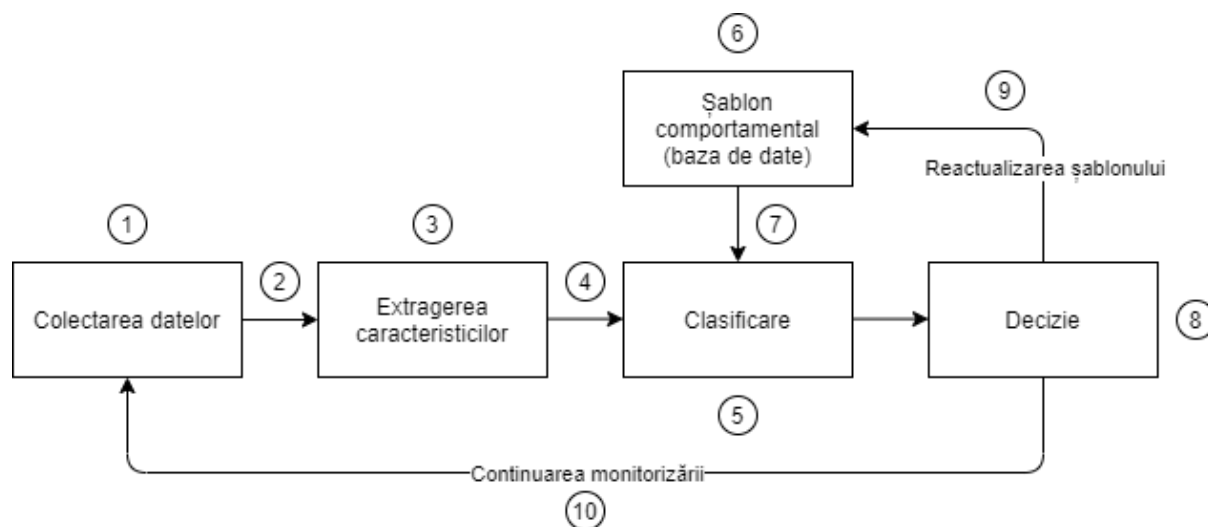


Figura 5.1: Puncte vulnerabile într-un sistem de autentificare continuă de tip biometric (figură adaptată și extinsă după [62])

Este cert faptul că în proiectarea și implementarea unui sistem de autentificare, fie că acesta are la bază strategii statice sau continue, cel mai important element este securitatea. De aceea, în testarea și validarea unui astfel de sistem, succesul este puternic influențat de rezistența la anumite tipuri de amenințări și vulnerabilități, comune soluțiilor software din aceeași categorie.

Conform lucrării [62], punctele vulnerabile pot fi împărțite în două clase: clasa celor referitoare la arhitectura sistemului de autentificare, respectiv clasa celor privitoare la performanță și la calitatea amprentelor extrase.

5.1 Analiza arhitecturii sistemului

Din punct de vedere al arhitecturii, punctele vulnerabile ale unui sistem tipic de autentificare continuă biometrică sunt prezentate în *Figura 5.1*. Un sistem de autentificare biometric general ar prezenta aceleași nivele de sensibilitate, cu mențiunea că punctele 9 și 10 nu ar exista într-o aplicație din această categorie.

La nivelul celui dintâi punct, reprezentat de modulul de colectare al datelor, respectiv de senzorul de input, o persoană malițioasă ar putea, de exemplu, să apeleze la tehnici de *spoofing*, fie prezentând senzorului date false (în cazul unui sistem biometric bazat pe amprentă, ar putea fi utilizată o proteză specială care să imite perfect amprenta victimei; în cazul unui sistem bazat pe voce, ar putea fi utilizate înregistrări audio ale persoanei respective,

etc.), fie exploatând similitudinile datelor proprii cu cele ale victimei (situația unor gemeni identici în cazul sistemelor bazate pe recunoașterea facială sau verificările de ADN, dar și exploatarea dificultății deosebirii geometriei trăsăturilor fețelor pentru anumite naționalități – în China, de pildă, procesul de recunoaștere și autentificare facială este mult mai dificil de securizat întrucât trăsăturile se pliază pe geometrii asemănătoare [63]).

În cazul sistemului propus, bazat pe biometria tastării, pentru a păcăli sistemul, ar trebui fie ca atacatorul să-și modifice propriul comportament de așa manieră încât să semene cât mai mult cu șablonul victimei, lucru aproape imposibil întrucât modul în care tastăm este extrem de greu de imitat. Totuși, sistemul ar putea avea de suferit la capitolul securitate în cazul apelării la metode de generare artificială a evenimentelor de apăsare, respectiv eliberare a tastelor, dar pentru acest lucru ar fi necesar accesul la amprenta comportamentală a utilizatorului de imitat – greu de obținut în cazul prezenței unor soluții de protecție *anti-malware* de bază la nivelul *endpoint*-ului.

La nivelul majorității legăturilor dintre module, o problemă de securitate foarte gravă ar putea fi generată de interceptarea informațiilor de pe canalul de comunicare. O astfel de vulnerabilitate ar putea permite inițierea cu succes a atacurilor de replicare (*replay attacks*), amprente legitime ale utilizatorului putând fi transmise mai târziu pentru ca o persoană malițioasă să obțină acces neautorizat. Această problemă este însă contracarată în sistemul propus cu ajutorul utilizării protocolul TLS v1.3 și a certificatelor de securitate.

O problemă activă rămâne însă posibilitatea unui atac de tip *Denial of Service*, care ar putea tăia legătura dintre componente, făcând sistemul indisponibil pentru utilizatorii săi. Cel mai mare impact asupra sistemului l-ar avea un atac de acest tip inițiat cu scopul de a întrerupe încheierea înrolării complete. Dacă sistemul nu reușește să colecteze numărul minim de amprente necesar identificării utilizatorului, acesta nu mai apelează la modulul biometric de autentificare, conectarea fiind posibilă cu ajutorul perechii nume de utilizator și parolă, transformându-se într-un sistem static, ne-biometric, vulnerabil până la colectarea tuturor amprentelor (care continuă totuși în fundal după conectarea tradițională).

Puncte vulnerabile în cadrul sistemului ar putea fi, în general, și modulele software de extragere, clasificare și decizie (3, 5 și 8), premergător integrării în rețea existând riscul ca aceste componente să fie înlocuite de programe de tip *trojan horse*, care să funcționeze conform unor reguli definite de o terță malițioasă. Pentru a preveni astfel de acțiuni este necesară verificarea și monitorizarea integrității fișierelor aplicației.

Alte probleme pot apărea la nivelul bazei de date (punctul 6), în cazul în care un atacator alterează sau șterge amprente stocate sau chiar reușește să fure datele cu totul. Chiar și în cazul în care baza de date asociată sistemului propus (atât fișierul cu parole, cât și cel cu amprente) ar ajunge la terțe malițioase, acest lucru nu ar afecta informațiile comportamentale ale utilizatorilor înrolați, amprentele acestora fiind stocate în mod criptat.

Întrucât parola utilizată la criptare trebuie să respecte anumite restricții precum dimensiune mare, utilizarea majusculilor, minusculilor, a caracterelor speciale și a cifrelor, aflarea acesteia și implicit, decriptarea amprentelor, este destul de dificilă.

Alterarea și ștergerea datelor poate fi împiedicată însă numai de protecția bazei de date la nivelul infrastructurii din care aceasta face parte.

5.2 Analiza performanței sistemului

Așa cum a fost expus și în *Subcapitolul 4.3*, performanța unui sistem biometric nu poate fi măsurată în același mod ca în cazul unui sistem de autentificare tradițional, în care între parola introdusă de utilizator la înregistrare și cea utilizată la reconectare trebuie să existe o potrivire perfectă. În cazul autentificării biometrice, sunt posibile patru tipuri de manifestări:

- sistemul poate să accepte utilizatorul legitim;
- sistemul poate să respingă utilizatorul nelegitim;
- sistemul poate să accepte utilizatorul nelegitim;
- sistemul poate să respingă utilizatorul legitim,

ultimele două situații fiind generatoare de erori de tip acceptanță falsă (și contribuind la creșterea FAR), respectiv de tip respingere falsă (și contribuind la creșterea FRR). Formulele pentru mărimile FAR și FRR au fost deja prezentate în *Capitolul 4*, acolo unde a fost și motivată necesitatea de a apela la câțiva voluntari pentru a măsura performanța sistemului.

Voluntarii selectați au avut vârste cuprinse între 20 și 46 de ani, familiaritatea acestora cu domeniul tehnic (înțelegând prin acesta mai mult experiența în dactilografie) putând fi clasificată ca fiind medie sau ridicată, în timp ce încrederea lor în sistemele biometrice de autentificare a fost autoevaluată ca fiind de la scăzută la ridicată. Toți voluntarii au utilizat aceeași stație la interacționarea cu sistemul, nefiind întrebuințate soluții software de tip *remote access/ control* întrucât s-a observat faptul că toate tehnologiile testate din această categorie introduceau întârzieri semnificative și neconsecvente, având un impact negativ asupra calității amprentelor colectate (această observație afectând totodată și numărul de voluntari care ar fi putut participa la faza de testare a algoritmilor, respectiv sistemului).

Totodată, voluntarii au fost încurajați să păstreze un comportament cât mai apropiat de cel obișnuit în timpul culegerii amprentelor, folosind editoare de text pentru a modifica documente, comunicând folosind rețele de socializare, căutând informații cu ajutorul motoarelor de căutare, alternând activitățile de tastare cu pauze cât mai naturale (cu distribuții și durate aleatorii).

Algoritmii de clustering DBSCAN și K-means nu s-au potrivit foarte bine problemei noastre, neobținând rezultate impresionante: o acuratețe la antrenare sub 50% și un *FRR* mai mare de 70% (dar un *FAR* aproape de 1%) în cazul lui DBSCAN, respectiv o acuratețe la antrenare de 75.34%, un *FRR* de 24.57% și un *FAR* mai mic de 3%, în cazul lui K-means.

În cazul algoritmului Elliptic Envelope, rezultatele s-au arătat a fi ușor îmbunătățite pentru cei șase voluntari, generând o medie de acuratețe la antrenare de 85.58%, o rată de rejecție falsă de 10.58% și o rată de acceptanță falsă de 12.91%.

Test-A	Test-B	Test-C	Test-D	Test-E	Test-F	
0.894	0.870	0.890	0.869	0.876	0.736	<i>TA</i>
0.150	0.100	0.000	0.100	0.000	0.285	<i>FRR</i>
0.056	0.000	0.053	0.116	0.000	0.550	<i>FAR</i>

Tabelul 5.1: Rezultatele algoritmului Elliptic Envelope pentru cei șase voluntari

Setul de date Test-F, corespunzător celui de-al șasea voluntar, a fost cel mai puțin amplu dintre teste, fiind alcătuit din cel mai mic număr de amprente, și declanșând o scădere drastică a performanței totale, demonstrând astfel că algoritmul nu se comportă foarte bine pe seturi mici. Această ipoteză a fost confirmată și în cazul evaluării performanței algoritmului pe mini-seturile de date extrase din colecția de „136 Million Keystrokes”, generând rezultate nesatisfăcătoare: în ciuda unui *FAR* de 1.911%, modelul a avut o acuratețe la antrenare de numai 6.207%.

Printre cele mai bune rezultate obținute au fost acelea produse de evaluarea algoritmului OC-SVM, acestea putând fi observate în *Tabelul 5.2* – un *TA* mediu de 92.15%, un *FRR* de 3.666% și un *FAR* de 1.183%.

Test-A	Test-B	Test-C	Test-D	Test-E	Test-F	
0.918	0.946	0.918	0.946	0.946	0.855	<i>TA</i>
0.050	0.050	0.010	0.100	0.000	0.010	<i>FRR</i>
0.027	0.010	0.010	0.006	0.010	0.008	<i>FAR</i>

Tabelul 5.2: Rezultatele algoritmului One-Class Support Vector Machine pentru cei șase voluntari

În cazul evaluării performanței pentru mini-seturile de date, acuratețea la antrenarea a fost de 71.531%, iar *FAR* egal cu 13.405%, mult mai bune decât în cazul Elliptic Envelope.

Rezultate evaluării algoritmului Local Outlier Factor, pentru cei șase voluntari, sunt prezentate în *Tabelul 5.3*, cu un *TA* mediu de 97.73%, un *FRR* egal cu 0 și un *FAR* de 1.08%. În cazul seturilor de date mai mici, algoritmul a reușit să atingă o acuratețe la antrenare de 91.605% și o rată de acceptanță falsă de 29.275%.

Test-A	Test-B	Test-C	Test-D	Test-E	Test-F	
0.988	0.992	0.981	0.992	0.984	0.927	<i>TA</i>
0.000	0.000	0.000	0.000	0.000	0.000	<i>FRR</i>
0.008	0.000	0.017	0.006	0.000	0.034	<i>FAR</i>

Tabelul 5.3: Rezultatele algoritmului Local Outlier Factor pentru cei șase voluntari

Dat fiind faptul că în clasificarea din sistemul final s-a luat decizia de a se utiliza o combinație între algoritmi OC-SVM și LOF, în funcție de numărul de amprente disponibile pentru utilizatorul respectiv în baza de date la momentul verificării, așa cum a fost expus în *Subcapitolul 4.3*, mărimile *FAR* și *FRR* sunt dificil de estimat într-un mediu real, întrucât depinde de timpul petrecut de utilizatori în sistemul respectiv și de cantitatea de informații culeasă de la tastatură, dar presupunând că într-un scenariu concret utilizatorii ar avea cel puțin 100 de amprente stocate, rezultatele vor fi apropiate de cele obținute de algoritmul Local Outlier Factor.

Tot la capitolul performanța sistemului ar trebui punctate câteva observații sau presupuneri confirmate la testare, în legătură cu autentificarea bazată pe comportamentul utilizatorului, deoarece acestea reprezintă importante dezavantaje ale acestui de tip de soluții software – modul în care tastează utilizatorul este extrem de variabil, depinzând de starea

acestui de spirit, de cea de sănătate, dar și de atenția pe care o îndreaptă asupra dactilografierii (dacă se încearcă autentificarea în timpul unei conversații la telefon, de pildă, sistemul ar putea da greș) [8].

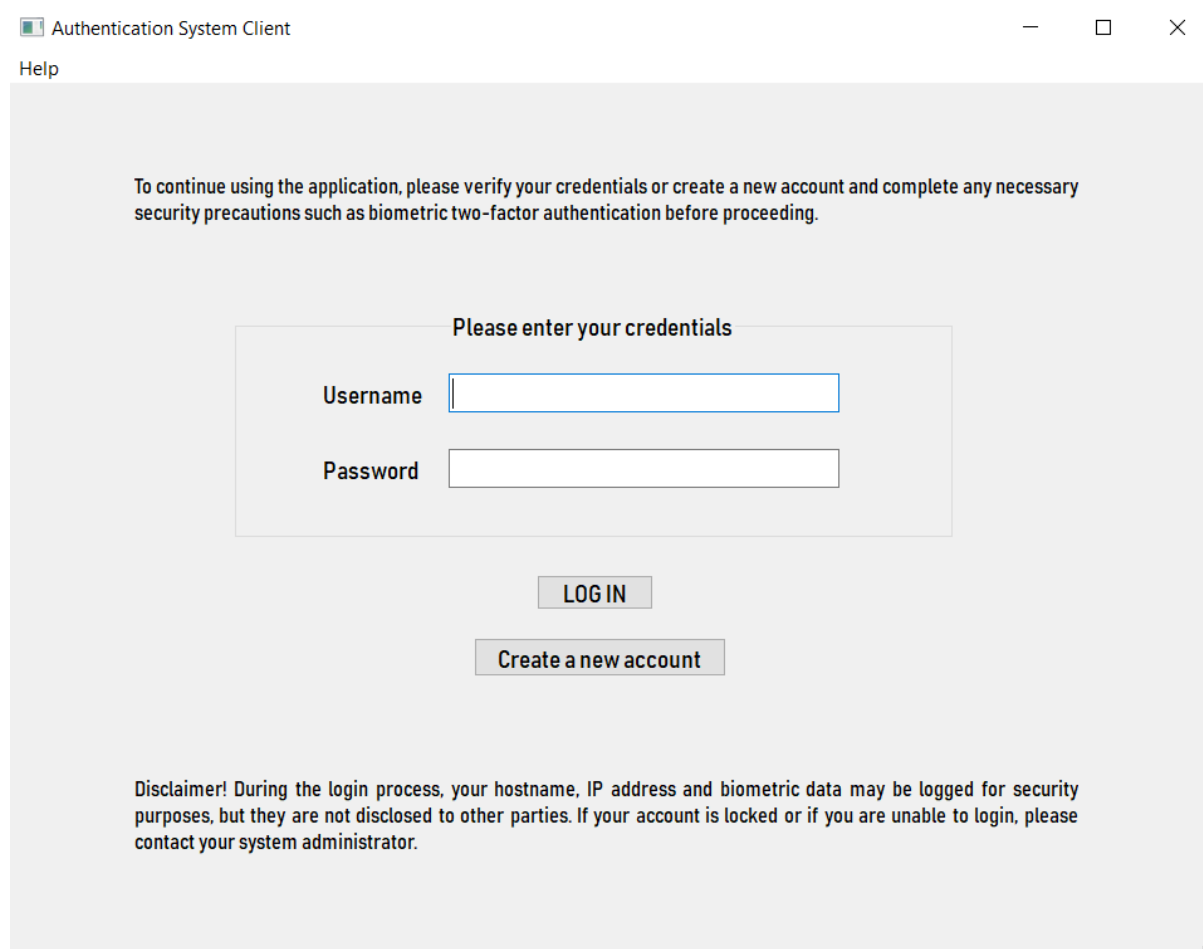
Deoarece sistemul propus este unul care monitorizează și înregistrează continuu comportamentul utilizatorilor săi, problemele sunt totuși ușor ameliorate în comparație cu cazul static, iar dacă schimbările comportamentale nu sunt atât de bruște, sistemul ar putea să recunoască în continuare, de cele mai multe ori, identitatea utilizatorului.

În cadrul testării propriu-zise efectuate de voluntari asupra sistemului, au fost observate și câteva probleme, precum o greșeală de proiectare a funcționalității de schimbare a parolei. Aceasta avea ca efect transmiterea unei notificări în legătură cu respingerea utilizatorului, chiar dacă aceasta era legitim, deoarece sistemul se bifurca în următoarele direcții: pe de o parte, continua verificarea cu ajutorul credențialelor noi (dacă utilizatorul se reconecta după modificare), iar pe de altă parte, continua verificarea și folosind parola veche (declanșând astfel clasificarea utilizatorului ca fiind nelegitim, și implicit apariția alertei). Această eroare a fost însă remediată prin reutilizarea funcției care asigura deconectarea completă a utilizatorului după ce aceasta își modifica parola de acces.

6. UTILIZAREA SISTEMULUI

6.1 Manual de utilizare a componentei client

Înrolarea



The screenshot shows a window titled "Authentication System Client" with standard window controls (minimize, maximize, close) in the top right corner. A "Help" link is located in the top left. The main content area has a light gray background and contains the following elements:

- A message: "To continue using the application, please verify your credentials or create a new account and complete any necessary security precautions such as biometric two-factor authentication before proceeding."
- A section titled "Please enter your credentials" containing:
 - A "Username" label next to a text input field.
 - A "Password" label next to a text input field.
- A "LOGIN" button.
- A "Create a new account" button.
- A disclaimer at the bottom: "Disclaimer! During the login process, your hostname, IP address and biometric data may be logged for security purposes, but they are not disclosed to other parties. If your account is locked or if you are unable to login, please contact your system administrator."

Figura 6.1: Pagina de conectare a utilizatorului

În momentul pornirii componentei client, își face apariția interfața grafică, minimalistă, pentru reducerea timpului pierdut, dar și foarte prietenoasă, făcând mai ușoară interacțiunea utilizatorului cu sistemul.

În partea de jos a ferestrei pot fi observate câteva informații despre datele colectate în timpul conectării, respectiv înrolării în aplicație, din motive de securitate: *hostname*-ul, adresa IP și informațiile biometrice. *Hostname*-ul și IP-ul vor fi utilizate pentru recunoașterea dispozitivului, iar informațiile biometrice vor juca un rol extrem de important pentru confirmarea identității utilizatorului conectat.

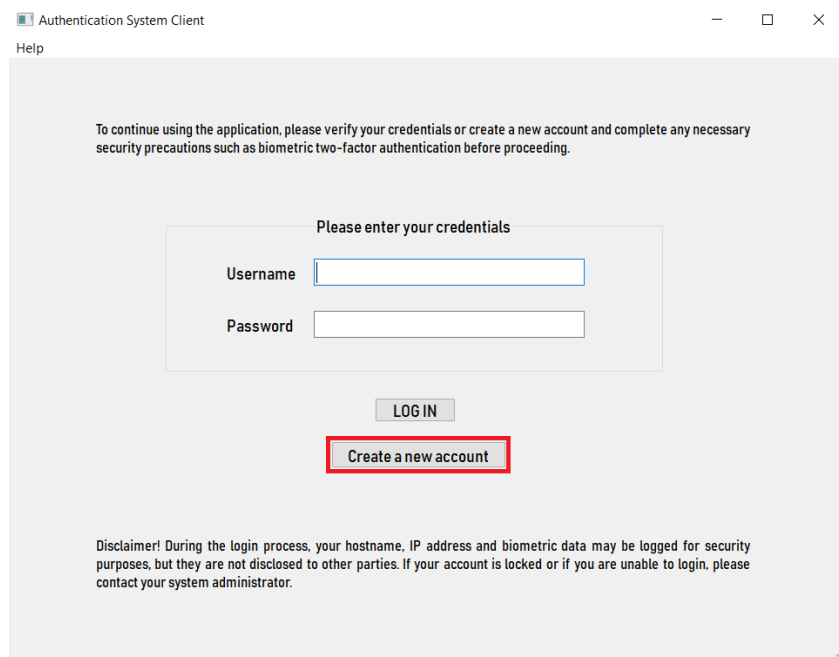


Figura 6.2: Butonul de creare a unui cont nou

Pentru înrolarea în sistem este necesară crearea unui cont nou, în cazul în care utilizatorul nu are încă unul, acest lucru putând fi realizat cu ajutorul butonului „Create a new account”, evidențiat în Figura 6.2.

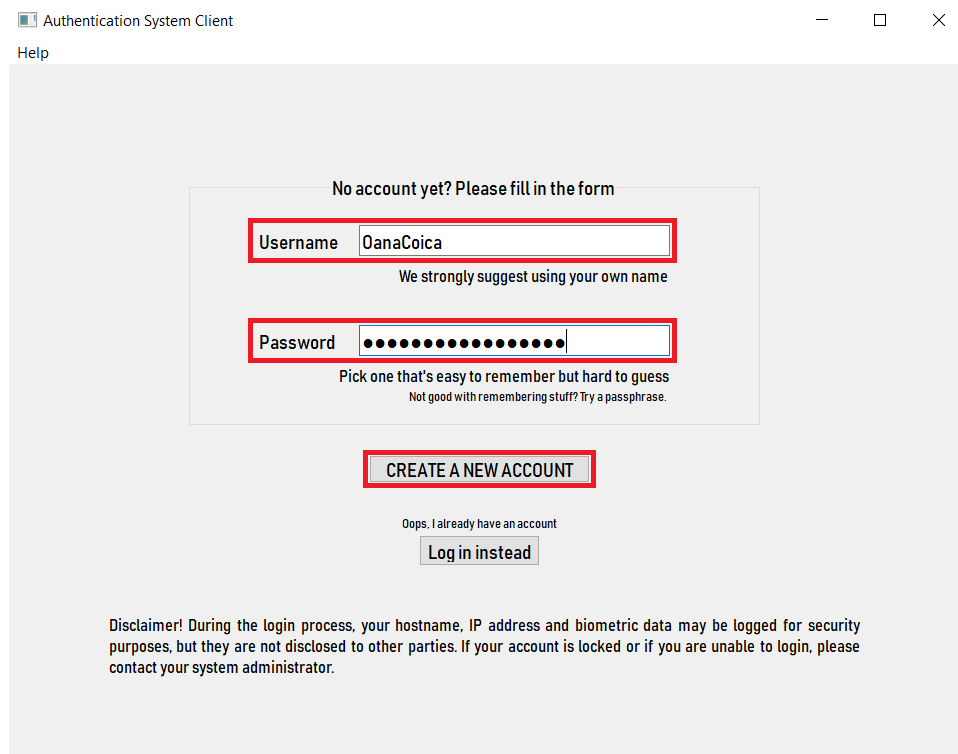


Figura 6.3: Formularul de înrolare

Apare un formular de înregistrare (*Figura 6.3*), informațiile cerute fiind un nume de conectare, folosit pentru identificare (sugestia noastră e ca acesta să coincidă cu numele real al persoanei), de minim 5 caractere, unic, și o parolă, de minim 10 caractere, utilizată pentru prima fază a autentificării. În urma completării formularului, prin apăsarea butonului „*CREATE A NEW ACCOUNT*”, cererea este transmisă către server, utilizatorul fiind înștiințat în cazul în care apare vreo neregulă (dacă numele este deja luat, dacă datele nu au dimensiunile minime obligatorii, etc.).

Următoarea etapă a înrolării o reprezintă „amprentarea” (*Figura 6.4*) – monitorizarea inițială a utilizatorului pentru a determina un comportament de referință, ce va fi necesar pentru faza a doua a autentificării. Observarea tiparului de a scrie va fi observată și în afara ferestrei aplicației, în fundal, astfel încât utilizatorul își poate continua activitățile obișnuite în timp ce sistemul culege datele de care are nevoie.

După încheierea colectării datelor comportamentale inițiale, are loc deconectarea, utilizatorul fiind trimis din nou în pagina observată în *Figura 6.1*.

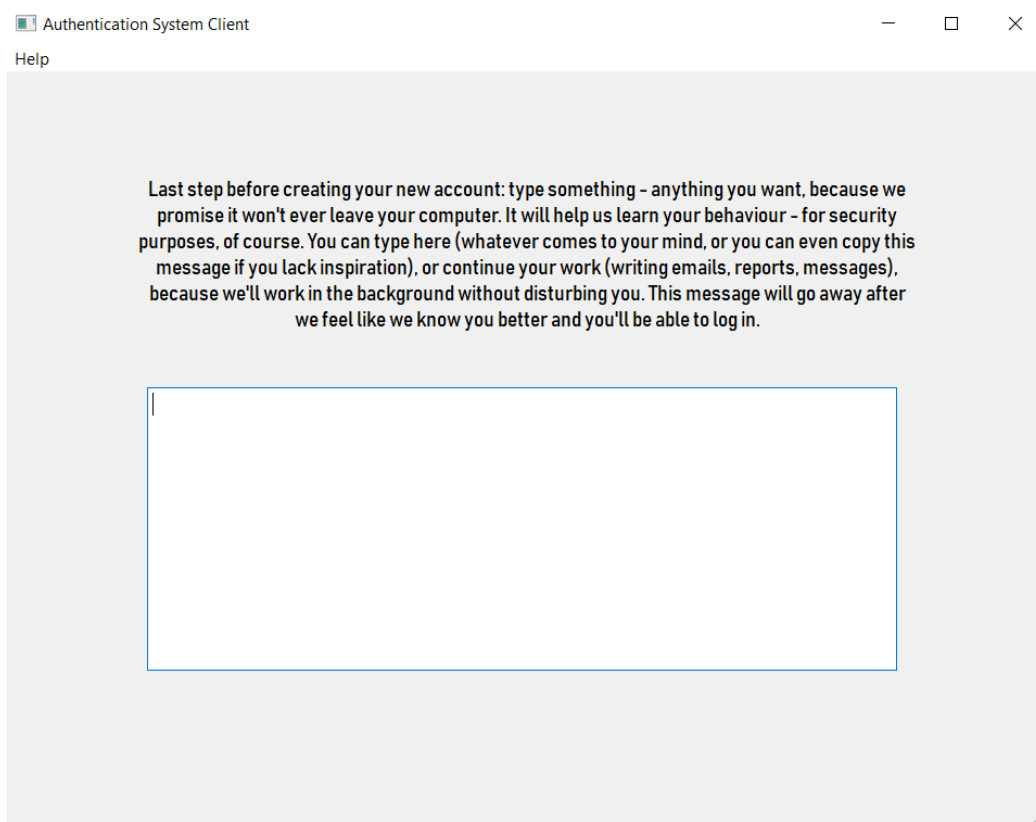


Figura 6.4: Momentul monitorizării inițiale

Autentificarea

Authentication System Client

Help

To continue using the application, please verify your credentials or create a new account and complete any necessary security precautions such as biometric two-factor authentication before proceeding.

Please enter your credentials

Username OanaCoica

Password ●●●●●●●●●●●●●●●●

LOG IN

Create a new account

Disclaimer! During the login process, your hostname, IP address and biometric data may be logged for security purposes, but they are not disclosed to other parties. If your account is locked or if you are unable to login, please contact your system administrator.

Figura 6.5: Faza I de autentificare

În pagina de conectare, după completarea câmpurilor credențialelor, utilizatorul poate trimite cererea de autentificare apăsând butonul „LOG IN” (*Figura 6.5*). În cazul în care datele introduse nu coincid cu cele stocate de către server, utilizatorul va fi înștiințat printr-o notificare de tip pop-up și va avea ocazia să mai încerce realizarea completării datelor. Totuși, din motive de securitate, înștiințarea nu va cuprinde informații despre componenta eronată, ci va considera perechea de credențiale ca pe un întreg.

În cazul în care datele de conectare sunt corecte, utilizatorului îi vor fi aduse la cunoștință declanșarea fazei secunde și începerea monitorizării comportamentului său (*Figura 6.6*).

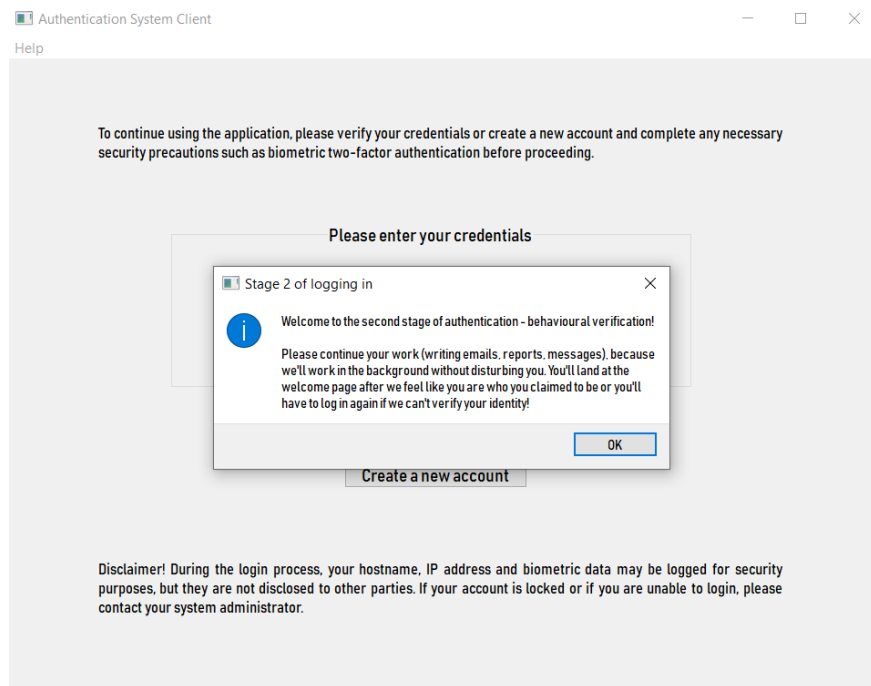


Figura 6.6: Faza a II-a de autentificare

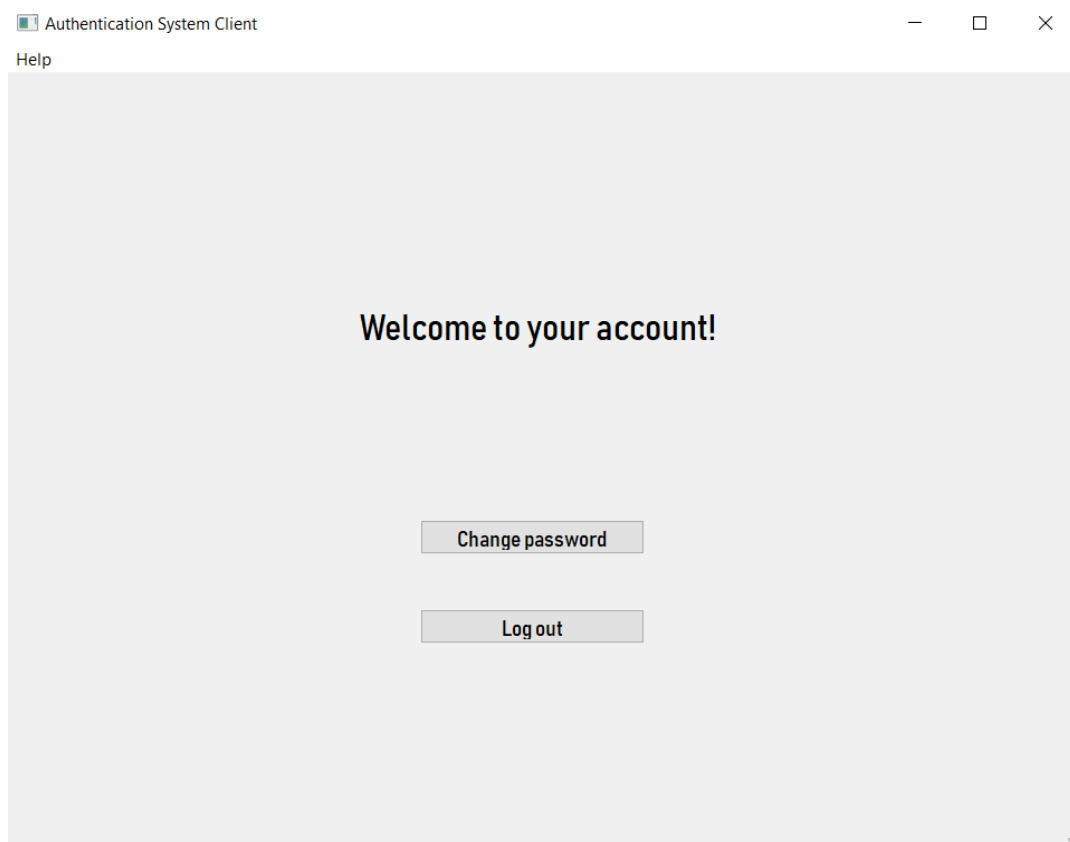
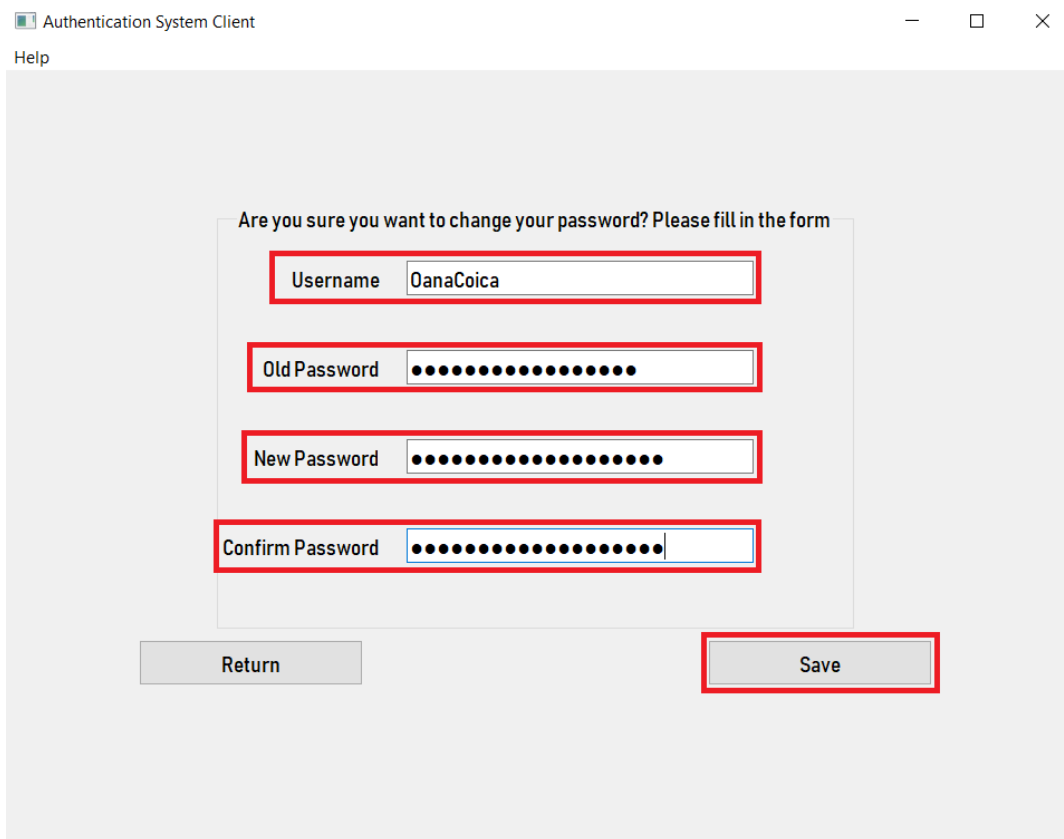


Figura 6.7: Pagina ce apare în situația autentificării complete reușite

După ce prima amprentă comportamentală ajunge la server, în cazul recunoașterii cu succes, ce determină trecerea de cea de-a doua etapă a autentificării, un mesaj de bun venit va apărea pe ecran (*Figura 6.7*), iar utilizatorul va avea acces la funcționalitatea de schimbare a parolei. Monitorizarea va continua în fundal, iar în momentul în care o schimbare de comportament va fi constatată, sistemul va presupune că aceasta a fost provocată de înlocuirea utilizatorului legitim cu un impostor, declanșând deconectarea imediată.

Schimbarea parolei

Pentru modificarea parolei, utilizatorul trebuie să fi trecut de faza a doua de autentificare, acest lucru însemnând aflarea în situația prezentată în *Figura 6.7*. Utilizând butonul „*Change password*”, completând formularul afișat (ilustrat în *Figura 6.8*) și trimițând cererea către server prin apăsarea butonului „*Save*”, în cazul în care credențialele introduse sunt corespunzătoare, iar parola nouă respectă regulile valabile și pentru crearea unui nou cont, schimbarea parolei este finalizată.



The screenshot shows a window titled "Authentication System Client" with a "Help" link. The main content area displays a form with the heading "Are you sure you want to change your password? Please fill in the form". The form includes four input fields, each with a red border: "Username" (containing "OanaCoica"), "Old Password", "New Password", and "Confirm Password" (all masked with dots). At the bottom of the form are two buttons: "Return" and "Save". The "Save" button is highlighted with a red border.

Figura 6.8: Formularul corespunzător schimbării parolei

După această etapă, utilizatorul este din nou solicitat să efectueze conectarea, trecând prin cele două etape de autentificare, respectiv pe de o parte, introducerea corectă a credențialelor noi, iar pe de altă parte, recunoașterea comportamentală.

Excepții, erori, probleme suplimentare

În cadrul oricărui sistem, mai pot apărea și probleme sau situații neprevăzute. În *Figura 6.9* sunt ilustrate mesajele de tip pop-up care pot pune uneori utilizatorul într-o stare de confuzie.

În cazul celui dintâi mesaj, cauza apariției sale este faptul că utilizatorul a introdus credențiale eronate, această situație apărând de obicei din cauza unor mici greșeli de tastare. Însă, în cazul în care acesta este convins că numele și parola introduse ar trebui să fie cele corecte, o altă cauză posibilă ar putea fi aceea că administratorul a făcut modificări asupra datelor de conectare – fie din cauza unor suspiciuni de fraudă (din extern sau intern), fie din dorința de prevenție. Astfel, metoda de soluționare a acestei probleme este contactarea administratorului, el având drepturile de re-modificare a credențialelor oricărui utilizator. Similar, și problema uitării parolei va fi rezolvată în același mod.

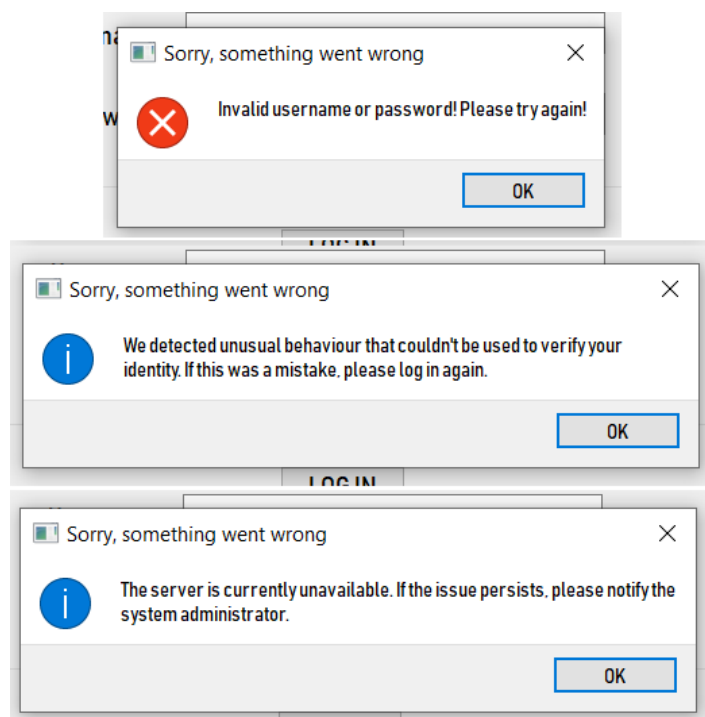


Figura 6.9: Erori și atenționări posibile

Cel de-al doilea mesaj este unul destul de obișnuit, mai ales în situația în care utilizatorul nu a interacționat suficient de mult cu sistemul, astfel încât acesta să îi învețe comportamentul pe atât de bine încât să nu mai existe fenomenul nerecunoașterii identității. În cazul în care acest mesaj apare în mod frecvent și utilizatorul, deși legitim, nu se mai poate conecta, indiferent de momentul și starea de spirit din momentul tentativei de autentificare, mai ales după un timp îndelungat de nefolosire a sistemului, acest lucru se poate întâmpla din cauza schimbărilor de comportament mari care au determinat „expirarea” amprentelor comportamentale de la momentul înregistrării. Soluția stă în continuare în mâinile administratorului, care poate șterge amprente de timp învechite.

Al treilea mesaj este cauzat de tentativele eșuate de conectare la server, ce ar putea fi provocate de ferestrele de mentenanță sau de restricțiile sistemelor de tip firewall (fie local, pe stația utilizatorului, fie la nivel de rețea). Utilizatorul este sfătuit ca în cazul în care situația persistă, să contacteze administratorul de rețea.

6.2 Manual de utilizare a componentei server

Pentru ca sistemul, alcătuit din aplicația client și aplicația server, să relaționeze în mod corect, este necesară pornirea unei instanțe de MySQL, existând mai multe moduri de a realiza acest lucru, prezentarea lor neîncadrându-se în obiectivul acestui manual. Totodată, este necesară prezența certificatelor de securitate, dar și completarea corectă a informațiilor despre modul de realizare a conexiunii cu baza de date.

În urma pornirii aplicației corespunzătoare componentei server, își va face apariția interfața grafică. În partea superioară a ferestrei, informațiile sunt organizate în două file (tab-uri) distincte, asemănătoare celor întâlnite în majoritatea browserelor. Prima filă, cea activă în mod implicit, prezintă istoricul acțiunilor efectuate de sistem, dar și de utilizatorii acestuia – precum crearea unui cont nou, conectarea la un cont existent, tentativa de conectare cu credențiale eronate, eșuarea completării fazei a doua de autentificare, schimbarea parolei (sau numelui de utilizator al) unui cont, ștergerea amprentelor vechi, ștergerea log-urilor vechi, reîncărcarea informațiilor, ștergerea unui cont, etc, aceste informații fiind însoțite de numele, *hostname*-ul și adresa IP ale celui care a întreprins respectiva acțiune (*System* este numele utilizat de sistemul în sine, dar totodată și de administratorul acestuia) și amprenta de timp corespunzătoare evenimentului, aceste detalii fiind utile în cazul unor evaluări de tip audit, dar și pentru analiza utilizării sistemului sau a unei eventuale tentative de fraudă.

Authentication System Manager

Help

History Logs Users List

	Username	Action	Timestamp	Hostname	IP Address
1	OanaCoica	logged in using password	2020-05-26 21:30:10	DESKTOP-F7C58AV	192.168.205.225
2	System	fetchd information about history logs	2020-05-26 21:29:59	Authentication System Manager	
3	System	fetchd information about history logs	2020-05-26 21:29:59	Authentication System Manager	
4	TestUser247	created a new account	2020-05-26 21:27:20	TestUser247's computer	192.168.94.247
5	TestUser246	created a new account	2020-05-26 21:27:12	TestUser246's computer	192.168.95.246
6	TestUser245	created a new account	2020-05-26 21:27:06	TestUser245's computer	192.168.176.245
7	TestUser244	created a new account	2020-05-26 21:27:00	TestUser244's computer	192.168.216.244
8	TestUser243	created a new account	2020-05-26 21:26:53	TestUser243's computer	192.168.190.243
9	TestUser242	created a new account	2020-05-26 21:26:47	TestUser242's computer	192.168.11.242
10	TestUser241	created a new account	2020-05-26 21:26:42	TestUser241's computer	192.168.104.241
11	TestUser240	created a new account	2020-05-26 21:26:36	TestUser240's computer	192.168.246.240
12	TestUser239	created a new account	2020-05-26 21:26:31	TestUser239's computer	192.168.249.239

Reload data

Delete old authentication patterns Delete old logs

Figura 6.10: Fila implicită a aplicației componente server, corespunzătoare înregistrărilor evenimentelor

Cea de-a doua filă a sistemului ilustrează lista utilizatorilor sistemului. La selectarea unui utilizator din listă, în partea din dreapta apar log-urile corespunzătoare acestuia. Sub tabelul corespunzător acestor înregistrări sunt prezente trei butoane: „*Change username*” – pentru schimbarea unui nume de utilizator, „*Change password*” – pentru resetarea forțată a unei parole (cu prețul pierderii datelor comportamentale corespunzătoare utilizatorului) și „*Delete user*” – pentru ștergerea utilizatorului selectat.

Butoanele din partea inferioară a ferestrei, prezente indiferent de fila selectată, sunt cele de reactualizare a datelor (pentru afișarea acestora în aplicație), de ștergere a amprentelor comportamentale și a log-urilor vechi. În cazul amprentelor, acestea sunt considerate învechite în cazul în care au trecut 60 de zile de la colectarea lor sau dacă nu fac parte din cele mai recente 750. Log-urile sunt considerate vechi dacă sunt mai vechi de doi ani.

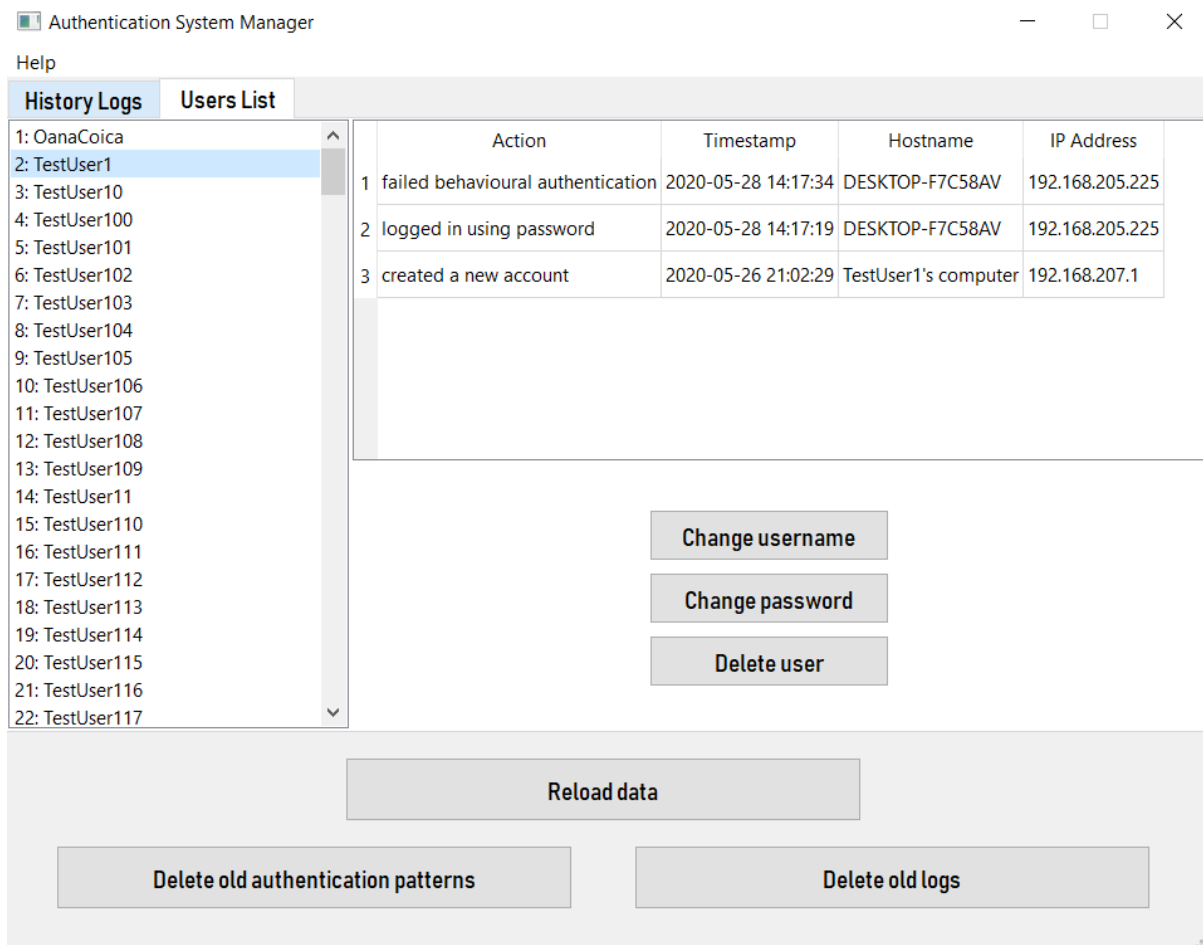


Figura 6.11: Cea de-a doua filă a aplicației componentei server, corespunzătoare listei de utilizatori

7. CONCLUZII ȘI DEZVOLTĂRI ULTERIOARE

Implementarea și testarea sistemului de autentificare propus au contribuit la extinderea cunoștințelor despre anumite noțiuni, protocoale, tehnologii și arhitecturi conceptuale. A fost studiată dinamica tastării ca modalitate de verificare a identității, experimentându-se cu noi abordări de extragere a caracteristicilor și construirii de amprente de referință corespunzătoare utilizatorilor, precum și cu algoritmi de învățare automată.

Totodată, a fost testat sistemul implementat, studiindu-se vulnerabilitățile posibile ale sistemelor de autentificare și eliminându-se acestea prin anumite decizii de proiectare sau oferindu-se idei de atenuare.

În urma testelor realizate au fost observate sau confirmate atât avantaje, cât și dezavantaje ale utilizării modalității de tastare ca indice biometric pentru autentificarea utilizatorilor. Printre avantaje se numără transparența, costurile reduse, absența nevoii de a apela la senzori suplimentari, caracterul non-invaziv și greutatea falsificării amprentelor. Dezavantajele sunt cele specifice tuturor indicilor comportamentali: instabilitatea în timp, dependența de starea utilizatorilor, dar și compromiterea permanentă în cazul în care amprente sunt furate sau dezvăluite.

Performanța sistemului propus ar putea fi îmbunătățită în urma testării acestuia pe mai mulți utilizatori, pentru perioade mai îndelungate, putând astfel evalua modul în care evoluează soluția software într-un plan mai apropiat de cel real, pe parcursul timpului. Totodată, la verificarea identității utilizatorilor ar putea fi luate în considerare și alte tipuri de indici biometrici, precum dispozitivul utilizat, momentul de timp la care se încearcă autentificarea, mișcărilor de mouse sau recunoașterea facială.

Cea din urmă modalitate s-ar putea dovedi extrem de utilă la identificarea persoanelor malițioase care încearcă să obțină acces neautorizat, dar acest modul suplimentar ar fi dificil de acceptat de către utilizatori dacă ar fi implementat ca metodă suplimentară de autentificare continuă, declanșând serioase îngrijorări cu privire la pierderea dreptului la intimitate.

În cadrul amprentării utilizatorilor ar putea fi utilizate în viitor și tehnici de stilometrie (*stylometry*), ce constau în cuantificarea elementelor de stil ale scrierii pentru a putea crea un șablon corespunzător fiecărui autor, printre aceste elemente de stil numărându-se atât caracteristici lexicale (de pildă, frecvența caracterelor sau cuvintelor, lungimea medie a unei propoziții, distribuțiile lungimilor cuvintelor, erorile lexicale, varietatea vocabularului activ),

cât și sintactice (utilizarea semnelor de punctuație și a caracterelor speciale) sau semantice (frecvența utilizării sinonimelor, antonimelor, polisemantismului) [6].

Poate cea mai importantă direcție de cercetare viitoare ar putea fi biometricele revocabile (*cancellable biometrics*), necesare pentru a proteja intimitatea utilizatorilor. Acest concept se referă la generarea de șabloane de referință înlocuibile sau anulabile, unice pentru fiecare aplicație și utilizator. Astfel, în bazele de date vor fi stocate versiuni transformate ale amprentelor, în locul versiunilor reale, permițând modelarea mai multor șabloane asociate unui singur set de date biometrice. Această conceptualizare se bazează pe trei principii:

- diversitate (șabloanele nu pot fi comune mai multor aplicații);
- reutilizabilitate (șabloanele sunt revocabile și emise din nou în cazul compromiterii);
- transformarea în sens unic, asemănătoare funcțiilor de *hashing* (șabloanele nu pot fi inversate spre a putea fi obținute datele biometrice reale corespunzătoare utilizatorilor) [64].

Atât sistemul propus, cât și exemplele de direcții de dezvoltare ulterioară dovedesc faptul că biometria bazată pe comportamentul utilizatorilor are un rol important în evoluția sistemelor de securitate ale prezentului și viitorului, menite să protejeze identitatea în era digitală.

REFERINȚE

- [1] V. Bjorn, "Access Control, Logical," in *Encyclopedia of Biometrics, Second Edition*, New York, Springer, 2015, pp. 1-5.
- [2] K. Curran, J. Doherty, A. McCann and G. Turkington, "Good Practice for Strong Passwords," *EDPACS The EDP Audit*, vol. Control, pp. 1-13, 2011.
- [3] N. Clarke, "Intrusive Authentication Approaches," in *Transparent User Authentication: Biometrics, RFID and Behavioural Profiling*, New York, Springer Publishing Company, 2011, pp. 61-110.
- [4] S. Jarecki, H. Krawczyk, M. Shirvanian and N. Saxena, "Two-factor authentication with end-to-end password security," *IACR International Workshop on Public Key Cryptography*, pp. 431-461, 2018.
- [5] R. Grimes, "The many ways to hack 2FA," *Network Security*, no. 9, pp. 8-13, 2019.
- [6] M. L. Brocardo, "PhD Thesis: Continuous Authentication using Stylometry," University of Victoria, 2015.
- [7] A. Alsultan and K. Warwick, "User-Friendly Free-Text Keystroke Dynamics Authentication for Practical Applications," *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4658-4663, 2013.
- [8] S. Karadag, "Undergraduate Thesis Report: Premise of Keystroke Dynamics as a Potential Authentication Method," University of Bedfordshire, 2017.
- [9] M. L. Ali, J. V. Monaco, C. C. Tappert and M. Qiu, "Keystroke biometric systems for user authentication," *Journal of Signal Processing Systems*, vol. 86, no. 2-3, pp. 175-190, 2017.
- [10] „KeyTrac,” [Interactiv]. <https://www.keytrac.net/>. [Accesat 11 Iunie 2020].
- [11] „Keystroke DNA,” [Interactiv]. <https://keystrokedna.com/>. [Accesat 11 Iunie 2020].
- [12] „TypingDNA,” [Interactiv]. <https://www.typingdna.com/>. [Accesat 11 Iunie 2020].
- [13] P. S. Teh, A. Teoh and S. Yue, "A Survey of Keystroke Dynamics Biometrics," *The Scientific World Journal*, vol. 2013, 2013.
- [14] „BehavioSec,” [Interactiv]. <https://www.behaviosec.com/>. [Accesat 11 Iunie 2020].
- [15] P. Bours and S. Mondal, "Continuous authentication with keystroke dynamics," *Norwegian Information Security Laboratory NISlab*, pp. 41-58, 2015.
- [16] H. A. Crawford, "A framework for continuous, transparent authentication on mobile devices," University of Glasgow, 2012.

- [17] Y. Deng and Y. Zhong, "Keystroke dynamics user authentication based on gaussian mixture model and deep belief nets," *ISRN Signal Processing*, 2013.
- [18] S. J. Quraishi and S. Bedi, "Keystroke Dynamics Biometrics, A tool for User Authentication - Review," *International Conference on System Modeling & Advancement in Research Trends (SMART)*, pp. 248-254, 2018.
- [19] S. Tenreiro de Magalhães and H. Santos, "An improved statistical keystroke dynamics algorithm," *China Conference*, pp. 1-5, 2006.
- [20] L. Cheng, F. Liu and D. Yao, "Enterprise data breach: causes, challenges, prevention, and future directions," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 5, p. e1211, 2017.
- [21] T. Dimkov, W. Pieters and P. Hartel, "Laptop theft: a case study on the effectiveness of security mechanisms in open organizations," *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp. 666-668, 2010.
- [22] L. Ponemon, "The cost of a lost laptop," Ponemon Institute, Traverse City, 2009.
- [23] S. C. Yadav and S. K. Singh, *An Introduction to Client/Server Computing*, New Delhi: New Age International, 2009.
- [24] P. Raj, A. Raman and H. Subramanian, *Architectural Patterns: Uncover essential patterns in the most indispensable realm of enterprise architecture*, Birmingham: Packt Publishing Ltd., 2017.
- [25] C. Chio and D. Freeman, *Machine Learning and Security: Protecting Systems with Data and Algorithms*, Sebastopol: O'Reilly Media, 2018.
- [26] L. P. Coelho and W. Richert, *Building Machine Learning Systems with Python*, Birmingham: Packt Publishing Ltd., 2015.
- [27] „PyCharm Features,” [Interactiv]. <https://www.jetbrains.com/pycharm/features/>. [Accesat 30 Aprilie 2020].
- [28] „Qt for Python Documentation,” [Interactiv]. <https://doc.qt.io/qtforpython/>. [Accesat 30 Aprilie 2020].
- [29] „PyQt5 Reference Guide,” [Interactiv]. <https://www.riverbankcomputing.com/static/Docs/PyQt5/>. [Accesat 30 Aprilie 2020].
- [30] N. Sherriff, *Learn Qt 5: Build modern, responsive cross-platform desktop applications with Qt, C++, and QML*, Birmingham: Packt Publishing Ltd, 2018.
- [31] „MySQL Documentation,” [Interactiv]. <https://dev.mysql.com/doc/>. [Accesat 30 Aprilie 2020].

- [32] R. J. Dyer, Learning MySQL and MariaDB: heading in the right direction with MySQL and MariaDB, Sebastopol: O'Reilly Media, Inc., 2015.
- [33] „Python 3.8.3 Documentation - socket,” [Interactiv]. <https://docs.python.org/3/library/socket.html>. [Accesat 30 Aprilie 2020].
- [34] „Python 3.8.3 Documentation - threading,” [Interactiv]. <https://docs.python.org/3/library/threading.html>. [Accesat 30 Aprilie 2020].
- [35] „NumPy v1.18 Manual,” [Interactiv]. <https://numpy.org/doc/>. [Accesat 30 Aprilie 2020].
- [36] „Scikit-learn User Guide,” [Interactiv]. https://scikit-learn.org/stable/user_guide.html. [Accesat 30 Aprilie 2020].
- [37] „Python 3.8.3 Documentation - ssl,” [Interactiv]. <https://docs.python.org/3/library/ssl.html>. [Accesat 30 Aprilie 2020].
- [38] „OpenSSL Documentation,” [Interactiv]. <https://www.openssl.org/docs/>. [Accesat 30 Aprilie 2020].
- [39] „Modern password hashing for your software and your servers - bcrypt,” [Interactiv]. <https://pypi.org/project/bcrypt/>. [Accesat 30 Aprilie 2020].
- [40] „Python 3.8.3 Documentation - hashlib,” [Interactiv]. <https://docs.python.org/3/library/hashlib.html>. [Accesat 30 Aprilie 2020].
- [41] „Diagrams.net,” [Interactiv]. <https://www.diagrams.net/>. [Accesat 30 Aprilie 2020].
- [42] „Microsoft Planner - A simple, visual way to organize teamwork,” [Interactiv]. <https://tasks.office.com/>. [Accesat 30 Aprilie 2020].
- [43] R. Giot, M. El-Abed and C. Rosenberger, "Keystroke dynamics authentication," *Biometrics*, vol. 1, pp. 157-182, 2011.
- [44] F. Monroe and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Generation computer systems*, vol. 16, no. 4, pp. 351-359, 2000.
- [45] X. Wan, "Influence of feature scaling on convergence of gradient iterative algorithm," *Journal of Physics: Conference Series*, vol. 1213, no. 3, 2019.
- [46] A. Géron, Hands-On Machine Learning with Scikit-Learn: Concepts, Tools, and Techniques to Build Intelligent Systems, Sebastopol: O'Reilly Media, 2017.
- [47] S. Raschka, „About Feature Scaling and Normalization - and the effect of standardization for machine learning algorithms,” 11 Iulie 2014. [Interactiv]. https://sebastianraschka.com/Articles/2014_about_feature_scaling.html. [Accesat 12 Iunie 2020].
- [48] C. Albon, Machine Learning with Python Cookbook, Sebastopol: O'Reilly Media, 2018.

- [49] V. Dhakal, A. Feit, P. O. Kristensson and A. Oulasvirta, "Observations on Typing from 136 Million Keystrokes," *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1-12, 2018.
- [50] V. Dhakal, A. Feit, P. O. Kristensson și A. Oulasvirta, „The 136M Keystrokes Dataset,” 2018. [Interactiv]. <https://userinterfaces.aalto.fi/136Mkeystrokes/>. [Accesat 30 Aprilie 2020].
- [51] X. Zeng and T. R. Martinez, "Distribution-balanced stratified cross-validation for accuracy estimation," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 12, no. 1, pp. 1-12, 2000.
- [52] T.-T. Wong and P.-Y. Yeh, "Reliable Accuracy Estimates from k-fold Cross Validation," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, 2019.
- [53] S. O. Al-Mamory and Z. M. Ali, *Using DBSCAN Clustering Algorithm in Detecting DDoS Attack*, Babylon/ Kufa, 2019.
- [54] J. Samuelsson, *Anomaly Detection in Console Logs*, Uppsala: Uppsala University, 2016.
- [55] B. Hoyle, M. M. Rau, K. Paech, C. Bonnett, S. Seitz and J. Weller, "Anomaly detection for machine learning redshifts applied to SDSS galaxies," *Monthly Notices of the Royal Astronomical Society*, vol. 452, no. 4, pp. 4183-4194, 2015.
- [56] S. S. Khan and M. G. Madden, "A survey of recent trends in one class classification," *Irish conference on artificial intelligence and cognitive science*, pp. 188-197, 2009.
- [57] A. D. Shieh and D. F. Kamm, "Ensembles of One Class Support Vector," *MCS: International Workshop on Multiple Classifier Systems*, pp. 181-190, 2009.
- [58] A. Bánhalmi, R. Busa-Fekete and B. Kégl, "A one-class classification approach for protein sequences and structures," *International Symposium on Bioinformatics Research and Applications*, pp. 310-322, 2009.
- [59] M. M. Breunig, H.-P. Kriegel, R. T. Ng and J. Sander, "LOF: identifying density-based local outliers," *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93-104, 2000.
- [60] N. Paulauskas and A. F. Bagdonas, "Local outlier factor use for the network flow anomaly detection," *Security and Communication Networks*, vol. 8, no. 18, pp. 4203-4212, 2015.
- [61] R. Giot, B. Dorizzi and C. Rosenberger, "Analysis of template update strategies for keystroke dynamics," *2011 IEEE Workshop on Computational Intelligence in Biometrics and Identity Management (CIBIM)*, pp. 21-28, 2011.
- [62] M. El-Abed, R. Giot, B. Hemery, C. Rosenberger and J.-J. Schwartzmann, "Towards the Security Evaluation of Biometric Authentication Systems," *International Conference on Security Science and Technology (ICSST)*, pp. 167-173, 2011.

- [63] M. I. Mihailescu, "PhD Thesis: About Security of Biometrics Authentication Protocols," University of Bucharest, Faculty of Mathematics and Computer Science, Bucharest, 2015.
- [64] A. Dwivedi, S. Kumar and M. Singh, "Cancellable Biometrics for Security and Privacy Enforcement on Semantic Web," *International Journal of Computer Applications*, vol. 21, no. 8, pp. 1-8, 2011.
- [65] O. A. Coica, „Continuous Authentication System - Source Code,” 1 Iulie 2020. [Interactiv]. <https://github.com/OanaAlexandraC/Continuous-Authentication-System>. [Accesat 1 Iulie 2020].

Anexa 1 – Secvență de cod pentru colectarea datelor brute

```
from pynput import keyboard
import csv
import time
import sys, os

class KeyLogger:
    def __init__(self):
        self.path = "logged_keystrokes.csv"
        self.start_time = time.time()
        with open(self.path, 'w', newline='') as file:
            self.keystrokes_file = csv.writer(file)
        with keyboard.Listener(
            on_press=self.pressed_key,
            on_release=self.released_key) as listener:
            listener.join()

    def pressed_key(self, key):
        with open(self.path, 'a', newline='') as file:
            self.keystrokes_file = csv.writer(file)
            try:
                self.keystrokes_file.writerow(["pressed", str(key), time.time() - self.start_time])
            except UnicodeEncodeError:
                pass

    def released_key(self, key):
        with open(self.path, 'a', newline='') as file:
            self.keystrokes_file = csv.writer(file)
            try:
                self.keystrokes_file.writerow(["released", str(key), time.time() - self.start_time])
            except UnicodeEncodeError:
                pass

    with open(self.path) as f:
        number_of_lines = sum(1 for line in f)
        if number_of_lines > 100:
            return False

# key_logger = KeyLogger()
```

Anexa 2 – Secvență de cod pentru extragerea caracteristicilor (amprentelor)

```
import csv
import sys
import os
from operator import itemgetter

def find_key_category(key):
    category = {
        "Key.esc": "l", "Key.delete": "r",
        "'1'": "l", "'2'": "l", "'3'": "l", "'4'": "l", "'5'": "l",
        "'6'": "r", "'7'": "r", "'8'": "r", "'9'": "r", "'0'": "r", "'-':": "r", "'='": "r",
        "'q'": "l", "'w'": "l", "'e'": "l", "'r'": "l", "'t'": "l",
        "'y'": "r", "'u'": "r", "'i'": "r", "'o'": "r", "'p'": "r", "'['": "r", "']'": "r", "'\\'": "r",
        "'a'": "l", "'s'": "l", "'d'": "l", "'f'": "r", "'g'": "l",
        "'h'": "r", "'j'": "r", "'k'": "r", "'l'": "r", "';'": "r", "':'": "r", "Key.enter": "r",
        "'z'": "l", "'x'": "l", "'c'": "l", "'v'": "l", "'b'": "l",
        "'n'": "r", "'m'": "r", "','": "r", "'.'": "r", "'/'": "r"
    }
    if key in category:
        return category.get(key)
    else:
        if key.lower() in category:
            return category.get(key.lower())
        else:
            return key

class DataExtractor:
    def __init__(self, path):
        # hold time
        self.hold_left = 0.0
        self.hold_right = 0.0
        self.hold_space = 0.0

        # latency
        self.pp_left_left = 0.0
        self.pp_left_right = 0.0
        self.pp_right_left = 0.0
        self.pp_right_right = 0.0

        self.rr_left_left = 0.0
        self.rr_left_right = 0.0
        self.rr_right_left = 0.0
        self.rr_right_right = 0.0

        self.pr_left_left = 0.0
        self.pr_left_right = 0.0
        self.pr_right_left = 0.0
        self.pr_right_right = 0.0

        self.path = path
        with open(self.path, newline='') as file:
            self.data = list(csv.reader(file))
        self.average_hold_time = []

    def delete_duplicates(self):
        i = 0
        while i < len(self.data) - 2:
            if self.data[i][0] == self.data[i + 1][0] and self.data[i][1] == self.data[i + 1][1]:
                del self.data[i + 1]
            else:
                i = i + 1

    def format_data(self):
        # self.data.pop()
        formatted_data = []
        for log in self.data:
            if (not log[1] == "Key.shift") & (not log[1] == "Key.shift_r") & (not log[1] == "Key.shift_l") & (not log[1] == "Key.caps_lock"):
                log[2] = float(log[2])
                formatted_data.append(log)
        self.data = formatted_data
        self.data = sorted(self.data, key=itemgetter(2))
        self.delete_duplicates()
        for log in self.data:
            log[2] = str(log[2])
```

```

def determine_average_hold_time(self):
    hold_time = []
    for i in range(0, len(self.data)):
        if (self.data[i][0]) == "pressed":
            for j in range(i + 1, len(self.data)):
                if (self.data[j][0]) == "released":
                    if self.data[i][1] == self.data[j][1]: # found a pair
                        pair = [self.data[i][1], float(self.data[j][2]) - float(self.data[i][2])]
                        hold_time.append(pair)
                        break
    for row in hold_time:
        row[0] = find_key_category(row[0])
    # print(hold_time)

    total = 0.0
    number = 0
    for row in hold_time:
        if row[0] == 'l':
            total += float(row[1])
            number += 1
    if number != 0:
        self.hold_left = total / number
    # print(self.hold_left)
    # [similar pentru self.hold_right, self.hold_space]

def determine_latency(self):
    pressed_key_timestamp = []
    for pressed in self.data:
        if pressed[0] == "pressed":
            # introducem valori de tipul (categoria tastei, timestamp)
            pair = [find_key_category(pressed[1]), pressed[2]]
            if not pair[0] == "Key.space":
                pressed_key_timestamp.append(pair)
    # print(pressed_key_timestamp)

    total_pp_left_left = 0.0
    number_pp_left_left = 0
    # [similar pentru total_pp_left_right, number_pp_left_right, total_pp_right_left,
    number_pp_right_left, total_pp_right_right, number_pp_right_right]
    for i in range(0, len(pressed_key_timestamp) - 1):
        key1 = pressed_key_timestamp[i][0]
        time1 = float(pressed_key_timestamp[i][1])
        key2 = pressed_key_timestamp[i + 1][0]
        time2 = float(pressed_key_timestamp[i + 1][1])
        if time2 - time1 < 1.5:
            if (key1 == "l") & (key2 == "l"):
                total_pp_left_left = total_pp_left_left + time2 - time1
                number_pp_left_left += 1
            elif (key1 == "l") & (key2 == "r"):
                total_pp_left_right = total_pp_left_right + time2 - time1
                number_pp_left_right += 1
            elif (key1 == "r") & (key2 == "l"):
                total_pp_right_left = total_pp_right_left + time2 - time1
                number_pp_right_left += 1
            elif (key1 == "r") & (key2 == "r"):
                total_pp_right_right = total_pp_right_right + time2 - time1
                number_pp_right_right += 1

    if number_pp_left_left != 0:
        self.pp_left_left = total_pp_left_left / number_pp_left_left
    # print(self.pp_left_left)
    # [similar pentru self.pp_left_right, self.pp_right_left, self.pp_right_right]
    #####

    released_key_timestamp = []
    for released in self.data:
        if released[0] == "released":
            # introducem valori de tipul (categoria tastei, timestamp)
            pair = [find_key_category(released[1]), released[2]]
            if not pair[0] == "Key.space":
                released_key_timestamp.append(pair)
    # print(released_key_timestamp)

    total_rr_left_left = 0.0
    number_rr_left_left = 0
    # [similar pentru total_rr_left_right, number_rr_left_right, total_rr_right_left,
    number_rr_right_left, total_rr_right_right, number_rr_right_right]

    for i in range(0, len(released_key_timestamp) - 1):
        key1 = released_key_timestamp[i][0]
        time1 = float(released_key_timestamp[i][1])
        key2 = released_key_timestamp[i + 1][0]
        time2 = float(released_key_timestamp[i + 1][1])
        if time2 - time1 < 1.5:
            if (key1 == "l") & (key2 == "l"):
                total_rr_left_left = total_rr_left_left + time2 - time1
                number_rr_left_left += 1

```

```

        elif (key1 == "l") & (key2 == "r"):
            total_rr_left_right = total_rr_left_right + time2 - time1
            number_rr_left_right += 1
        elif (key1 == "r") & (key2 == "l"):
            total_rr_right_left = total_rr_right_left + time2 - time1
            number_rr_right_left += 1
        elif (key1 == "r") & (key2 == "r"):
            total_rr_right_right = total_rr_right_right + time2 - time1
            number_rr_right_right += 1

    if number_rr_left_left != 0:
        self.rr_left_left = total_rr_left_left / number_rr_left_left
    # print(self.rr_left_left)
    # [similar pentru number_rr_left_right, number_rr_right_left, number_rr_right_right]

    #####

    total_pr_left_left = 0.0
    number_pr_left_left = 0
    # [similar pentru total_pr_left_right, number_pr_left_right, total_pr_right_left,
    number_pr_right_left, total_pr_right_right, number_pr_right_right]

    pressed_released_key_timestamps = []
    for i in range(0, len(self.data) - 1):
        if self.data[i][0] == "pressed":
            # introducem valori de tipul (categorie tasta apasata, timestamp apasare, timestamp
            # eliberare)
            triple = [find_key_category(self.data[i][1]), float(self.data[i][2])]
            for j in range(i + 1, len(self.data)):
                if (self.data[j][0] == "released") & (self.data[j][1] == self.data[i][1]):
                    triple.append(float(self.data[j][2]))
                    break
            if not triple[0] == "Key.space":
                if len(triple) == 3:
                    pressed_released_key_timestamps.append(triple)
    # print(pressed_released_key_timestamps)

    for i in range(1, len(pressed_released_key_timestamps)):
        # pressed_released_key_timestamps[i] - pressed
        # pressed_released_key_timestamps[i-1] - released
        time1 = pressed_released_key_timestamps[i - 1][2]
        time2 = pressed_released_key_timestamps[i][1]
        if time2 - time1 < 1.5:
            if (pressed_released_key_timestamps[i][0] == 'l') & (pressed_released_key_timestamps[i -
1][0] == 'l'):
                total_pr_left_left += time2 - time1
                number_pr_left_left += 1
            elif (pressed_released_key_timestamps[i][0] == 'l') & (pressed_released_key_timestamps[i -
1][0] == 'r'):
                total_pr_left_right += time2 - time1
                number_pr_left_right += 1
            elif (pressed_released_key_timestamps[i][0] == 'r') & (pressed_released_key_timestamps[i -
1][0] == 'l'):
                total_pr_right_left += time2 - time1
                number_pr_right_left += 1
            elif (pressed_released_key_timestamps[i][0] == 'r') & (pressed_released_key_timestamps[i -
1][0] == 'r'):
                total_pr_right_right += time2 - time1
                number_pr_right_right += 1

    if number_pr_left_left != 0:
        self.pr_left_left = total_pr_left_left / number_pr_left_left
    # print(self.pr_left_left)
    # [similar pentru self.pr_left_right, self.pr_right_left, self.pr_right_right]

    def get_keystroke_dynamic_information(self):
        return [self.hold_left, self.hold_right, self.hold_space,
                self.pp_left_left, self.pp_left_right, self.pp_right_left, self.pp_right_right,
                self.rr_left_left, self.rr_left_right, self.rr_right_left, self.rr_right_right,
                self.pr_left_left, self.pr_left_right, self.pr_right_left, self.pr_right_right]

    def run(self):
        self.format_data()
        # print(self.data)
        self.determine_average_hold_time()
        self.determine_latency()

# data_extractor = DataExtractor("E:/Program Files/AuthenticationSystem/test.csv")
# data_extractor.run()
# print(data_extractor.get_keystroke_dynamic_information())

```

Anexa 3 – Glosar

2FA	Two-Factor Authentication (autentificare în doi factori)
AES	Advanced Encryption Standard
DBSCAN	Density-Based Spatial Clustering of Applications with Noise (algoritmul de clusterizare spațială bazată pe densitate pentru aplicațiile cu variații incontrolabile)
DCAP	Data-Centric Audit and Protection (auditare și protecție centrate pe date) Data Loss Prevention (prevenție a pierderilor de date) / Data Leak
DLP / DLPD	Prevention and Detection (dectecție și prevenție a scurgerilor de informații)
FAR	False Acceptance Rate (rata de acceptare falsă)
FN	False Negative (numărul de negative false)
FP	False Positive (numărul de pozitive false)
FRR	False Rejection Rate (rata de respingere falsă)
IP	Internet Protocol
OC-SVM	One-Class Support Vector Machines (mașini cu suport vectorial pentru probleme cu o clasă)
SHA	Secure Hash Algorithm
SMS	Short Message Service (serviciul pentru mesaje scurte)
SSL	Secure Sockets Layer
TA	Training Accuracy (acuratețea la antrenare)
TLS	Transport Layer Security
TN	True Negative (numărul de negative adevărate)
TP	True Positive (numărul de pozitive adevărate)

Anexa 4 – Lista figurilor și tabelelor

Figura 3.1: Topologie de tip client-server.....	15
Figura 3.2: Captură task-uri din Microsoft Planner asociate aplicației	18
Figura 3.3: Arhitectura conceptuală a sistemului pentru înrolarea unui utilizator	18
Figura 3.4: Arhitectura conceptuală a sistemului pentru autentificarea unui utilizator	19
Figura 4.1: Componentele unui sistem de autentificare continuă bazat pe caracteristici comportamentale	21
Figura 4.2: Împărțirea logică a tastaturii pentru extragerea caracteristicilor	23
Figura 4.3: Schema bazei de date	31
Figura 5.1: Puncte vulnerabile într-un sistem de autentificare continuă de tip biometric (figură adaptată și extinsă după [62])	33
Tabelul 5.1: Rezultatele algoritmului Elliptic Envelope pentru cei șase voluntari.....	36
Tabelul 5.2: Rezultatele algoritmului One-Class Support Vector Machine pentru cei șase voluntari	37
Tabelul 5.3: Rezultatele algoritmului Local Outlier Factor pentru cei șase voluntari.....	37
Figura 6.1: Pagina de conectare a utilizatorului.....	39
Figura 6.2: Butonul de creare a unui cont nou	40
Figura 6.3: Formularul de înrolare.....	40
Figura 6.4: Momentul monitorizării inițiale	41
Figura 6.5: Faza I de autentificare	42
Figura 6.6: Faza a II-a de autentificare	43
Figura 6.7: Pagina ce apare în situația autentificării complete reușite	43
Figura 6.8: Formularul corespunzător schimbării parolei.....	44
Figura 6.9: Erori și atenționări posibile	45
Figura 6.10: Fila implicită a aplicației componentei server, corespunzătoare înregistrărilor evenimentelor.....	47
Figura 6.11: Cea de-a doua filă a aplicației componentei server, corespunzătoare listei de utilizatori	48