Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувачка кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«14» червня 2022р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи
***бакалавра*.**
(назва освітнього ступеня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Програмний модуль багатофакторної автентифікації та авторизації»

Виконавець: студент IV курсу, групи КБ-42

_____ **Вадим ШРАМЕНКО**
(підпис) (прізвище ім'я)

	Прізвище, ініціали	**Підпис**
Керівник	Лариса МИРУТЕНКО	

Нормоконтроль	Олександр ТОРОШАНКО	

Київ 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувачка кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«01» листопада 2021 р

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентові **КБ-42** **Вадиму Олександровичу Шраменко**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Програмний модуль багатофакторної автентифікації та авторизації

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №5 від 29.10.2021 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Технології автентифікації, авторизації та адміністрування дій
Методи автентифікації та авторизації, що використовують паролі та
PIN-коди та методи строгої автентифікації, їх вразливості та методи захисту.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Аналіз сучасних методів і засобів автентифікації та авторизації.
Побудова раціональної системи автентифікації та авторизації в
інформаційно-комунікаційних системах.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розробка програмного модуля багатофакторної

автентифікації та авторизації для використання в застосунках або веб-сервісах.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29.10.2021 року

Завдання видав _____ (підпис) Лариса МИРУТЕНКО. (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Вадим ШРАМЕНКО. (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.10.2021 – 27.01.2022	*виконано*
2	Аналіз літератури	28.01.2022 – 11.02.2022	*виконано*
3	Аналіз методів автентифікації	12.02.2022 – 24.03.2022	*виконано*
4	Дослідження основних вразливостей	25.02.2022 – 24.03.2022	*виконано*
5	Дослідження методів захисту	25.03.2022 – 07.04.2022	*виконано*
6	Визначення методу для реалізації	08.04.2022 – 20.04.2022	*виконано*
7	Вибір необхідних технологій	21.04.2022 – 05.05.2022	*виконано*
8	Реалізація програмного модуля	06.05.2022 – 20.05.2022	*виконано*
9	Реалізація методів захисту інформації	21.05.2022 – 04.06.2022	*виконано*
10	Оформлення пояснювальної записки	05.06.2022 – 08.06.2022	*виконано*
11	Підготовка до захисту	09.06.2022 – 13.06.2022	*виконано*

Завдання видав _____ (підпис) Лариса МИРУТЕНКО (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Вадим ШРАМЕНКО (ім'я, прізвище)

Термін подання дипломної роботи до ЕК 08 червня 2022 року

РЕФЕРАТ

Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел та додатків. Основний текст займає 68 сторінки, включає в себе зміст, вступ, три розділи дипломної роботи, 13 рисунків, 1 таблицю, висновки та список джерел. Крім того, робота містить 2 додатки із загальною кількістю сторінок 9.

Метою роботи є розробка програмного модуля багатофакторної автентифікації та авторизації.

Для досягнення зазначеної мети поставлено наступні завдання:

- Проаналізувати основні методи автентифікації та авторизації.
- Проаналізувати вразливості систем автентифікації та авторизації.
- Визначити оптимальний метод безпечної багатофакторної автентифікації та авторизації.
- Розробити програмний модуль автентифікації та авторизації з використанням засобів та механізмів захисту.

Об'єктом дослідження є процес розробки програмного модуля автентифікації та авторизації з використанням засобів та механізмів захисту.

Предметом дослідження є набір механізмів, що реалізують захист автентифікації та авторизації.

Практичною цінністю отриманих результатів є розробка програмного модуля багатофакторної автентифікації та авторизації, який може використовуватись для підвищення безпеки облікових даних в застосунках або веб-сервісах.

Ключові слова: автентифікація, авторизація, база даних, вразливості, захист персональних даних, облікові дані, python.

ЗМІСТ

ВСТУП

Актуальність даної роботи визначається тим, що на даний момент більшість інформаційних систем потребують підтвердження особистості і зазвичай користувачі використовують недосконалі паролі. Тому, веб-сайти, які мають форми реєстрації, мають повідомляти наскільки захищений введений пароль. Небезпечні паролі є одними з найбільших вразливостей, які існують в організації.

Щоб користувач отримав доступ до захищеного ресурсу в Інтернеті, він повинен пройти автентифікацію. Існують різні форми автентифікації. Найпоширенішою є схема звичайного імені користувача та пароля. Ця схема дуже проста у реалізації, але вона страждає від вразливостей безпеки і вимагає від користувача запам’ятовувати паролі до всіх облікових записів. Двофакторна автентифікація може бути однією з рішень для підвищення безпеки там, де однофакторна автентифікація відсутня. Однак, залежно від реалізації, двофакторна автентифікація все ще може бути небезпечною і навіть більш незручною для користувачів.

Останніми днями шахрайство в Інтернеті є одним з основних недоліків для широкого використання комерційних додатків. Тому існують важливі аспекти безпеки, що впливають на конфіденційність та доступність, а саме: ідентифікація, автентифікація та авторизація. На сьогоднішній день існує багато рішень для багатофакторної автентифікації, від простого пароля до останніх технологій, заснованих на RFID (радіочастотна ідентифікація) або біометричних даних.

У цій роботі проведено огляд існуючих методів автентифікації, їх плюси та мінуси, а також запропоновано програмну реалізацію багатофакторної автентифікації.

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ ПРО АВТЕНТИФІКАЦІЮ ТА АВТОРИЗАЦІЮ

1.1 Поняття та процес автентифікації

Автентифікація відноситься до процесу ідентифікації особи, зазвичай на основі імені користувача, пароля та певного типу додаткової перевірки. Автентифікація підтверджує, що особа є тим, ким вона себе видає, що запобігає несанкціонованому доступу до програми, системи, мережі чи пристрою, але не впливає на права доступу особи. У системах безпеки автентифікація є окремою формою авторизації, процесом допуску осіб до об’єктів системи на основі їхньої особистості. Користувачі, зазвичай, ідентифікуються з ідентифікатором користувача, а автентифікація виконується, коли користувач надає облікові дані, наприклад пароль, який відповідає цьому ідентифікатору користувача. Цей пароль повинен бути відомий тільки користувачеві.

Після автентифікації користувач або процес піддаються процесу авторизації, щоб визначити, чи має об'єкт доступ до захищеного ресурсу або системи. Тобто, користувач може бути автентифікований, але не може отримати доступ до ресурсу.

Автентифікація та авторизація дуже часто взаємозаміняються. Автентифікація це процес перевірки ідентичності зареєстрованого користувача перед наданням доступу до захищеного ресурсу, а авторизація є процесом перевірки того, що автентифікований користувач має права на доступ до ресурсів.

Автентифікація використовується в усіх компаніях та сервісах. Організації використовують автентифікацію, щоб контролювати, які користувачі мають доступ до корпоративних мереж і ресурсів, а також визначати та контролювати, які процеси та сервери мають до них доступ. Компанії також використовують автентифікацію, щоб дозволити віддаленим працівникам безпечно отримувати доступ до своїх програм і мереж.

Для підприємств та інших великих організацій автентифікацію можна здійснити за допомогою системи єдиного входу (SSO), що реалізує доступ до всіх підсистем без повторного введення логіну/паролю. Для цього користувачеві достатньо ввести логін та пароль лише раз для однієї з підсистем і він матиме доступ до всіх інших.

Традиційно автентифікація здійснюється за допомогою систем або ресурсів, до яких здійснюється доступ, наприклад, ІС автентифікує користувачів, використовуючи власну систему паролів, реалізовану локально, використовуючи ідентифікатори(логіни) для входу і паролі.

Кожен користувач при реєстрації(самостійно або адміністратором) в системі одержує набір персональних реквізитів (зазвичай використовують пару логін-пароль). При кожній наступній спробі входу він має використати вище зазначену інформацію(вона має бути унікальна для кожного користувача). На підставі вказаної інформації система робить висновок про особистість та надає відповідні права.

Загальні етапи для автентифікації:

1. Початковий етап: користувач не автентифікований.

2. Етап підключення: користувач вимагає від інформаційної системи (ІС) використання функції, що вимагає автентифікація. ІС перевіряє справжність користувача.

3. Етап автентифікації: користувач автентифікований і відкривається сеанс. ІС надає користувачеві необхідні функції.

4. Етап відключення: користувач відключається і стан повертається до початкового кроку. Цей етап може бути ініційований після певного періоду бездіяльності, або дією користувача.

ІС може вимагати різних рівнів автентифікації, наприклад, рівень для адміністраторів і рівень для користувачів. У такій системі рівень автентифікації поділено за шкалою: рівень 0 для неавтентифікованих користувачів з найнижчими правами в системі, рівень N для адміністратора з повними правами, і один або кілька рівнів від 0 до N. Ця схема полягає в тому, що для переході на більш високий рівень довіри з боку ІС може знадобитися автентифікація [1].

Для автентифікації застосовуються різні протоколи. Протокол автентифікації - це тип протоколу комп'ютерного зв'язку або криптографічного протоколу, спеціально розроблений для передачі даних автентифікації між двома об'єктами [2]. Це найважливіший рівень захисту, необхідний для безпечного зв'язку в комп'ютерних мережах. Найчастіше використовують наступні протоколи:

1. Протокол автентифікації пароля(анлг. PAP - Password Authentication Protocol) - є одним із найстаріших протоколів автентифікації. Автентифікація ініціалізується тим, що клієнт надсилає пакет з обліковими даними (ім’я користувача та пароль) на початку з’єднання, при цьому клієнт повторює запит на автентифікацію, поки не буде отримано підтвердження [3]. Це дуже небезпечно, тому що облікові дані надсилаються відкрито і багаторазово, що робить його вразливим навіть до найпростіших атак, таких як підслуховування та атаки «людина посередині». Незважаючи на широку підтримку, зазначено, що якщо реалізація пропонує більш надійний метод автентифікації, цей метод має бути запропонований перед PAP. Змішана автентифікація (наприклад, один і той же клієнт по черзі використовує як PAP, так і CHAP) також не очікується, оскільки автентифікація CHAP буде скомпрометована, якщо PAP надсилає пароль у вигляді простого тексту.

2. Протокол автентифікації виклику рукостискання(англ. CHAP - Challenge-handshake authentication protocol) Процес автентифікації в цьому протоколі завжди ініціалізується сервером / хостом і може виконуватися в будь-який час протягом сеансу, навіть повторно. Сервер відправляє випадковий рядок (зазвичай довжиною 128 Б). Клієнт використовує пароль і рядок, отримані в якості параметрів для геш-функції MD5, а потім відправляє результат разом з ім'ям користувача у вигляді звичайного тексту. Сервер використовує ім'я користувача для застосування тієї ж функції і порівнює обчислений і отриманий геш.

3. Розширений протокол автентифікації (англ. EAP - Extensible Authentication Protocol) - це структура автентифікації, а не конкретний механізм автентифікації [4]. Він забезпечує деякі загальні функції та узгодження методів автентифікації, які називаються методами EAP. На даний момент визначено близько 40 різних методів. Методи, визначені в RFC IETF, включають EAP-MD5, EAP-POTP, EAP-GTC, EAP-

TLS, EAP-IKEv2, EAP-SIM, EAP-AKA і EAP-AKA'. Загальний алгоритм: сервер відправляє запит на автентифікацію однорангового вузла; вузол відправляє пакет відповіді у відповідь на дійсний запит; сервер відправляє додатковий пакет запиту, і вузол відповідає; діалог триває до тих пір, поки сервер не зможе автентифікувати однорангового вузла (неприйнятні відповіді на один або кілька запитів), і в цьому випадку реалізація сервера повинна передати повідомлення про помилку EAP; в якості альтернативи, діалог автентифікації може тривати до тих пір, поки сервер не з'ясує, що автентифікація пройшла успішно, і в цьому випадку автентифікатор повинен передати успіх EAP [5].

4. TACACS+(Terminal Access Controller Access-Control System). Протокол, розроблений Cisco і випущений як відкритий стандарт, починаючи з 1993 року. Хоча TACACS+ походить від TACACS, він є окремим протоколом, який обробляє послуги автентифікації, авторизації та обліку (AAA). TACACS+ значною мірою замінив своїх попередників. TACACS + розділяє компоненти, тому вони можуть бути відокремлені і оброблятися на окремих серверах (він навіть може використовувати інший протокол, наприклад, для авторизації). Він використовує TCP (протокол управління передачею) для транспортування і шифрує весь пакет.

5. Служба віддаленої автентифікації користувачів (англ. RADIUS - Remote Authentication Dial-In User Service). Це мережевий протокол, який забезпечує централізоване керування автентифікацією, авторизацією та обліком (AAA) для користувачів, які підключаються та використовують мережеву службу.

6. DIAMETR. Походить від RADIUS і включає в себе безліч поліпшень, таких як використання більш надійного транспортного протоколу TCP або SCTP і більш високий рівень безпеки завдяки TLS.

7. OpenID це відкритий стандарт і децентралізований протокол автентифікації, який просувається некомерційним фондом OpenID. Він дозволяє користувачам проходити автентифікацію на сайтах, які співпрацюють (відомі як довіряючі сторони, або RP) за допомогою послуги стороннього постачальника ідентифікаційної інформації (IDP), усуваючи потребу веб-майстрам надавати власні спеціальні системи входу та дозволяючи користувачам входити в систему на кілька

непов’язаних веб-сайтів без необхідності мати окрему особу та пароль для кожного [6].

8. Universal 2nd Factor (U2F) є відкритим стандартом, який посилює та спрощує двофакторну автентифікацію за допомогою спеціалізованих пристроїв універсальної послідовної шини (USB) або пристроїв ближнього зв’язку (NFC) на основі подібних технологій безпеки, які є в смарт-картках. Його змінює проект FIDO2, який включає стандарт веб-автентифікації W3C (WebAuthn) і протокол Alliance's Client to Authenticator Protocol (CTAP2). Алгоритм взаємодії з U2F на стороні браузера такий:

a. Користувач проходить верифікацію логіну та паролю;

b. Залежна сторона, наприклад Google, через U2F JS API запитує підпис виклику(challenge);

c. Браузер надсилає запит на пристрій. Якщо користувач підтвердив, наприклад, за допомогою натискання кнопки або іншим чином, своє бажання зробити двофакторну автентифікацію, то пристрій повертає підпис виклику

d. Браузер передає підпис залежній стороні;

e. Залежна сторона верифікує підпис.

Таким чином, користувач повинен буде торкнутися кнопки, щоб зареєструватися, а також може бути попереджений браузером. Сторона, що довіряє, може розмістити екрани, які проведуть користувача через ці кроки. Реєстрація — це дуже цінна операція — вона дає джерелу можливість дуже ретельно перевіряти користувача, і до неї потрібно ставитися дуже серйозно. Під час автентифікації (або в загальному випадку, коли онлайн-сервіс або веб-сайт потребує ретельної перевірки користувача, запитуючи підпис), користувач повинен активувати пристрій, щоб продемонструвати присутність користувача, перш ніж підпис може статися.

Існують також інші протоколи автентифікації, а саме: Kerberos, AKA, Basic access authentication, CAVE-based authentication, CRAM-MD5, Digest, Host Identity Protocol (HIP), LAN Manager, NTLM.

1.2 Стандарти для регулювання автентифікації

Стандарти ідентифікації — це відкриті специфікації та протоколи, що надають чіткі вказівки щодо того, як розробити системи автентифікації та авторизації для керування ідентифікацією, безпечного переміщення особистих даних і вирішувати, хто може отримати доступ до програм і даних, щоб декілька сторін могли легко досягти сумісності.

Використання стандартів автентифікації підвищує безпеку та знижує ризик, оскільки кінцевий користувач повинен бути ідентифікований та автентифікований лише один раз постачальником ідентифікаційних даних, а потім ця ідентифікаційна інформація може використовуватися в кількох системах.

Національний стандарт ДСТУ ISO/IEC 9798-1:2015 (ISO/IEC 9798-1:2010, IDT) «Інформаційні технології. Методи захисту. Автентифікація об’єктів. Частина 1. Загальні положення». Цей стандарт установлює модель автентифікації і загальні вимоги та обмеження для механізмів автентифікації об’єктів, які використовують методи захисту. Ці механізми використовують, щоб підтвердити, що об’єкт той, за кого себе видає. Об’єкт, який треба автентифікувати, доводить свою ідентичність, показуючи свою обізнаність з таємною інформацією. Механізми визначено як обміни інформацією між об’єктами і, за потреби, обмінами з довіреною третьою стороною [7]. У цьому національному стандарті зазначено вимоги, які відповідають законодавству України.

NIST Special Publication 800-63B Digital Identity Guidelines. Ці рекомендації містять технічні вимоги до федеральних агентств, які впроваджують послуги цифрової ідентифікації, і не мають на меті обмежувати розробку або використання стандартів за межами цієї мети. Ці рекомендації зосереджені на автентифікації суб’єктів, які взаємодіють з державними системами через відкриті мережі, встановлюючи, що даний користувач є абонентом, який був попередньо автентифікований. Результат процесу автентифікації може використовуватися локально системою, яка виконує автентифікацію, або може бути підтверджений деінде в об’єднаній системі ідентифікації. Цей документ визначає технічні вимоги

до кожного з трьох рівнів гарантії автентифікації. Ця публікація замінює відповідні розділи спеціальної публікації NIST (SP) 800-63-2 [8].

1.3 Типи та методи автентифікації

Основним типом автентифікації користувача вважається використання ідентифікатора та пароля, і залежить від того, чи користувач знає обидві частини інформації: логін (ідентифікатор користувача або ім'я користувача) і пароль. Цей тип отримав назву однофакторна автентифікація (рисунок.1.1). Вона поділяється на:

- логічні (паролі, ключові фрази, які вводяться з клавіатури комп'ютера чи клавіатури спеціалізованого пристрою);
- ідентифікаційні (носієм ключової інформації є фізичні об'єкти: дискета, магнітна карта, смарт-карта, штрих-кодова карта тощо);
- біометричні (аналіз унікальних характеристик людини, наприклад: відбитки пальців, малюнок райдужної оболонки ока, голос, обличчя).

Часто використовувані фактори включають перевірку за допомогою (1) чогось відомого користувачеві (наприклад, пароля), (2) чогось, що є у користувача (наприклад, смарт-карти або маркера безпеки), і (3) чогось, що користувач знає (наприклад, використання біометричних даних). Через їх підвищену складність, багатофакторні системи автентифікації важче зламати, ніж ті, що використовують будь-який один фактор.

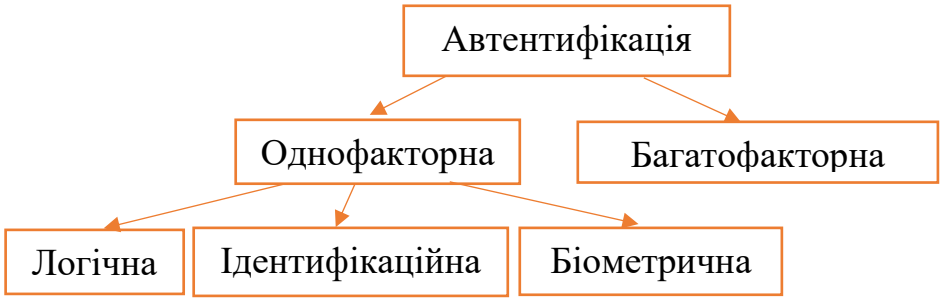


Рисунок.1.1 – Система технологій ідентифікації та автентифікації

Зазвичай багатофакторна автентифікація поділяється на двофакторну, трьохфакторну та чотирьохфакторну. Двофакторна автентифікація, як правило,

двоетапний процес, що складається з введення ім'я користувача і пароля та PIN-коду з фізичним або мобільним токеном. В трьохфакторній автентифікації додається фактор біометричні дані. Чотирьохфакторна автентифікація, до вже зазначених в трьхфакторній даних, додає місце розташування.

Багатофакторна автентифікація є одним з найбільш ефективних способів підвищення безпеки завдяки використанню декількох рівнів автентифікації, оскільки якщо один елемент пошкоджений, то обліковий запис користувача залишиться в безпеці.

Найбільшим недоліком багатофакторної автентифікації є збільшення складності управління як для адміністраторів, так і для кінцевих користувачів. Багатьом менш технічним користувачам може бути важко налаштувати та використовувати багатофакторну автентифікацію. Крім того, існує ряд інших поширених проблем:

- Типи багатофакторної автентифікації, які вимагають від користувачів певного обладнання, можуть спричинити значні витрати та адміністративні витрати.
- Користувачі можуть втратити доступ до своїх облікових записів, якщо вони втратять або не зможуть використовувати інші свої фактори.
- Багатофакторної автентифікація вносить додаткову складність у програму.
- Багато рішень багатофакторної автентифікації додають зовнішні залежності до систем, що може викликати вразливості безпеки або окремі точки збою.
- Процеси, реалізовані для того, щоб користувачі могли обійти або скинути таку автентифікацію, можуть бути використані зловмисниками.
- Вимагання багатофакторної автентифікації може перешкодити деяким користувачам отримати доступ до програми.

Також, автентифікація розрізняється за рівнем безпеки, що забезпечується об'єднанням чинників з однієї або декількох з трьох категорій факторів автентифікації:

1. «сильна» автентифікація;
2. електронна автентифікація;
3. безперервна автентифікація.

Сильна автентифікація використовує багатофакторну автентифікацію для підтвердження особи клієнта при вході в систему та авторизації транзакцій. У ньому також використовується автентифікація на основі ризику, яка допомагає запобігти шахрайству, визначаючи рівень ризику для кожної операції в режимі онлайн або мобільного банку та рівень автентифікації, необхідний кожної операції.

Електронна автентифікація — це процес, який використовується для підтвердження особистих даних користувача в електронному вигляді. Існують різні методи електронної автентифікації, які можуть використовуватися для автентифікації користувача, від пароля до більш високих рівнів безпеки, які використовують багатофакторну автентифікацію. Існує чотири типи схем електронної автентифікації [9]:

- локальна автентифікація (дані зберігаються локально і можуть бути скомпрометовані);
- централізована автентифікація (дозволяє кожному користувачеві використовувати одні і ті ж облікові дані для доступу до різних служб);
- глобальна централізована автентифікація (використання третьої сторони через глобальну централізовану схему автентифікації дозволяє користувачеві отримати прямий доступ до служб автентифікації);
- глобальна автентифікація і веб-додаток або портал (вважається найбезпечнішою схемою, включає мінімум два фактори, щоб дозволити доступ до необхідних службам і можливість підписувати документи).

Безперервна автентифікація – це метод підтвердження особи клієнта в режимі реального часу. Безперервна автентифікація використовує безліч потоків даних, що дозволяє механізму ризику оцінити та розпізнати унікальні рухи та моделі поведінки клієнта під час сесії. Клієнт не помічає, що його місцезнаходження, пристрій, оточення, ритм роботи клавіатури та багато іншого порівнюється з профілем того, як він зазвичай взаємодіє зі своїм телефоном та банківським додатком у рамках свого досвіду користувача.

Безперервна автентифікація не збільшує час, який витрачає клієнт на автентифікацію, за умови, що його поведінка не відхиляється від прийнятої моделі.

Якщо це так, система захисту від шахрайства запросить користувача до додаткової автентифікації.

Існує безліч джерел даних, які програма може надати механізму оцінки ризиків. Одним із прикладів є поведінкова біометрія, яка має широке визначення своєї функціональності. Поведінкова біометрія може бути взаємодією користувача з мобільним додатком, наприклад, як людина тримає телефон або як проводить пальцем по екрану.

Типи поведінкової біометрії, що використовуються для безперервної автентифікації [10]:

– Як людина тримає телефон: домінуюча рука, якою вона користується при розмові по телефону, та кут, під яким тримає телефон, аналізуються за допомогою поведінкової біометрії.

– Те, як людина друкує та як швидко вона друкує, визначає ритм натискання клавіш.

– Сила натискання пальців, яку людина використовує при наборі тексту, може бути зведена у відомий шаблон, що допоможе запобігти крадіжці особистих даних та знизити ризик шахрайства в Інтернеті.

– Моделі перегортання та прокручування враховують, чи проводить людина пальцем вправо або вліво сенсорним екраном пристрою і як прокручує сторінку вгору або вниз на пристрої.

– Хода, або те, як людина ходить, також є поведінковою ознакою, яку можна вивчити щодо виявлення закономірностей.

Все це дозволяє виявити аномалії в моделі поведінки клієнта з пристроєм і програмою. Крім того, поведінкова біометрія дозволяє виявити шкідливе програмне забезпечення таке, як боти, які можуть перехоплювати натискання клавіш людини, щоб розкрити її банківську інформацію, оскільки рухи бота відрізнятимуться від натискань клавіш людини.

Розглянемо детальніше методи автентифікації, а саме: статичної автентифікації за допомогою спільного секрету; апаратна автентифікація; криптографічна автентифікація на основі виклику-відповіді; радіочастотна

ідентифікація; і біометричні дані. Докладніше розглянемо їх використання, обмеження та потенційні переваги та недоліки.

1.3.1 Статична автентифікація за допомогою спільного секрету

Статична автентифікація найбільш відома через її основну реалізацію: автентифікацію на основі ідентифікатора/паролю, оскільки це широко використовуване рішення для перевірки ідентичності запитувача. У загальній схемі статичної автентифікації користувач доводить, що він знає загальну таємницю, як правило, відкриваючи її автентифікатору. Потім він порівнює дані зі значенням, збереженим у таблиці перевірки, щоб виявити, чи це правильне значення. Спільним секретом може бути:

– Пароль: список символів, значущих чи незначущих, зазвичай завдовжки від шести до десяти символів;

– Парольна фраза: часто використовується дуже подібно до пароля, але оскільки вона довша, вона максимізує простір для ключів, забезпечуючи кращу безпеку;

– Код доступу: це чисто цифровий пароль, який часто використовується для автентифікації власника смарт-картки (PIN-код);

– Нетекстові паролі: графічні паролі, засновані на русі миші;

– Велика кількість або масив випадково вибраних байтів: для автентифікації між машинами

Паролі, при правильному використанні, можуть забезпечити достатній рівень безпеки для багатьох організацій. Але, все ж таки, вони визнаються найслабшим засобом перевірки достовірності. Щоб значно підвищити надійність статичної автентифікації використовують наступні заходи:

– накладання технічних обмежень: встановлення мінімальної довжини пароля, використання в паролі різних груп символів (букви в двох регістрах, цифри, спец-символи і т.п.);

– встановлення терміну дії паролів, їх періодична зміна;

– обмеження кількості невдалих спроб входу в систему;

– використання програмних генераторів паролів.

Підвищення надійності пароля — це рішення, яке дозволяє уникнути атак зі словником або зробити атаки грубою силою нездійсненними. Загальновизнано, що довжина пароля визначає безпеку пароля, однак це не зовсім так: міцність пароля скоріше пов’язана з його ентропією. Наприклад, користувач вибирає, скажімо, пароль із семи символів, який забезпечує від шістнадцяти до двадцяти восьми біт ентропії.

Пароль, переданий від користувача до автентифікатора, без будь-якого шифрування вважається відомим паролем. Незважаючи на те, що ця методика широко використовується, вона має очевидну відсутність безпеки. Зловмисник може перехопити пароль і легко відтворити його, щоб отримати доступ до ІС. Тому головне занепокоєння паролів — це відсутність безпеки їх передачі по каналу, найкращий спосіб подолати цю проблему — це довести користувачу, що він/вона знає пароль, не надсилаючи його по каналу.

Недоліки статичної автентифікації виражаються загрозами та вразливостями, такими, як:

1. Отримання зловмисником паролю шляхом підбору або підглядання;
2. Ненавмисна передача пароля сторонній особі;
3. Отримання бази даних системи;
4. Зберігання пароля у відкритому вигляді та доступному місці;
5. Перехоплення паролю при передачі відкритими каналами зв’язку;
6. Програмні закладки;
7. Помилки в програмному забезпеченні.

Наступним фактором безпеки статичної автентифікації є спосіб зберігання паролю: у вигляді гешу або зашифрованим. При використанні гешування необхідно забезпечити відмінність значень гешу, отриманих з різних паролів та однакових. Для цього зазвичай використовують деяку випадкову інформацію(наприклад, псевдовипадкові числа). При шифруванні паролю велике значення мають спосіб генерації і зберігання ключа шифрування, наприклад: генерація ключа програмно і

зберігання в системі(або на зовнішньому носієві) або генерація ключа на основі вибраного адміністратором пароля, який вводиться в систему при кожному запуску[11]. Найбільш безпечне зберігання паролів забезпечується при їх гешуванні і подальшому шифруванні отриманих геш-значень.

1.3.2 Апаратна автентифікація

Апаратна автентифікація використовує предмет, ключ для визначення особистості. На сьогодні найбільш поширенні два типи пристроїв: токени та картки(безконтактні-карти, смарт-карти, магнітні карти).

Основним недоліком статичних паролів є їх відсутність захисту від атак повторного відтворення, отже, мета механізму одноразових паролів(OTP) полягає в тому, щоб знищити можливість повторного відтворення паролів із створенням нового пароля для кожного використання. Системи OTP можна розглядати як міст між автентифікацією статичного пароля та кращим методом автентифікації. Це полегшує міграцію застарілих програм, які були розроблені, щоб покладатися лише на паролі (мейнфрейми, веб-сайти, IC організації і т.д.). Єдиним компонентом, який потребує змін, є автентифікатор. Токен OTP зазвичай складається з пристрою з РК-дисплеєм (рідкокристалічний дисплей), на якому відображаються буквено-цифрові символи. Він може мати кнопку для створення одноразових паролів, а деякі блокуються PIN-кодом (чия клавіатура або безпосередньо на пристрої, або на зчитувачі), тому їх можна вважати двофакторною автентифікацією (рисунок. 1.2).

Оскільки OTP генерує пароль, для перевірки потрібна синхронізація між токеном і автентифікатором. Існує кілька категорій одноразових паролів, залежно від синхронізованих лічильників, синхронізованих за часом, з використанням безпечного каналу або зі спільним списком паролів.

Рисунок 1.2 – Приклад OTP токена.

Синхронізований із лічильником OTP, який іноді також називають «математичний геш-ланцюжок OTP» або HOTP, часто має на увазі токен з кнопкою на ньому. Кожне натискання кнопки генерує новий пароль, який можна використовувати для входу. Кожен раз, коли OTP запитується та перевіряється, коефіцієнт переміщення збільшується на основі лічильника. Згенерований код дійсний, доки ви активно не запитаєте інший і не буде перевірено сервером автентифікації. Генератор OTP і сервер синхронізуються щоразу, коли код перевіряється і користувач отримує доступ[12]. За допомогою HOTP одночасно можуть бути активними кілька активних OTP-кодів, це означає, що зловмисник може отримати один із кодів (наприклад, підслуховуючи), який ще не використовувався жертвою, і успішно увійти.

У синхронізованому за часом OTP маркер має внутрішній годинник і так само монітор. Нові паролі генеруються на основі значення поточної позначки часу, а не на основі спільного секретного або попереднього пароля. Час, протягом якого кожен пароль діє, називається кроком часу. Як правило, тимчасові кроки становлять 30 або 60 секунд після чого він більше не буде дійсним, і потрібно буде подати запит на новий, щоб отримати доступ до програми.

OTP також може бути надіслано заявнику через захищений канал. Користувача маже бути автентифіковано через незахищений канал, але

автентифікатор може надати йому випадковий одноразовий пароль через канал третьої сторони, який вважається захищеним. Потім заявник надсилає OTP через незахищений канал, щоб підтвердити свою особу на моніторі.

Програмні токени набагато більш поширені в сучасному світі, оскільки до них можна отримати доступ через мобільні програми, такі як Microsoft Authenticator або Google Authenticator.

Смарт-карта — це захищений мікрочіп, який зазвичай використовується для генерування, зберігання та роботи з криптографічними ключами. Автентифікація за допомогою смарт-карт працює за допомогою смарт-карт, пристроїв зі смарт-картками та програмного забезпечення для автентифікації. Користувачі підключають смарт-карту до хост-комп'ютера, програмне забезпечення на головному комп’ютері взаємодіє з матеріалом ключів та іншими секретами, що зберігаються на смарт-картці, для автентифікації користувача.

Смарт-карти вважаються дуже сильною формою автентифікації, оскільки криптографічні ключі та інші секрети, що зберігаються на картці, дуже добре захищені як фізично, так і логічно, і тому їх надзвичайно важко вкрасти.

Додаткова безпека, яку забезпечує смарт-карта, забезпечується за рахунок користувацького досвіду, оскільки смарт-карти потрібно фізично носити з собою та вставляти в головний комп’ютер щоразу, коли вони хочуть пройти автентифікацію з нею. Користувачі також обмежені хост-пристроями, на яких встановлено програмне забезпечення інтерфейсу карти.

Смарт-карти забезпечують підвищену безпеку в порівнянні з картками з магнітною смугою. Вони можуть містити мікропроцесори, які можуть обробляти дані безпосередньо без віддаленого підключення. Крім того, інформацію, що зберігається на смарт-картці, неможливо легко видалити, змінити або отримати. Навіть якщо смарт-карта потрапить у руки зловмисника, малоймовірно, що людина зможе створити копію та порушити безпеку. Дані смарт-картки можна оновлювати віддалено, не випускаючи нову картку. Такі картки не можна дублювати, оскільки вони зашифровані та мають унікальний ідентифікатор.

Незважаючи на безліч функцій, вбудованих у смарт-картки, вони мають деякі обмеження:

– Відсутність мобільності користувачів: мобільність користувачів — важливий аспект віддаленої роботи — можлива лише в тому випадку, якщо IT-адміністратори встановлять зчитувачі смарт-карт на кожному пристрої, до якого користувачі мають доступ в організації. На жаль, це неможливо, оскільки кінцеві пристрої повинні підтримувати ті самі стандартні інтерфейси зчитувача карток, або, в крайньому випадку, використовувати універсальний фірмовий зчитувач карт.

– Дорогі пристрої для зчитування карток: смарт-карти не зовсім дорогі, але зчитувачі карток. Початкові інвестиції в технологію смарт-карт можуть бути непосильними для стартапів з обмеженими фінансовими ресурсами.

– Повільна продуктивність: реалізація автентифікації за допомогою смарт-карт у деяких інфраструктурах може сповільнити їх продуктивність, особливо під час початкового завантаження, коли користувачі входять на свої робочі станції. Таким чином, IT-менеджери повинні переконатися, що їх апаратне забезпечення відповідає мінімальним специфікаціям для автентифікації смарт-карт.

– Загроза втрати: смарт-карти легкі, і користувачі можуть легко втратити їх або зламати, не помітивши. Таким чином, організації повинні мати надійні резервні заходи.

Існують дешевші варіанти смарт-карт що використовують магніти, штрих-коди та ін.

Ключ безпеки USB (також званий ключем U2F) — це тип апаратного захисту, який нагадує USB-накопичувач і підключається до одного з USB-портів комп’ютера. На практиці ключ безпеки є фізичним пристроєм безпеки з абсолютно унікальною ідентичністю. Він містить невеликий чіп з усіма протоколами безпеки та кодом, який дозволяє йому підключатися до серверів і підтверджувати вашу особу. Він використовується для того, щоб переконатися, що ви дійсно отримуєте доступ до сайту чи служби.

Деякі ключі безпеки навіть мають вбудовані NFC та/або Bluetooth, що робить їх ідеальними для використання з новішими смартфонами Android та iOS. Ключі

працюють із такими браузерами, як Google Chrome, а також із такими веб-сервісами, як Gmail, Facebook, Dropbox, 1Password, Twitter, GitHub, Microsoft та багатьма іншими.

Головна перевага таких ключів є також їх найбільша слабкість: це єдина точка доступу до ваших облікових записів. Тому там, де хакер майже не зможе отримати доступ до ваших облікових записів, він також унеможливить доступ до ваших власних облікових записів у випадку, якщо ви втратите ключ безпеки.

1.3.3 Автентифікація на основі криптографічного виклику-відповіді

Автентифікація виклик-відповідь використовує криптографічний протокол, який дозволяє довести, що користувач знає пароль, не розкриваючи сам пароль. Використовуючи цей метод, програма спочатку отримує випадковий виклик від сервера. Потім він обчислює відповідь, застосовуючи криптографічну геш-функцію до виклику сервера в поєднанні з паролем користувача. Нарешті, програма надсилає відповідь разом із оригінальним викликом назад на сервер. Через «односторонні» властивості геш-функції неможливо відновити пароль із відповіді, надісланої програмою.

Отримавши відповідь, сервер застосовує ту саму геш-функцію до виклику разом із власною копією пароля користувача. Якщо отримане значення відповідає відповіді, надісланій програмою, це з дуже високим ступенем ймовірності вказує на те, що користувач ввів правильний пароль.

Слід звернути увагу на те, що база даних користувачів не зберігає фактичні паролі користувачів. Замість цього зберігається криптографічний геш пароля в поєднанні з випадково згенерованим значенням солі. Це унеможливлює відновлення паролів користувачів у разі порушення безпеки сервера. Якщо користувач забуде свій пароль, ніхто не зможе сказати яким він був. Служба підтримки клієнтів може допомогти скинути пароль, але це все.

Оскільки сервер не має доступу до паролів із відкритим текстом, описаний вище протокол виклик-відповідь модифіковано, щоб використовувати геші паролів

замість справжніх паролів. Але передача гешованого пароля є проблемою через перебір і атаки за словником. Крім того, гешовані паролі містять багато інформації про секретний пароль.

Основною відповіддю на вирішення цієї проблеми є використання криптографії як симетричної, так і асиметричної для реалізації автентифікації типу «виклик-відповідь».

У симетричному випадку автентифікатор надсилає запит заявнику. Він може бути великим цілим числом або масивом цілих чисел.

Як правило, заявник потім шифрує виклик за допомогою загального ключа і надсилає результат назад до автентифікатора, який порівнює результат з тим, що він також обчислив. В асиметричному випадку позивач володіє приватним ключ, пов'язаний із сертифікатом, який містить відносний відкритий ключ. Заявник надає автентифікатору свій сертифікат (і його відкритий ключ).

Автентифікатор відправляє випадкове число як запит, і заявник використовує свій закритий ключ, щоб підписати виклик (або зашифрувати його), і надсилає результат назад автентифікатору. Потім автентифікатор перевіряє підпис з відкритим ключем (або розшифровує відповідь), щоб перевірити, чи виклик був перевірений.

Основною проблемою заявника є захист закритого ключа (або загального ключа) користувача. Для цього програмне забезпечення на комп'ютері заявника може запечатати закритий ключ паролем або кодовою фразою на пристрої користувача.

Інший рішення полягає у використанні захищеного елемента із закритим сховищем ключів: смарт-карти, захищеного USB-токена, TPM та ін.

Автентифікація на основі криптографічного виклику-відповіді є дуже ефективним і впевненим методом автентифікації, якщо він реалізований правильно. Однак з боку монітора обслуговування PKI (інфраструктури відкритих ключів) може забирати багато часу, енергії та грошей.

1.3.4 RFID та NFC

RFID (Radio Frequency IDentification) це метод контролю доступу, що використовує радіохвилі для передачі унікального ідентифікатора між міткою, вбудованою в RFID-карту, і зчитувачем RFID-карт.

Рішення контролю доступу на основі RFID містить як мітки, так і зчитувачі карток. Теги містять унікальний ідентифікатор і вбудовуються в пластикові картки або токени. Зчитувач RFID має антену, яка постійно випромінює радіочастотне поле короткої дії.

Щоб отримати доступ, користувач представляє свою RFID-карту до пристрою зчитування RFID-карт, який можна закріпити на стіні біля дверей або турнікета для фізичного доступу. Для доступу до комп'ютера або мережі зчитувач можна під'єднати через USB або вбудувати.

Теги можуть бути пасивними або активними. Пасивні мітки живляться лише від падаючої електромагнітної хвилі від зчитувача і, таким чином, мають менший робочий діапазон. Активні мітки живляться від батареї і можуть мати більший радіус дії, до сотень метрів.

Коли карта потрапляє в зону дії зчитувача, індукується електричний струм, який активує мітку. Потім тег повідомляє читачеві свій унікальний ідентифікатор, який можна використовувати для надання або заборони доступу. Ім'я користувача не потрібне, хоча від користувача можна за бажанням попросити ввести пароль або PIN-код для додаткової безпеки, якщо зчитувач RFID має клавіатуру.

Для автентифікації RFID можна використовувати низькочастотні (125 кГц) і високочастотні (13,56 МГц) пристрої. Зазвичай налаштування 125 кГц RFID дешевші у реалізації, але забезпечують меншу безпеку, оскільки в більшості випадків передається лише один числовий ідентифікатор.

Завдяки вищій частоті налаштування 13,56 МГц можуть підтримувати більш заповнений обмін даними, що дозволяє використовувати двонаправлений зв'язок, безконтактні смарт-картки та зв'язок ближнього поля (NFC). Деякі налаштування,

відомі як «подвійні режими» або «подвійні частоти», навіть підтримують обидві бездротові частоти для гнучкої та застарілої підтримки

Основні компоненти технології RFID [13]:

– RFID-мітки – це те, що зберігає та передає дані, які необхідно розшифрувати. Теги можна прикріпити до активів для надсилання даних на антену. Мікрочіп, вбудований у тег, є тим, що зберігає ідентифікатор тега та програмовані дані, пов’язані з активом. Ці збережені дані потім передаються до зчитувача через антени.

– Антени є необхідними елементами в системі RFID, оскільки вони передають дані RFID-мітки до зчитувача. Без певного типу антени RFID, незалежно від того, чи є вона вбудованою чи окремою, зчитувач RFID не може правильно надсилати та отримувати сигнали на мітки RFID.

– Зчитувачі RFID (рисунок 1.3) підключаються до антени і отримують дані з RFID-мітки. Зчитувач - це те, що приймає і перетворює радіохвилі в цифрові дані в комп'ютерній базі даних.Є два типи читачів. Існують фіксовані зчитувачі та мобільні зчитувачі. Фіксовані зчитувачі зазвичай кріпляться на стінах або інших об’єктах і залишаються в одному місці, щоб зчитувати дані, збережені в тегу. Мобільні зчитувачі можна встановити або перенести куди завгодно.

– Комп’ютерна база даних для обробки даних, що зберігаються в мітках. Це програмне забезпечення може програмувати теги, керувати пристроями та даними, віддалений моніторинг та конфігурацію обладнання.

–

Рисунок 1.3 – Приклад RFID зчитувача

Назвемо основні переваги технології RFID:

- не вимагає прямої видимості для сканування або ідентифікації;
- зчитувачі можуть прочитати сотні тегів за секунди;
- теги можуть бути переписані та використані повторно;
- дані тегів зашифровані, а також їх можна заблокувати для додаткової безпеки;
- мітки довговічні і витримують удари;
- теги можуть мати додаткову інформацію, надруковану на них, таку як інструкції, штрих-коди, або назви компаній;
- системи можна інтегрувати з іншими внутрішніми системами або процесами;
- теги можуть містити більше даних, ніж інші типи тегів або міток.

Відповідно основними недоліками RFID є:

- системи RFID можуть реагувати на певні події і фактори навколишнього середовища;
- початкова вартість системи вища, ніж при звичайному оптичному скануванні;
- компанії-партнери не можуть використовувати ту саму технологію, що спричиняє відключення;
- уразливість до програмних атак включаючи атаки підслуховування та повторного відтворення, атаки «людина посередині», а також клонування та спуфінг, але багато з цих загроз пом’якшуються за допомогою RFID-міток нового покоління, які використовують частоту 13,56 мгц;
- в разі пошкодження тега потрібна резервна система;
- загрози конфіденційності.

NFC або Near Field Communication також є бездротовою технологією, але, у порівнянні з RFID, вона забезпечує зв’язок на короткій відстані між сумісними пристроями. NFC потребує щонайменше одного передавального пристрою та іншого для прийому сигналу.

Однією з причин, чому легко сплутати технологію NFC і RFID, є те, що перша працює на частоті 13,56 МГц. Деякі високочастотні зчитувачі RFID працюють на однаковому частотному рівні – звідси і плутанина. Стандарти та протоколи NFC базуються на існуючих стандартах RFID, включаючи ISO/IEC 14443, FeliCa, ISO/IEC 18092, а також стандарти, визначені NFC Forum. Тобто, іншими словами, технологія NFC спирається на існуючий високочастотний RFID і часто використовується в рішеннях для контролю доступу [14].

Технологія NFC використовує тільки змінне магнітне поле, отже не випромінюється енергія у вигляді радіохвиль. Це запобігає виникненню будь-яких перешкод між подібними пристроями або будь-яким радіозв'язком, що працює на тій самій частоті. Система, заснована на технології NFC, зазвичай складається з ініціатора (зчитувача) і цілі (бирка, картка, наклейка або брелок). Теги NFC містять дані і, як правило, доступні лише для читання. Ці теги можуть безпечно зберігати особисті дані з пам’яттю від 96 до 8192 байт. Як і у випадку з технологією RFID, зв’язок NFC зазвичай поділяється на активну та пасивну.

Рисунок 1.4 – Приклад NFC зчитувача

Незважаючи на те, що обидві технології на перший погляд схожі, існує 5 ключових відмінностей між обома технологіями [14].

1. Технологія NFC працює на зменшеному діапазоні, який часто називають близькістю. RFID, з іншого боку, може зчитувати мітки на відстані до 10 м, що робить його найкращим рішенням для ідентифікації транспортних засобів і доступу.

2. NFC здатний до двостороннього зв'язку, тому може запропонувати унікальні та складні рішення, такі як емуляція карти та одноранговий зв'язок.

3. На відміну від RFID-міток, за допомогою технології NFC можна зчитувати лише одну мітку. Це може обмежити випадки його використання і означає, що мітки RFID часто краще підходять для середовищ, де є багато компонентів, які можна відстежувати.

4. Завдяки більшому простору для зберігання, пристрої NFC можуть зберігати та передавати більше даних, ніж пристрої RFID, які можуть переносити лише просту ідентифікаційну інформацію.

5. Через обмежений діапазон зчитування зчитувачі на основі NFC дешевші, ніж рішення RFID на великій відстані. Це робить NFC ефективнішим рішенням для компаній, які мають обмежений бюджет, але все ж хочуть використовувати високоякісне рішення.

1.3.5 Біометрична автентифікація

Біометрична автентифікація – це підтвердження людини за її біологічними ознаками. Вона не вимагає від користувача що-небудь приносити або запам'ятовувати для автентифікації, їм просто потрібно взаємодіяти з певною формою інтерфейсу. Методи біометричної автентифікації включають сканування відбитків пальців, розпізнавання обличчя, сканування сітківки ока та розпізнавання голосу. Найпоширенішим прикладом кожного з цих методів є розблокування мобільних пристроїв.

Біометричні характеристики можна розділити на три основні класи, а саме «морфологічні», «поведінкові» та «біологічні». Морфологічні пов'язані з формою тіла, наприклад сітківка, голос, відбитки пальців або долонь, райдужна оболонка ока, геометрія руки, розпізнавання обличчя, вух, шкіра, вени, стать та ін.

Поведінкові пов’язані з поведінкою людини, наприклад хода, підпис, динаміка натискання клавіш, голос, водіння. Біологічні пов’язані з внутрішньою частиною живого організму, серцебиття, запах, ДНК, кров та ін.

Голос можна класифікувати як за морфологічними, так і за поведінковими ознаками, оскільки кожна людина має різний голосовий тракт, але розпізнавання голосу в основному базується на вивченні способу розмови людини, тому зазвичай класифікується як поведінковий. Деякі дослідники ввели термін біхевіометрія для поведінкового класу біометрії.

Автентифікація за відбитками пальців є найбільш використовуваним методом біометричної автентифікації: у 2018 році було випущено понад мільярд смартфонів, які містять їх. Автентифікація відбитків пальців працює за допомогою алгоритму деталей (minutiae), зберігаючи візерунки відбитків пальців у вигляді точок у системі координат. Користувач налаштовує його, виконуючи початкове сканування свого відбитка пальця, щоб натиснути та отримати якомога більше ділянок відбитка пальця. Потім, коли справа доходить до автентифікації, користувач поміщає свій зареєстрований палець на сканер, і якщо хребти збігаються з системою координат, користувач буде автентифікований і отримає доступ. Використання сканерів відбитків пальців має високу надійність у порівнянні з іншими методами біометричної автентифікації. Швидше за все, це пов’язано з тим, що технологія набагато більш запроваджена і що немає двох людей з однаковими відбитками пальців.

Розпізнавання обличчя працює так само, як і алгоритм деталей, але замість того, щоб переносити візерунок в систему координат, він дивиться на риси обличчя (рисунок. 1.5), дві особливості, які вважаються особливо важливими, - це відстань між очима людини та відстань від чола до підборіддя [15]. Автентифікація на обличчі також може використовуватися як безконтактний метод автентифікації, оскільки користувачеві не потрібно встановлювати фізичний контакт для проведення автентифікації, натомість це камера, яка зчитує риси обличчя користувачів і порівнює їх із внутрішньою базою даних функцій, і якщо збігається, автентифікація проводиться успішно.

Рисунок 1.5 – Приклад схеми розпізнавання обличчя

В розпізнавані обличчя є певні недоліки, а саме: загроза приватності, помилкові спрацювання(хоча це дуже рідко), злочинці можуть обдурити розпізнавання обличчя, надягаючи маски або маскування обличчя, старіння знижує ефективність, а також інші дослідження показали, що розпізнавання обличчя менш ефективне для ідентифікації кольорових людей і жінок [15].

Сканери сітківки ока — це метод автентифікації, який сканує сітківку ока користувача. Основним методом автентифікації за допомогою сканування сітківки є використання візерунка кровоносних судин у сітківці(рисунок 1.6), ці моделі є унікальними для кожної людини [16]. Як і автентифікація за допомогою розпізнавання обличчя, цей метод також можна вважати безконтактним методом автентифікації, оскільки око користувача сканується камерою, а візерунки кровоносних судин узгоджуються з тими, які зберігаються в системній базі даних. Початкове зображення, відскановане сканерами сітківки ока, розбивається на різні етапи, щоб прискорити відповідність шаблонів.

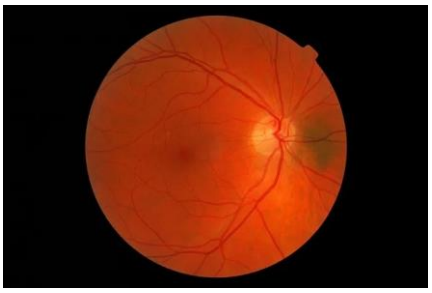


Рисунок 1.6 – Візерунок кровоносних судини у сітківці ока

Біометрична система автентифікації на основі сітківки ока має свій набір сильних і слабких сторін. Сильні сторони можна описати наступним чином [16]:

1) Рисунок кровоносних судин сітківки майже ніколи не змінюється протягом життя людини.

2) Оскільки сітківка розташована на задньому кінці ока, вона не піддається впливу зовнішнього середовища, на відміну від інших органі, таких як відбитки пальців, геометрія руки, обличчя тощо.

3) Стабільні та відмінні риси сітківки можуть витягуватися з судинної структури.

4) У порівнянні з іншими біометричними характеристиками, середній розмір вектора ознак сітківки є дуже малим, що може призвести до швидшої перевірки та обробки ідентифікації. Тоді як вектори ознак більшого розміру можуть уповільнити час обробки.

Недоліки такого способу автентифікації:

1) Якщо у людини є якісь очні захворювання, такі як тверда глаукома, катаракта тощо, то процес ідентифікації буде дуже складним.

2) Отримання зображення передбачає співпрацю суб'єкта, тягне за собою контакт з окуляром, а також технологія сканування сітківки ока не для людей, які носять контактні лінзи.

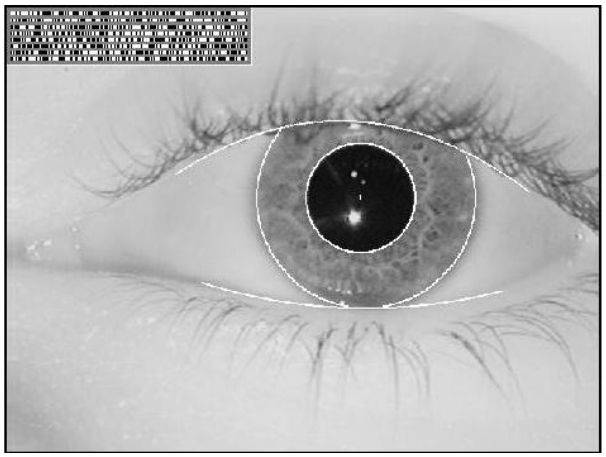
3) Структура кровоносних судин сітківки може розкрити деякі медичні стани цієї людини, які є ще одним фактором, що перешкоджає суспільному прийняттю системи біометричної автентифікації на основі сітківки.

4) Пристрої, які використовуються для отримання зображень сітківки ока, дуже дорогі у придбанні та впровадженні

Розпізнавання райдужної оболонки – це процес використання видимого та ближнього інфрачервоного світла для отримання висококонтрастної фотографії райдужної оболонки ока людини. Сканування райдужної оболонки збирає близько 240 біометричних ознак, об’єднання яких є унікальним для кожного ока.

Біометричні сканери розпізнавання райдужної оболонки ока працюють шляхом освітлення райдужної оболонки невидимим інфрачервоним світлом, щоб виявити унікальні візерунки, які не видно неозброєним оком. Сканери райдужної оболонки виявляють і виключають вії, повіки та дзеркальні відблиски, які зазвичай

блокують частини райдужки. Кінцевим результатом є набір пікселів, що містить лише райдужну оболонку. Далі аналізується шаблон ліній і кольорів ока, щоб



отримати бітовий шаблон, який кодує інформацію в райдужній оболонці. Цей бітовий шаблон оцифровується і порівнюється зі збереженими шаблонами в базі даних для перевірки (відповідність шаблону один до одного) або ідентифікації (відповідність шаблону один до багатьох). Приклад такого сканування зображено на рисунку 1.7.

Рисунок 1.7 – Приклад сканування райдужки ока

Автентифікація на основі голосу це технологія, яка дозволяє користувачам отримувати доступ до сервісів за допомогою мови. Голосові характеристики, які використовуються для автентифікації, мають низку реалізацій від зіставлення на мобільному пристрої до повторної автентифікації по телефону з корпоративним кол-центром або до пристроїв інтернету речей. Характеристики голосу часто вимірюються за допомогою визначення живості, коли користувачеві пропонується вимовити унікальну фразу для поточної транзакції, або їх можна виміряти пасивно і забезпечити сильний додатковий рівень безперервної безпеки.

Через недосконаліть голосових відбитків система призначена для використання як другий фактор багатофакторної автентифікації. Сервер зберігає голосові функції, пов’язані з обліковим записом користувача, на етапі реєстрації, коли кілька зразків звуку від одного користувача використовуються для вилучення функцій і створення загального голосового відбитка для користувача.

Велика проблема біометричної автентифікації за допомогою голосу полягає в тому, що «біометрична спільнота ще не встановила верхніх меж для біометрії обличчя та голосу» [17]. Неможливо знати, скільки типів голосу існує і чи є люди, які говорять так само. Також неможливо виміряти, наскільки може змінитися голос людини, а це означає, що можна помилково вважатися кимось іншим, а хтось може імітувати чужий голос. Це одне з найбільш відкритих питань у сфері біометричної автентифікації за допомогою голосу.

Отже, основні недоліки такої автентифікації:

1) Вразливість до злому: Коли система зберігає вокальні дані, вона робить це так само, як і інші дані: на сервері чи базі даних. Якщо ця точка не захищена, хакери можуть вкрасти дані та використати їх для створення підроблених записів.

2) Не можна замінити: якщо біометричні дані скомпрометовані, їх не можна замінити, як пароль, тому що не можна просто змінити свій голос.

3) Не на 100% точний. Жоден метод автентифікації не є надійним. Це так само справедливо і для розпізнавання мовлення. Проте галузеві інновації в галузі штучного інтелекту та запобігання спуфінгу зробили біометричні дані життєздатними навіть для безпечних додатків, таких як обробка платежів.

4) Залежність від середовища: для розпізнавання мовлення потрібна досить тиха зона. Звукові артефакти за межами голосу користувача можуть заважати автентифікації, що може спричинити проблеми, якщо використовується лише голос.

Висновки до розділу 1

Автентифікація та авторизація є важливими складовими забезпечення безпеки. Вони дозволяють доступ лише верифікованим особам або машинам. На сьогодні існує багато методів та стандартів, які регулюють автентифікацію, але всі вони мають свої недоліки.

Найпопулярнішим методом автентифікації вважається пароль, але без використання додаткових факторів він не є надійним. Адміністратор системи має

враховувати всі фактори загроз і обрати таке поєднання методів автентифікації, що має забезпечити найвищий рівень безпеки. Привілеї та права, надані користувачу, залежать від дозволів, що надані адміністратором. Тому, окрім звичайної автентифікації потрібно постійно стежити за діями користувача (безперервна автентифікація), адже зловмисник завжди на крок попереду.

Розглянуті методи можуть поєднуватись для забезпечення багатофакторної автентифікації і зменшення загроз безпеці. Зрозуміло, що незважаючи на проблеми з деякими методами автентифікації, багатофакторна автентифікація, яка використовується в тій чи іншій формі, є більш ефективною для захисту, ніж використання лише імені користувача та пароля.

РОЗДІЛ 2
АНАЛІЗ ВРАЗЛИВОСТЕЙ ТА МЕТОДІВ ЗАХИСТУ АВТЕНТИФІКАЦІЇ ТА АВТОРИЗАЦІЇ

2.1 Основні вразливості автентифікації та авторизації

Вразливості автентифікації – це проблеми, які впливають на процеси автентифікації та роблять веб-сайти та програми сприйнятливими до атак безпеки, під час яких зловмисник може маскуватися під законного користувача.

Існує кілька способів, через які можуть виникнути вразливості автентифікації, найпоширенішими з яких є ігнорування проблемних зон, помилки в коді чи логіці та неправильний вибір користувача, який може поєднуватися з двома попередніми. Якщо програма має поганий захист від грубої сили, зловмисники можуть скористатися цим, щоб отримати доступ навіть до добре захищених облікових записів або дешевий масовий доступ до погано захищених. Так само, якщо є логічні помилки або помилки кодування, зловмисники можуть обійти частину або весь процес автентифікації.

Розглянемо найпоширеніші вразливості на основі автентифікації:

1. Слабкий захист від атаки грубої сили.

Атака грубої сили - це спроба отримати незаконний доступ до системи або облікового запису користувача шляхом введення великої кількості випадково згенерованих або попередньо згенерованих комбінацій імен користувачів і паролів, доки не знайдуть ефективний.

Якщо є недоліки в системі захисту від грубої сили, наприклад, в логіці автентифікації, брандмауері або протоколі безпечної оболонки (SSH), зловмисники можуть захопити облікові дані та процеси, що поставить під загрозу безпеку облікових даних користувачів.

2. Слабкі паролі.

Використання передбачуваних даних може призвести до вразливостей у процесі автентифікації. Передбачувані імена користувачів можуть полегшити зловмисникам націлювання на конкретних користувачів.

Замість того, щоб використовувати повну атаку грубої сили, зловмисники будуть шукати облікові записи з легкими паролями, які використовуються занадто часто. Вони спробують використовувати звичайні облікові дані, як-от "admin", "admin1" і "password1". Без обмежень щодо слабких паролів навіть сайти, захищені від атак грубої сили, можуть виявитися скомпрометованими.

3. Перебір імені користувача.

Перебір імені користувача не є вразливістю автентифікації. Але це може полегшити життя зловмиснику, знизивши вартість інших атак, таких як атаки грубої сили або слабкі перевірки облікових даних.

Проблема з перебором імен користувачів полягає в тому, що зловмисники можуть визначити, які імена користувачів є дійсними. Потім вони можуть спробувати зламати дійсні облікові записи користувачів за допомогою методів грубої сили, не витрачаючи час і гроші на перевірку безлічі недійсних імен облікових записів.

4. Слабке управління сесіями.

Вразливість в управлінні ідентифікаторами сеансів призводить до викрадення дійсних автентифікованих сеансів. Це одна з поширених веб-вразливостей для обходу паролів.

Існує кілька вразливостей, пов’язаних із неправильним керуванням сеансом, як-от відсутність тайм-аутів сеансу, розкриття ідентифікаторів сеансів у URL-адресах, файли cookie сеансу без встановленого прапорця Http-Only та погане визнання сеансу недійсним.

Якщо зловмисники можуть захопити контроль над наявним сеансом, вони легко потрапляють в систему, приймаючи ідентичність вже автентифікованого користувача, повністю минаючи процес аутентифікації.

5. Зберігання сеансу.

Прапорець «Запам’ятати мене» або «Зберегти мене в системі» під формою входу дозволяє дуже легко залишатися в системі після закриття сеансу. Він генерує файл cookie, який дозволяє пропустити процес входу.

Однак це може призвести до вразливості автентифікації на основі файлів cookie, якщо зловмисник може передбачити файл cookie або визначити його шаблон генерації. Вони можуть використовувати шкідливі методи, такі як атаки грубої сили, щоб передбачити файли cookie, і міжсайтові сценарії (XSS), щоб зламати облікові записи користувачів, дозволяючи зловмисному серверу використовувати законні файли cookie.

6. Небезпечна зміна та відновлення пароля.

Процес скидання пароля створює вразливість автентифікації, якщо програма використовує слабкий механізм відновлення пароля, як-от прості таємні запитання, відсутність CAPTCHA або електронні листи зі скиданням пароля з занадто довгими або відсутніми тайм-аутами.

Якщо функція відновлення пароля є несправною, зловмисники потенційно можуть використовувати методи грубої сили або доступ до інших зламаних облікових записів, щоб отримати доступ до облікових записів користувачів і облікових даних, які добре захищені за звичайних обставин.

7. Вразлива логіка автентифікації.

Логічні вразливості поширені в програмних додатках. Це відбувається в результаті поганого кодування або дизайну, що впливає на автентифікацію та авторизаційний доступ, а також на функціональність програми.

Погана логіка програми може бути викликана зловживанням функціональністю, слабкими заходами безпеки або пропущеним кроком у процедурі перевірки.

8. Людська недбалість.

Згідно зі звітом Shred-it 2020 [18], до 31% керівників C-suite повідомили, що недбалість співробітників є другою основною причиною порушення їх даних. Людська помилка може призвести до серйозних вразливостей автентифікації, якими

набагато легше скористатися, ніж атаки грубої сили, ін’єкції SQL та обхід автентифікації. Ця недбалість включає такі дії, як:

- залишати комп’ютер увімкненим і розблокованим у громадському місці;
- втрата пристроїв через крадіжку;
- витік конфіденційної інформації незнайомцям;
- написання поганого коду.

9. SQL Injection

Ін’єкція SQL – це вектор атаки, який використовує шкідливий код SQL неочікуваним способом для маніпулювання та доступу до бази даних. Ін’єкції SQL можуть увімкнути атаки на механізми автентифікації шляхом крадіжки відповідних даних (наприклад, погано захищених гешів паролів) із незахищеної бази даних. Вони також можуть обійти механізми автентифікації, якщо введений код SQL виконується внутрішнім (і вже авторизованим) інструментом, який не зміг достатньо перевірити зовнішній вхід.

10. Слабка двофакторна автентифікація.

Хоча двофакторна автентифікація ефективна для безпечної автентифікації, вона може викликати критичні проблеми з безпекою, якщо її не запровадити належним чином.

Зловмисники можуть визначити чотири- та шестизначні коди підтвердження двофакторна автентифікації за допомогою атак із заміною SIM-карти, якщо вони надіслані через SMS. Деяка двофакторна автентифікація також не є справді двофакторною; якщо користувач намагається отримати доступ до конфіденційної інформації на вкраденому телефоні за допомогою кешованих облікових даних, «другий фактор», який надсилає повідомлення на той самий телефон, не додає додаткової безпеки.

Уразливості двофакторної автентифікації також можуть виникнути, якщо немає захисту від грубої сили для блокування облікового запису після певної кількості спроб входу.

Атаки на біометричні системи, що використовують вищеописані вразливості:

1. Атаки на рівні обробки та передачі.

Оскільки багато біометричних систем передають зразки даних на локальні або віддалені робочі станції для обробки, також важливо, щоб ця передача була безпечною, щоб передача не була перехоплена, прочитана або змінена. Більшість біометричних систем шифрують дані під час передачі, але не всі програми та пристрої підходять для шифрування. Розробники повинні оцінити ступінь, до якого зразки даних можуть бути доступні під час передачі або під час зберігання, і вони повинні визначити застосовні методи безпеки системи та найкращі методи. Наприклад, заходи проти спуфінгу, шифрування даних під час передачі та застосування відповідних резервних методів є критичними аспектами безпеки біометричної системи. Ці методи можна вдосконалити за рахунок впровадження багатофакторної автентифікації та рандомізації.

2. Атаки на вхідному рівні.

Основні атаки на рівні вхідних даних це спуфінг та обхід, оскільки використовуються вразливості в точці отримання даних та початкової обробки. Хоча підробка є найчастішою уразливістю на рівні входу, інші вразливості на рівні вхідних даних можуть бути такими ж проблематичними, як-от «перевантаження». «Перевантаження» — це спроба перемогти або обійти систему, пошкодивши пристрій введення або перевантаживши його, намагаючись створити помилки. Це також іноді називають атакою переповнення буфера для інших механізмів безпеки. Прикладом такого типу атаки для біометричної системи може бути швидке спалахування яскравого світла на оптичні датчики відбитків пальців або пристрої для розпізнавання обличчя, які можуть порушити їх належне функціонування. Силіконові датчики можна легко пошкодити, замикаючи їх або обливши водою.

Оскільки, багато біометричних систем покладаються на чутливе обладнання, яке можна відносно легко перевантажити, користувачі можуть мати можливість викликати збій пристрою або системи. Системи повинні бути спроектовані таким чином, щоб у разі перевантаження основні функції не мали збоїв. А коли біометричні пристрої більше не можуть виконувати свою цільову функцію, необхідно визначити та застосувати резервні процеси. Людина, яка викликає збій біометричної системи, може робити це, знаючи, що, як наслідок, двері без охорони

можуть використовуватися як тимчасовий альтернативний засіб входу. Системи безпеки повинні враховувати потенційну функціональну несправність біометричних систем і пристроїв за допомогою відповідних заходів резервного копіювання.

3. Атаки на внутрішні системи

Атака на базу даних сховища шаблонів є найбільш очевидним типом серверної атаки. Загроза несанкціонованої модифікації або заміни збережених шаблонів може призвести до помилкового прийняття або помилкового відхилення залежно від мотивів зловмисника. Якщо зловмисник може знайти спосіб ввести шаблони безпосередньо в базу даних сховища, то зловмисник може ввести його/її в систему без дотримання відповідних процедур реєстрації. Зловмисник також може захопити ідентичність уповноваженої особи, замінивши оригінальний шаблон своїм власним шаблоном, тим самим зберігаючи привілеї, пов’язані з уповноваженою особою. Якщо шаблон зламаний, його можна повторно використати в атаці повторного відтворення. Хоча обхід атак повторного відтворення розглядається в попередньому розділі, компрометація збережених шаблонів є однією з найважливіших загроз, які слід враховувати при розробці розподіленої біометричної системи.

Ці види атак можна запобігти за допомогою методологій шифрування та гешування. Застосування загальних методологій безпеки баз даних також може підвищити рівень складності для зловмисника. Зловмисник може змінити або замінити відповідну підсистему або підсистему рішень, щоб вона давала результат за бажанням зловмисника. Це серйозна загроза в мережевому середовищі. Таку атаку можна обійти, застосовуючи методи безпеки, такі як перевірка цілісності коду та принципи створення надійних систем. Атака «Відмова в обслуговуванні» (DOS), спрямована на внутрішні підсистеми, також є дуже реальною загрозою. Перевантаження процесорів внутрішньої підсистеми надлишковим трафіком може призвести до недоступності сервісів. Аналіз трафіку та моніторинг трафіку зазвичай використовуються для запобігання атакам DOS.

4. Атаки під час реєстрації

Даний тип атак полягає в довірі до процесу перевірки заявленої особистості особи, впевненість у дійсності пов’язаних документів і надійність у справжності виданих електронних облікових даних. Приклади таких атак:

- Реєстрація дійсних біометричних даних особи зі створеною чи заміненою особою. У цьому сценарії особа використовує/реєструє власні біометричні дані під фальшивою або припущеною ідентичністю, що згодом дозволяє цій особі отримати несанкціонований доступ та здійснювати транзакції електронної комерції та інші логічні та/або фізичні активи, такі як комп’ютери, мережі, бази даних, програми та об'єктів.
- Реєстрація замінених або замінених біометричних даних (не їх власних) разом із дійсним ідентифікатором, який згодом може бути використаний третьою стороною для маскування та отримання доступу до систем електронної комерції та/або інших логічних чи фізичних активів.
- Реєстрація підмінених або фальшивих біометричних даних з фальшивою або несправжньою ідентичністю, яку згодом можна використовувати для отримання доступу до системи.
- Змова з оператором реєстрації. У цьому сценарії можна сприяти будь-якому з перерахованого вище, а також несанкціонованому введенню або модифікації записів системних даних або введення до них.
- Зовнішні атаки на станцію реєстрації та/або інші компоненти системи, з якими вона взаємодіє. Приклади включають спуфінг, передачу через прослуховування, «Людина посередині» та Replay.

2.2 Методи захисту автентифікації та авторизації

Хоча вразливості автентифікації легко визначити, вони сильно впливають на кібербезпеку. Розглянемо методи запобігання вразливостям на основі автентифікації та збереження критичної інформації в безпеці:

1. Впровадження надійної системи захисту від грубої сили: атак грубої сили можна запобігти шляхом блокування облікових записів, обмеження швидкості, моніторингу на основі IP, брандмауерів програм і CAPTCHA.

2. Застосування безпечної політики паролів: використання засобу перевірки паролів, який повідомляє користувачам, наскільки надійні їхні паролі в режимі реального часу. Також впровадження автентифікації без пароля за допомогою таких стандартів, як FIDO2, щоб зменшити ризик і стрес, пов'язаний з керуванням паролями.

3. Застосування суворої безпеки транспорту HTTP (HSTS): це змушує веб-сеанси використовувати шифрування TLS, запобігаючи доступ до конфіденційної інформації під час передачі.

4. Вимкнення перерахування імен користувачів: генеруючи ту саму помилку для помилки входу, незалежно від того, чи було ім'я користувача дійсним чи недійсним, ви змушуєте зловмисника використовувати грубу силу не лише до паролів, а й до імен користувачів, що робить перебір складнішим.

5. Змінення заголовків файлів cookie: зміна заголовків cookie захищає їх від зловмисних атак. Використання тегів HttpOnly і SameSite під час налаштування заголовків cookie запобігає атакам XSS(Cross-Site Scripting) і CSRF(Cross-Site Request Forgery) відповідно.

6. Перевірка кодування: це важливо для виявлення будь-яких вразливостей та їх усунення. Існує статичний та динамічний аналіз.

7. Додання другого фактору: хоча це і ускладнює користування це підвищує безпеку в рази. В ідеалі двофакторна автентифікація має бути реалізована за допомогою спеціального пристрою або програми, яка безпосередньо генерує код підтвердження. Оскільки вони спеціально створені для забезпечення безпеки, вони, як правило, більш безпечні.

2.3 Визначення оптимального методу для багатофакторної автентифікації та авторизації

Проблема безпеки автентифікації полягає в великій кількості вразливостей та особливого місця в забезпеченні конфіденційності даних. Тому, на основі вищеописаних даних проведемо порівняльну характеристику методів автентифікації.

Таблиця 2.1

Порівняння методів автентифікації

Засіб	**Вартість**	**Безпека**	**Мобільність**	**Простота в користуванні**	**Спец. ПЗ**	**Додаткові особливості**
Пароль	безкоштовно	низька	-	Висока	Не потребує	Вразливість до багатьох атак
USB-ключі	Залежить від виробника.	висока	Є	Середня	Потребує	Загроза втрати
OTP-токени	Залежить від виробника.	Висока	Є	Середня	Потребує сервер автентифікації	Обмежений час використання
Смарт-карти	Висока	Висока	Є	Середня	Так	Потребує зчитувальний пристрій, загроза втрати карти
Програмні токени	безкоштовно	середня	-	Висока	Так	Уразливість до програмних атак

продовження табл.2.1

RFID та NFC	середня	Середня	Є	Середня	Так	Уразливість до програмних атак та середовища
Біометрика	Дуже висока	Відносно Висока	Залежить від реалізації	Низька	Так	Людська недосконалість

Отже, можна зробити висновок, що більшість методів хоч і забезпечують високу безпеку є не зручними в користуванні та потребують додаткових витрат. Тому, необхідно розробити систему, яка використовуватиме підтвердження у вигляді фрази або речення, що надсилаються на електронну пошту.

В даному методі захист надісланого листа покладається на постачальника послуги. В самому листі неявно вказано, яке речення має бути використано в підтвердженні входу. Даний підхід унеможливлює атаки грубої сили, повторного входу та багатьох інших, оскільки фраза буде потрібна при кожному вході або зміні паролю. Таким чином, навіть перехопивши облікові дані, зловмисник не зможе ввійти без підтвердження з листа.

Запропонований метод вважається двофакторною автентифікацію, що значно підвищує рівень безпеки додатка. Переваги такого методу: стійкість до багатьох атак, не потребує додаткових витрат та спеціального пз. Недоліками можна вважати залежність від поштового оператора (Google, Microsoft).

Висновки до 2 розділу

В даному розділі досліджено вразливості автентифікації та атаки, які їх використовують. Розглянуті вразливості автентифікації є в кожній системі чи концепції. На сьогодні реалізується велика кількість атак, саме тому неможливо в

повній мірі забезпечити захист використовуючи лише один засіб автентифікації. Тому зараз найбільшої популярності набувають методи багатофакторної автентифікації, що були проаналізовані в розділі 1.4.

Було запропоновано метод з використанням електронної пошти, який можна використовувати разом з статичною автентифікацією. Дане поєднання значно підвищує безпеку даних.

РОЗДІЛ 3
ПОБУДОВА ПРОГРАМНОГО МОДУЛЯ

3.1 Опис технічних засобів

Для реалізації поставленої задачі було запропоновано метод автентифікації, який поєднує статичну автентифікацію та електронну пошту. Для розробки програмного модуля було обрано наступні засоби:

- Python:

Python — це інтерпретована мова загального призначення високого рівня динамічного програмування. Його структура дизайну зосереджена на читабельності коду разом із змістовним використанням значних відступів.

Python сильний на серверних платформах і настільних комп’ютерах, а отже, це ідеальна мова програмування на стороні сервера..

У порівнянні з Java та C, для мови програмування Python потрібно менше кроків. Побудова його мови та підхід, спрямований на об'єкт, допомагає програмістам писати логічні та зрозумілі коди для малих і великомасштабних проектів.

Мова Python надає користувачеві величезну бібліотеку. Стандартна бібліотека Python є величезною, і майже всі функції, які потрібно виконати, доступні в її бібліотеці. Це тому, що вона має величезну підтримку спільноти та корпоративного спонсорства. Бібліотеки, такі як NumPy, SciPy і Matplotlib, дозволяють ефективно використовувати Python у наукових обчисленнях, зі спеціалізованими бібліотеками, такими як Biopython і Astropy, які забезпечують функціональні можливості для певної області. SageMath — це система комп'ютерної алгебри з інтерфейсом ноутбука, програмованим на Python: його бібліотека охоплює багато аспектів математики, включаючи алгебру, комбінаторику, числову математику, теорію чисел і обчислення. Python зазвичай використовується в проектах зі штучним інтелектом і

в проектах машинного навчання за допомогою таких бібліотек, як TensorFlow, Keras, Pytorch і Scikit-learn.

GNU Debugger використовує Python як гарний принтер для показу складних структур, таких як контейнери C++. Esri рекламує Python як найкращий вибір для написання скриптів в ArcGIS. Він також використовувався в кількох відеоіграх і був прийнятий як перша з трьох доступних мов програмування в Google App Engine, двома іншими є Java та Go.

Багато операційних систем включають Python як стандартний компонент. Він постачається з більшістю дистрибутивів Linux, AmigaOS 4 (з використанням Python 2.7), FreeBSD (як пакет), NetBSD і OpenBSD (як пакет) і може використовуватися з командного рядка (термінал). У багатьох дистрибутивах Linux використовуються інсталятори, написані на Python.

Python широко використовується в індустрії інформаційної безпеки, включаючи розробку експлойтів [21].

- Visual Studio:

Visual Studio — це інтегроване середовище розробки (IDE) від Microsoft. Він використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-додатків, веб-сервісів та мобільних додатків. Visual Studio використовує такі платформи розробки програмного забезпечення Microsoft, як Windows API, Windows Forms, Windows Presentation Foundation, Windows Store і Microsoft Silverlight.

Visual Studio включає редактор коду, який підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. Інтегрований налагоджувач працює і як налагоджувач на рівні джерела, і як налагоджувач на рівні машини. Інші вбудовані інструменти включають профайлер коду, конструктор для створення програм із графічним інтерфейсом користувача, веб-дизайнер, конструктор класів і дизайнер схем баз даних. Він приймає плагіни, які розширюють функціональність майже на кожному рівні, включаючи додавання підтримки систем контролю джерел (наприклад, Subversion і Git) та додавання нових наборів інструментів, таких як редактори та візуальні конструктори для мов, що стосуються домену, або наборів

інструментів для інших аспектів життєвого циклу розробки програмного забезпечення (наприклад, клієнт Azure DevOps : Team Explorer)

Visual Studio є одним із середовищ розробки, що дозволяє працювати з Python. Перевагою даної IDE в порівнянні, наприклад, з PyCharm, слід зазначити перш за все те, що в її безкоштовній редакції VS 2019 Community доступні ряд функцій та можливостей, які в тому ж PyCharm доступні лише у платній версії Professional Edition. Наприклад, це веб-розробка за допомогою різних фреймворків. У той же час засоби для розробки на Python у Visual Studo доступні поки що у версії для Windows.

3.2 Реалізація програми

Для реалізації програми було обрано клієнт-серверну архітектуру. На стороні клієнта проводиться отриманих даних з форми автентифікаціїї та надсилання на сервер даних. На стороні сервера відбувається створення бази даних користувачів, обробка запитів та надсилання повідомлень на пошту для проведення автентифікації та авторизації.

Встановлення з’єднання з сервером реалізовано звичайним сокетом для наглядної демонстрації. В реальності, має бути реалізовано захищене з’єднання через TLS (Transport Layer Security) з використанням інфраструктури відкирих ключів.

Далі відбувається обмін даними з сервером. Частина цього процесу зображена на рисунку 3.1. Основна задача клієнта відправка на сервер введених користувачем даних та отримання відповіді про результат автентифікації.

База даних на сервері реалізована за допомогою модуля Pickle в Python. В основному він використовується для серіалізації та десеріалізації структури об'єкта Python.

```
client.send((name+'+'+ hashlib.sha256((name + passw).encode('ascii')).hexdigest()).encode('ascii'))
answer = client.recv(1024).decode('ascii')
if answer == "Wrong":
    print("Name or pass dont correct")
    client.close()
    return False
if answer == "Correct":
    cn = 0
    while cn < 3:
        client.send(input("Enter email phrase").encode('ascii'))
        answer = client.recv(1024).decode('ascii')
        if answer == "555" : return [name, 555]
        elif answer == "777" : return [name, 777]
        elif answer == "NotOK" :
            print("Wrong")
            cn += 1
            continue
```

Рисунок 3.1 – Програмний код обміну повідомленнями

Іншими словами, це процес перетворення об’єкта Python в потік байтів для збереження його у файлі/базі даних, підтримки стану програми впродовж сеансів або транспортування даних по мережі. Вибраний потік байтів можна використовувати для повторного створення оригінальної ієрархії об’єктів шляхом видалення потоку. Весь цей процес подібний до серіалізації об’єктів у Java або .Net. Створення БД зображено на рисунку 3.2. Для цього створено окрему функцію.

```
def LoadList():
    if path.isfile('adata.db'):
        with open('adata.db','rb') as f:
            db = pickle.load(f)
    else: # Create some database to test this program..
        db = {"Admin": [haslib.sha256("Test123".encode("ascii")).hexdigest(), "777", "shramenko3@gmail.com"] }
        with open('adata.db', 'wb') as f:
            pickle.dump(db, f)
    return db
```

Рисунок 3.2 – Створення бази даних

Процес автентифікації відбувається за наступним алгоритмом:

1. Отримання від клієнта повідомлення з іменем користувача та гешованим паролем.

2. Розділення гешу та імені користувача.

3. Перевірка наявності імені користувача в базі даних. Якщо немає, надсилання відповіді про невдачу.

4. Порівняння даних від клієнта з обчисленими на сервері.

5. Надсилання повідомлення на пошту та перевірка отриманого від клієнта.

6. Подальша обробка запитів.

Даний алгоритм зображено у вигляді схеми в ДОДАТКУ А та у вигляді коду, що подано на рисунку 3.3.

```
while True:
    client, client_addr = server.accept()
    print('Connected by', client_addr)
    try:
        data = client.recv(1024).decode('ascii')
        msg = data.split('+')
        if db[msg[0]] == False:  client.send("Wrong".encode('ascii'))
        if hashlib.sha256((msg[0]+db[msg[0]][0]).encode('ascii')).hexdigest() == msg[1] :
            client.send("Correct".encode('ascii'))
            key = ''.join([random.choice(list('123456789qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) for x in range(10)])
            sendmail(key, db[msg[0]][2])
            mailkey = client.recv(1024).decode('ascii')
            count = 0
            while count < 3:
                if key == mailkey:
                    client.send(db[msg[0]][1].encode('ascii'))
```

Рисунок 3.3 – Автентифікація

Надсилання пошти реалізовано у вигляді окремої функції, що зображена на рисунку 3.4. Процес надсилання листа відбувається з використанням захищеного з'єднання TLS. Для цього існує модуль smtplib. Модуль smtplib визначає об'єкт сеансу клієнта SMTP, який можна використовувати для надсилання пошти на будь-яку Інтернет-машину з демоном прослуховування SMTP або ESMTP. В даному модулі функція starttls переводить SMTP-з'єднання в режим TLS. Усі наступні

```
def sendmail(message, to):
    msg = MIMEMultipart()
    password = [redacted]
    msg['From'] = "dreamornightmares@gmail.com"
    msg['To'] = to
    msg['Subject'] = "Do Not Answer"
    msg.attach(MIMEText("Одноразовий код:" + message, 'plain'))

    server = smtplib.SMTP('smtp.gmail.com: 587')
    server.starttls()

    server.login(msg['From'], password)
    server.sendmail(msg['From'], msg['To'], msg.as_string())

    server.quit()
```

команди SMTP будуть зашифровані.

Рисунок 3.4 – Функція надсилання пошти

Для наглядної демонстрації на пошту надсилається одноразовий код (рисунок 3.5), замість парольної фрази (речення), як було запропоновано в попередньому розділі.

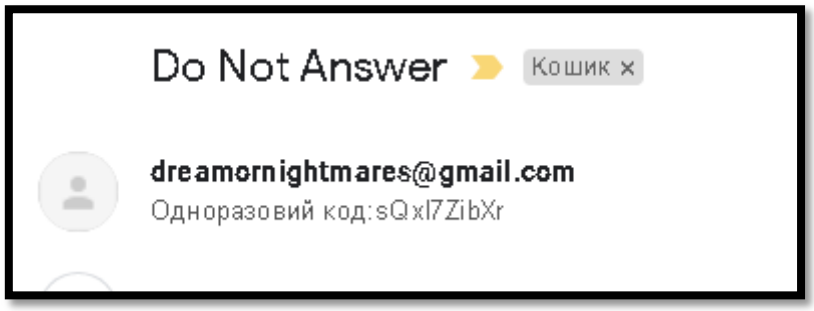


Рисунок 3.5 – Приклад надісланого повідомлення

3.3 Безпека інформації в модулі

Для забезпечення конфіденційності даних користувачів було застосовано наступні технології:

- Гешування: Паролі в базі даних зберігаються у вигляді гешу, перевірка повідомлення на сервері відбувається гешуванням логіна па пароля. Використовується метод SHA-256. Він являє собою однонаправлену функцію для створення цифрових відбитків фіксованої довжини (256 біт, 32 байт) із вхідних даних розміром до 2,31 ексабайт (2^{64} біт) і є приватним алгоритмом із сімейства криптографічних алгоритмів SHA-2 (Secure Version Hash Algorithm 2) опублікованим АНБ США у 2002 році.
- Шифрування: Файл бази даних зберігається на сервері у зашифрованому вигляді з використанням AES або шифру Калина. Реалізація цих шифрів виходить за межі проекту, але в концепцію програми входить. Калина — блочний симетричний шифр описаний у національному стандарті України ДСТУ 7624:2014 «Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення». Стандарт набрав чинності з 1 липня 2015 року наказом Мінекономрозвитку від 2 грудня 2014 року №1484. Застосування сього стандарту в модулі дозволить використовувати його у держустановах. Було виявлено деякі атаки на скорочені варіанти шифру[20], але вони не є практичними, тому є очевидним використати саме його. Ідеальним варіантом було б застосування гомоморфного шифрування — це форма шифрування, яка дозволяє користувачам виконувати обчислення зашифрованих даних без попереднього їх розшифрування.

- TLS: криптографічний протокол, що призначений для забезпечення безпеки зв'язку в комп'ютерній мережі. Протокол TLS має на меті забезпечити криптографію, включаючи конфіденційність, цілісність та автентичність за допомогою сертифікатів, між двома або більше комп’ютерними програмами, що спілкуються. Він виконується на прикладному рівні і сам складається з двох шарів: запису TLS і протоколів рукостискання TLS. В даній програмі використаний для встановлення з’єднання з поштовим сервером та обміну даними між клієнтом та сервером модуля.
- Логування: Логи — це файли, які містять системну інформацію про роботу сервера або комп’ютера та певні дії користувача або програми. Їх призначення — запис операцій, які виконують на сервері, для подальшого аналізу цих операцій адміністратором. Постійний аналіз журналів дозволить визначити помилки в роботі системи, діагностувати зловмисну активність, зібрати статистику. Функція логування представлено на рисунку 3.6.

```
def log(msg):
    logging.basicConfig(
        level=logging.DEBUG,
        filename = "mylog.log",
        format = "%(asctime)s - %(module)s - %(levelname)s - %(funcName)s: %(lineno)d - %(message)s",
        datefmt='%H:%M:%S',
    )
    logging.info(msg)
```

Рисунок. 3.6 – Функція логування

Висновки до 3 розділу

У цьому розділі окреслено межі реалізованого прототипу. Аспекти, на яких вирішено не акцентувати увагу під час впровадження, це UX, естетичний вигляд, відновлення облікового запису. Ці аспекти не входили в сферу реалізації прототипу, оскільки було вирішено зосередитися на демонстрації основ багатофакторної автентифікації. Крім того, вирішено не надавати пріоритет цим аспектам через обмежену тривалість цього дипломного проекту.

Що стосується досвіду користувача, було створено додаток, щоб продемонструвати двофакторну автентифікацію, не приділяючи занадто багато

уваги тому, що користувач може відчувати, використовуючи прототип. Крім того, є інші дрібні деталі, які потрібно буде виправити, перш ніж буде розгорнуто готове для клієнта рішення.

Що стосується естетичного вигляду розробленого додатку, використовувався консольний проект, і, таким чином, деякі естетичні аспекти були пропущені або можуть бути покращені.

Відновлення облікового запису описано в розділі 2.1. Однак не було впроваджено жодного із запропонованих методів для обробки відновлення облікового запису, оскільки вважалося, що це виходить за рамки реалізації прототипу.

У прототипі програми були розглянуті та реалізовані такі аспекти безпеки, як гешування паролів, запобігання ін’єкції SQL та Грубої сили, блокування користувача після кількох невдалих спроб входу та обмеження часу дії емейл-коду.

Крім того, запропоноване рішення є функціональною реалізацією прототипу. Воно має на меті представити, як можливий метод двофакторної автентифікації може працювати.

ВИСНОВОК

Автентифікація – це процес перевірки ідентичності зареєстрованого користувача перед наданням доступу до захищеного ресурсу. Авторизація є процесом перевірки того, що автентифікований користувач має права на доступ до ресурсів.

Існують різні методи, за допомогою яких можна було б реалізувати автентифікацію, а саме: статична автентифікація за допомогою спільного секрету; апаратна автентифікація; криптографічна автентифікація на основі виклику-відповіді; радіочастотна ідентифікація; і біометричні дані.

В дипломній роботі представлене програмне рішення з використанням статичної автентифікації та електронної пошти. Реалізація рішення дозволяє уникнути болісної міграції до двофакторної автентифікації, оскільки запропонований прототип дозволяє збільшити розгортання, забезпечуючи при цьому бажане рішення з точки зору низької вартості та підвищення безпеки. Більше того, оскільки це програмне рішення, воно не вимагає, щоб користувач мав додатковий пристрій, оскільки він може працювати на існуючому мобільному пристрої користувача з програмою електронної пошти. Незважаючи на багато переваг, дане рішення також має недоліки. Основним недоліком є проблеми щодо відновлення облікового запису.

В результаті виконання дипломної роботи були вирішені наступні завдання:

- Проаналізовано основні методи автентифікації та авторизації.
- Проаналізовано вразливості систем автентифікації та авторизації.
- Визначено оптимальний метод безпечної багатофакторної автентифікації та авторизації.
- Розроблено програмний модуль автентифікації та авторизації з використанням засобів та механізмів захисту.

Під час виконання дипломного проекту виникли певні обмеження. Основне обмеження стосується аналізу існуючих готових рішень. Через обмежений час і

недостатні знання, аналізувати інші готові рішення, крім описаного, було складно. Іншим обмеженням була відсутність оцінки UX(Досвід користувача) при застосуванні даного методу автентифікації. Хоча іноді для визначення очікуваного UX можна було покладатися на наявні дослідження, в інших випадках доводилося робити деякі припущення, засновані на факторах, які впливають на UX. Можна було б уникнути цих припущень, провівши спеціально розроблене дослідження, але обмежений час зробив це нездійсненним. Це означає, що запропоноване рішення може бути неоптимальним з точки зору UX.

Всі поставлені задачі були виконані в повному обсязі. Головна мета роботи була досягнута. Розроблений модуль забезпечить захищений доступ до сервісів, що не мають коштів на дорогі рішення, але піклуються про безпеку даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Review on Authentication Method [Електронний ресурс]. – Режим доступу: https://hal.archives-ouvertes.fr/hal-00912435/PDF/A_Review_on_Authentication_Methods.pdf

2. "An Overview of Different Authentication Methods and Protocols[Електронний ресурс]. – Режим доступу: https://www.sans.org/reading-room/whitepapers/authentication/overview-authentication-methods-protocols-118

3. Автентифікація телекомунікаційних мереж та мереж передачі даних [Електронний ресурс]. – Режим доступу: https://web.archive.org/web/20160304080620/http://data.cedupoint.cz/oppa_e-learning/2_KME/044.pdf

4. Extensible Authentication Protocol [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Extensible_Authentication_Protocol#EAP-MD5

5. Extensible Authentication Protocol [Електронний ресурс]. – Режим доступу: https://doubleoctopus.com/security-wiki/protocol/extensible-authentication-protocol/

6. "What is an OpenID? [Електронний ресурс]. – Режим доступу: http://openid.net/get-an-openid/what-is-openid/

7. ДСТУ ISO/IEC 9798-1:2015 Інформаційні технології. Методи захисту. Автентифікація об’єктів. Частина 1. Загальні положення (ISO/IEC 9798-1:2010, IDT) [Електронний ресурс]. – Режим доступу: http://online.budstandart.com/ua/catalog/doc-page?id_doc=66917

8. NIST Special Publication 800-63B [Електронний ресурс]. – Режим доступу: https://pages.nist.gov/800-63-3/sp800-63b.html

9. Electronic authentication [Електронний ресурс]. – Режим доступу: https://wikiukuk.top/wiki/Electronic_authentication

10. Безперервна автентифікація [Електронний ресурс]. – Режим доступу: https://www.onespan.com/ru/topics/continuous-authentication

11. Дослідження стійкості системи парольного захисту[Електронний ресурс]. – Режим доступу: http://antibotan.com/file.html?work_id=521527

12. What’s the Difference Between OTP, TOTP and HOTP [Електронний ресурс]. – Режим доступу: у https://www.onelogin.com/learn/otp-totp-hotp

13. RFID: The Technology Making Industries Smarter [Електронний ресурс]. – Режим доступу: https://blog.ttelectronics.com/rfid-technology

14. RFID vs. NFC [Електронний ресурс]. – Режим доступу: https://blog.nortechcontrol.com/rfid-vs-nfc

15. What is facial recognition? How facial recognition [Електронний ресурс]. – Режим доступу: https://us.norton.com/internetsecurity-iot-how-facial-recognition-software-works.html

16. RETINA BASED BIOMETRIC AUTHENTICATION SYSTEM: A REVIEW Jarina B. Mazumdar, S. R. Nirmala

17. Biometric Authentication using Voice [Електронний ресурс]. – Режим доступу: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.237&rep=rep1&type=pdf

18. Vulnerabilities of Biometric Authentication “Threats and Countermeasures” [Електронний ресурс]. – Режим доступу: https://www.ripublication.com/irph/ijict_spl/ijictv4n10spl_01.pdf

19. Shred-it Data Protection Report [Електронний ресурс]. – Режим доступу: :https://www.shredit.com/content/dam/shred-it/global/documents/Shred-it_2020-Data-Protection-Report_US.pdf.coredownload.inline.pdf

20. Improved Meet-in-the-Middle Attacks on Reduced-Round Kalyna-128/256 and Kalyna-256/512 [Електронний ресурс]. – Режим доступу: https://eprint.iacr.org/2016/722.pdf

21. "Core Security". Core Security [Електронний ресурс]. – Режим доступу: https://www.coresecurity.com/

22. D. M’Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, ‘HOTP: An HMAC-Based One-Time Password Algorithm’. Request for Comments, Dec-2005 [Електронний ресурс]. – Режим доступу: http://www.rfc-editor.org/rfc/rfc4226.txt

23. The ultimate mobile email statistics overview [Електронний ресурс]. – Режим доступу: https://www.emailmonday.com/mobile-email-usage-statistics/

24. W. Bedford, J. Gregg, and S. Clinton, ‘Implementing Technology to Prevent Online Cheating: A Case Study at a Small Southern Regional University (SSRU)’, J. Online Learn. [Електронний ресурс]. – Режим доступу: http://jolt.merlot.org/vol5no2/gregg_0609.pdf

25. ‘German Defense Minister von der Leyen’s fingerprint copied by Chaos Computer Club’ [Електронний ресурс]. – Режим доступу: http://www.dw.com/en/germandefense-minister-von-der-leyens-fingerprint-copied-by-chaos-computer-club/a18154832

26. What is two-factor authentication (2FA)? [Електронний ресурс]. – Режим доступу: https://securenvoy.com/what-is-2fa/

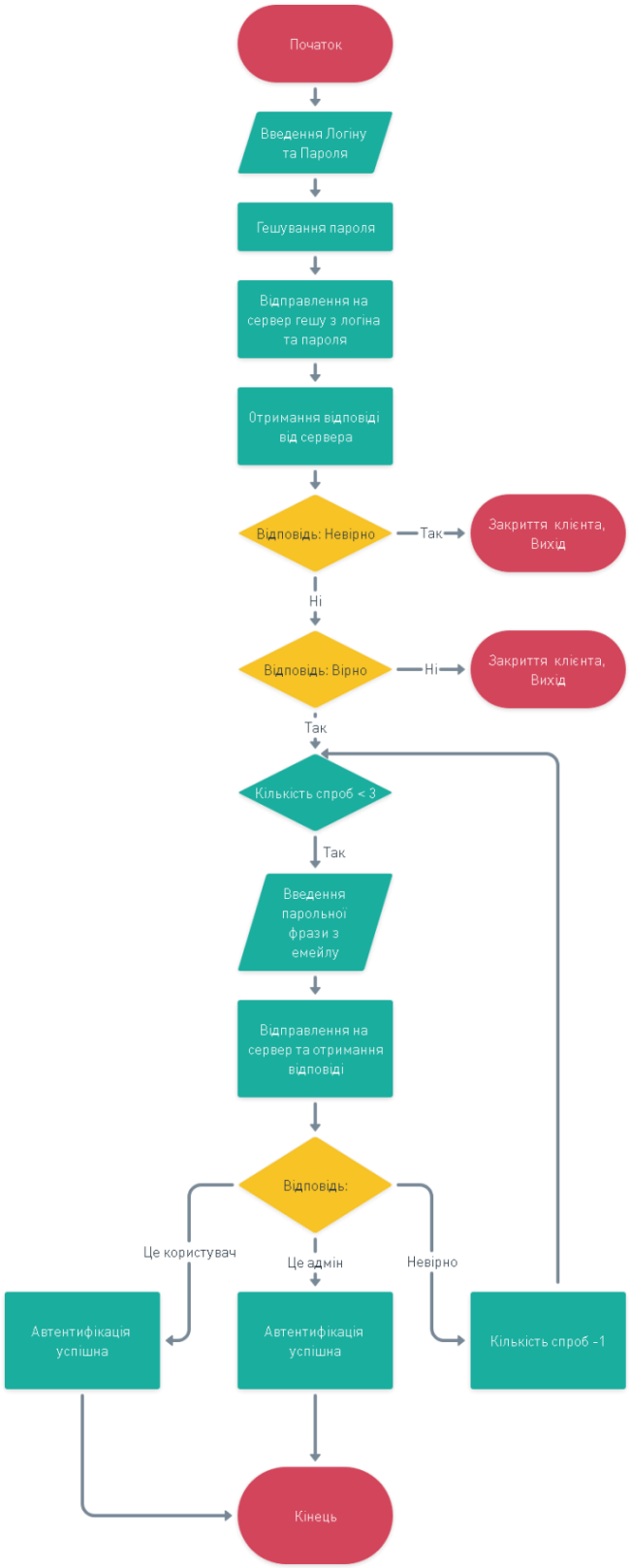
27. TLS/SSL wrapper for socket objects)? [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/ssl.html

28. SMTP protocol client [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/smtplib.html

29. hashlib — Secure hashes and message digests [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/hashlib.html?highlight=hashlib#module-hashlib

30. pickle — Python object serialization [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/pickle.html?highlight=pickle#module-pickle

ДОДАТОК А



Схеми алгоритмів модуля

Рисунок 1– Алгоритм клієнтської частини

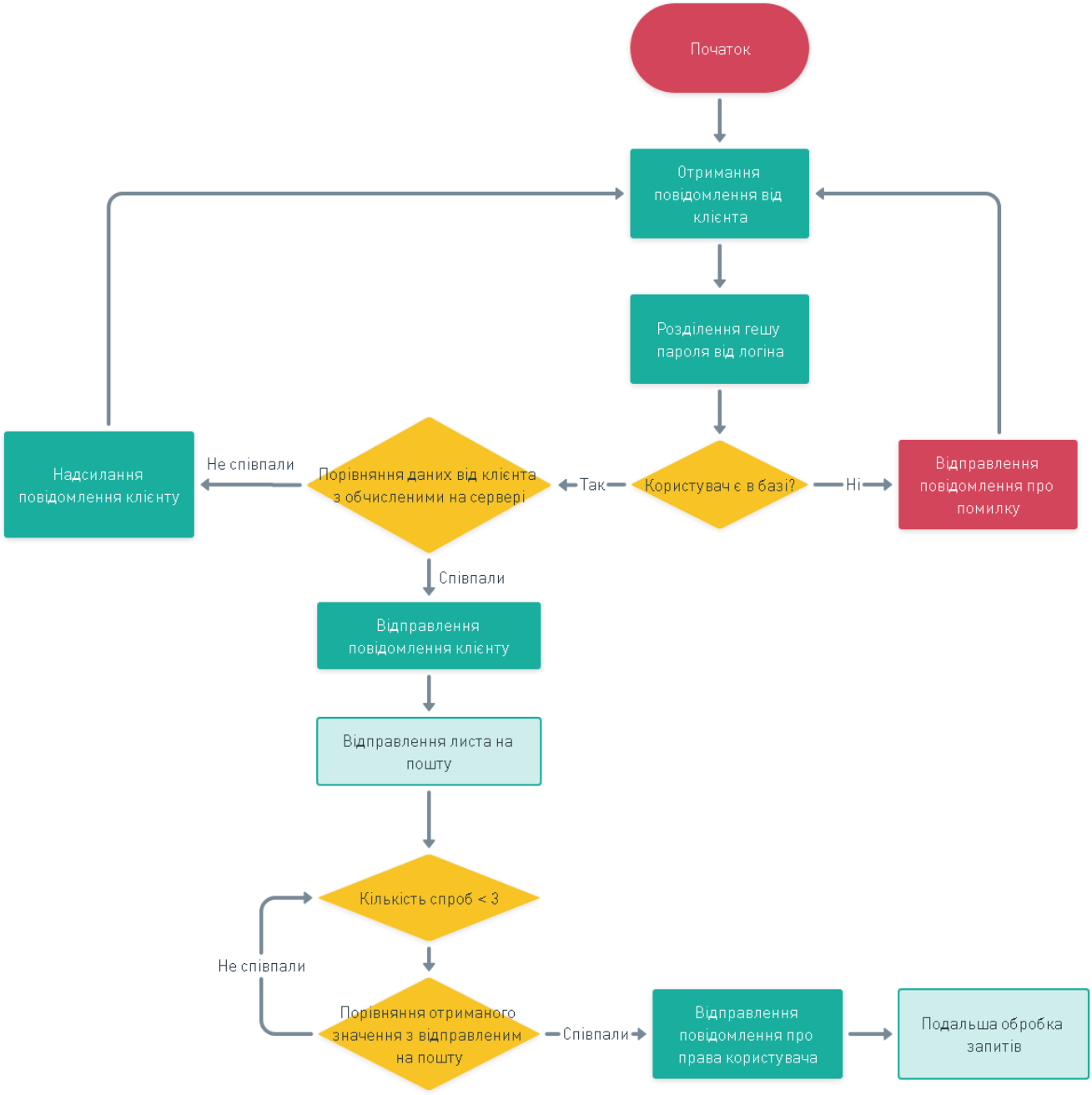


Рисунок 2 – Алгоритм серверної частини

ДОДАТОК Б

Програмний код модуля автентифікації та авторизації

Частина 1. Клієнт:

```
import socket
import hashlib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import smtplib
from datetime import datetime, date
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('192.168.1.8', 1234))
def changepass():
    cn = 0
    while cn <3:
        ps = input("NewPass: ")
        u, d, l, c, = 0,0,0,0
        L = len(ps)
      #Перевірка довжини паролю
        if L <= 9:
            cn +=1
            print("too short")
            continue
        else:
            symb = list(ps)
            for i in  symb:
                if i.isupper():u+=1
                elif i.isdigit():d+=1
                elif i.islower():l+=1
```

```
            else: c+=1
        if u != 0 && d != 0 && l!=0 && c != 0:
         print("Pass changed")
         client.send(("1"+"+"+name).encode('ascii'))
        else :
         print("Pass must contain upper lower digits and specials symb!!!!!")
         continue
       return True

def listusers():
    client.send("2"+"+l".encode('ascii'))
    print(client.recv(1024).decode())
    return True
def delluser():
        name = input("Name User to delete: ")
    client.send(("3"+"+"+name).encode('ascii'))
def adduser():
        name = input("Name User to delete: ")
    r = input("role: ")
    e = input("email: ")
    client.send(("4"+"+"+name+"+"+r+"+"+e).encode('ascii'))
    return True
def authenticate():
    name = input("Введіть імя: ")
    passw = haslib.sha256(input("Введіть падоль: ").encode("ascii"))hexdigest()
    client.send((name+'+'+ hashlib.sha256((name +
passw).encode('ascii')).hexdigest()).encode('ascii'))
    answer = client.recv(1024).decode('ascii')
    if answer == "Wrong":
        print("Name or pass dont correct")
```

```
        client.close()
        return False
    if answer == "Correct":
        cn = 0
        while cn < 3:
            client.send(input("Enter email phrase:").encode('ascii'))
            answer = client.recv(1024).decode('ascii')
            if answer == "555" : return [name, 555]
            elif answer == "777" : return [name, 777]
            elif answer == "NotOK" :
                print("Wrong")
                cn += 1
                continue
            else:
                print("Something wrong")
                client.close()
                return False
    else:
        print("Something wrong")
        client.close()
        return False
```

Частина 2. Сервер:

```
import socket
import hashlib
import random
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import smtplib
import sys
import pickle
```

```
from os import path
import logging
def sendmail(message, to):
    msg = MIMEMultipart()
    password = "w5zHm6h86d5B"
    msg['From'] = "dreamornightmares@gmail.com"
    msg['To'] = to
    msg['Subject'] = "Do Not Answer"
    msg.attach(MIMEText("Одноразовий код:" + message, 'plain'))
    server = smtplib.SMTP('smtp.gmail.com: 587')
    server.starttls()
    server.login(msg['From'], password)
    server.sendmail(msg['From'], msg['To'], msg.as_string())
    server.quit()
    print ("successfully")
def LoadList():
    if path.isfile('adata.db'):
        with open('adata.db','rb') as f:
            db = pickle.load(f)
    else: # Create some database to test this program..
        db = {"Admin": [haslib.sha256("Test123".encode("ascii")).hexdigest(), "777",
"shramenko3@gmail.com"] }
        with open('adata.db', 'wb') as f:
            pickle.dump(db, f)
    return db
def read_users():
    db = LoadList()
    with open('adata.db','rb') as f:
        db = pickle.load(f)
    return db
```

```
def delete_user(user):
# try to delete the given user, handle if the user doesn't exist.
    db = LoadList()
    try:
        del db[user]
    except KeyError:
        print("{} Користувача не існує в db".format(user))
        print(db)
    # write the 'new' db to the file.
    with open('adata.db', 'wb') as f:
        pickle.dump(db, f)
def addUser(user, passwd):
    db = LoadList()
    db[user] = passwd
    with open('adata.db','wb') as f:
        pickle.dump(db, f)
def log(msg):
         logging.basicConfig(
          level=logging.DEBUG,
                filename = "mylog.log",
                format = "%(asctime)s - %(module)s - %(levelname)s -
%(funcName)s: %(lineno)d - %(message)s",
          datefmt='%H:%M:%S',    )
    logging.info(msg)
def serv():
    db = LoadList()
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(("192.168.1.6", 1234))
    server.listen(10)
    while True:
```

```
        client, client_addr = server.accept()
        print('Connected by', client_addr)
        log("New Connection: "+ client_addr)
        try:
            data = client.recv(1024).decode('ascii')
            msg = data.split('+')
            if db[msg[0]] == False:
                    client.send("Wrong".encode('ascii'))
                    log("Worg loging name!")
            if hashlib.sha256((msg[0]+db[msg[0]][0]).encode('ascii')).hexdigest() ==
msg[1] :
                client.send("Correct".encode('ascii'))
                log("User {} entered.".format(msg[0]))
                key =
".join([random.choice(list('123456789qwertyuiopasdfghjklzxcvbnmQWERTYUIOP
ASDFGHJKLZXCVBNM')) for x in range(10)])
                sendmail(key, db[msg[0]][2])
                mailkey = client.recv(1024).decode('ascii')
                count = 0
                while count < 3:
                    if key == mailkey:
                        client.send(db[msg[0]][1].encode('ascii'))
                        #спиок обробки запитів редагурання бд
                        client.close()
                        break
                    else:
                        client.send("NotOK".encode('ascii'))
                        log("User {} entered wrong message".format(msg[0]))
                        count += 1
            else:
```

```
                client.send("Wrong".encode('ascii'))
            except ConnectionResetError:
                print("Error")
                client.close()
                continue
if __name__=="__main__":serv()
```