



**POLITECHNIKA
BYDGOSKA**
Wydział Telekomunikacji,
Informatyki i Elektrotechniki

**PRACA DYPLOMOWA
MAGISTERSKA**
na kierunku informatyka stosowana

**Analiza i implementacja wielowarstwowego uwierzytel-
nienia (MFA) z wykorzystaniem specyfikacji FIDO 2**

**Analysis and implementation of multi-layer authentication (MFA)
using the FIDO 2 specification**

Adam Szreiber
121515

Promotor dr inż. Jacek Majewski

Bydgoszcz, luty 2025

Metryka pracy dyplomowej

Dane ogólne

Nazwa Uczelni	Politechnika Bydgoska im. Jana i Jędrzeja Śniadeckich
Wydział	Telekomunikacji, Informatyki i Elektrotechniki
Kierunek	Informatyka Stosowana
Tryb studiów	niestacjonarne
Dane autora	Adam Szreiber, 121515
Dane promotora	dr inż. Jacek Majewski

Dane dotyczące pracy dyplomowej

Język pracy	język polski [PL]
Tytuł pracy	Analiza i implementacja wielowarstwowego uwierzytelnienia (MFA) z wykorzystaniem specyfikacji FIDO 2
Opis pracy	Przegląd literaturowy w zakresie implementacji metod autentyfikacji - poziomy zabezpieczeń i ich wdrożenia. Techniczne porównanie działania mechanizmów autentyfikacji. Projekt i implementacja autentyfikacji z wykorzystaniem specyfikacji FIDO 2 dla usługi poczty elektronicznej (środowisko badawcze dla specyfikacji FIDO 2). Weryfikacja i analiza przepływu informacji uwierzytelniającej (np. w kanale komunikacji klient-serwer). Analiza i warstwowe testy zabezpieczeń. Wnioski końcowe dotyczące warunków wdrożenia dla różnych platform pocztowych.
Typ pracy	magisterska
Streszczenie	W niniejszej pracy autor dokonał kompleksowego przeglądu literatury w zakresie metod uwierzytelniania, koncentrując się w szczególności na autentykacji opartej na sekrecie oraz specyfikacji FIDO2. W ramach analizy, ocenił poziom bezpieczeństwa, możliwości implementacji oraz wskazał potencjalne słabości każdej z metod. Szczegółowo omówił kluczowe elementy FIDO2, a także identyfikuje czynniki uwierzytelniające. Podstawę do badań stanowił zaimplementowany serwis mailowy, wykorzystujący uwierzytelnianie FIDO2 jako mechanizm zabezpieczający. Przeprowadzone badania obejmowały analizę komunikacji klient-serwer, weryfikację skuteczności procesu autentykacji, porównanie wdrożonego rozwiązania z obowiązującymi standardami technicznymi, a także wykazanie odporności mechanizmu na ataki typu phishing.
Słowa kluczowe	FIDO 2, MFA, klucz bezpieczeństwa, WebAuthn

Diploma thesis record

General information

University name	Bydgoszcz University of Science and Technology
Faculty	Telecommunications, Computer Science and Electrical Engineering
Field of study	Applied Computer Science
Mode of study	part-time
Author's information	Adam Szreiber, 121515
Supervisor's information	Jacek Majewski, BEng, PhD

Data regarding the diploma thesis

Language of thesis	Polish [PL]
Title of thesis	Analysis and implementation of multi-layer authentication (MFA) using the FIDO 2 specification
Description	Literature review on the implementation of authentication methods - security levels and their implementation. Technical comparison of the operation of authentication mechanisms. Design and implementation of authentication using the FIDO 2 specification for the e-mail service (research environment for the FIDO 2 specification). Verification and analysis of the flow of authentication information (e.g. in the client-server communication channel). Analysis and layered security tests. Final conclusions regarding the implementation conditions for various e-mail platforms.
Type of thesis	Master's
Abstract	In this paper, the author conducted a comprehensive review of the literature on authentication methods, focusing in particular on secret-based authentication and the FIDO2 specification. As part of the analysis, he assessed the level of security, implementation possibilities, and indicated potential weaknesses of each method. He discussed in detail the key elements of FIDO2 and identified authentication factors. The basis for the research was an implemented email service using FIDO2 authentication as a security mechanism. The research included analysis of client-server communication, verification of the effectiveness of the authentication process, comparison of the implemented solution with applicable technical standards, and demonstration of the mechanism's resistance to phishing attacks.
Keywords	FIDO 2, MFA, security key, WebAuthn

Spis treści

Spis treści	4
1 Wstęp	6
1.1 Cel pracy	7
2 Przegląd literatury	8
2.1 Analiza metod autentyfikacji	8
2.2 Podział mechanizmów	9
2.3 Metody autentyfikacji	10
2.3.1 Hasła	10
2.3.2 Klucze bezpieczeństwa	12
2.3.3 OAuth	14
2.3.4 Biometria	17
2.4 Poziomy zabezpieczeń	18
2.5 Techniczne porównanie działania mechanizmów autoryzacji	19
2.5.1 Autentykacja oparta na sekrecie	19
2.5.2 Autentykacja FIDO2	25
3 Specyfikacja FIDO2	31
3.1 Rejestracja	33
3.2 Autentykacja	37
4 Implementacja platformy badawczej	44
4.1 Architektura projektu	44
4.2 Ochrona przed spamem i wirusami	48
4.3 Tworzenie konta w ShildMail	49
4.4 Rejestracja urządzeń autoryzacyjnych	49

5 Analiza informacji uwierzytelniających w kanale komunikacyjnym.	56
5.1 Weryfikacja implementacji	58
5.2 Analiza warstwowa zabezpieczeń	61
6 Wnioski	66
6.0.1 Napotkane trudności	67
6.0.2 Możliwości rozwoju	68
6.0.3 Zdefiniowanie czynności do implementacji	69
Bibliografia	71
Spis rysunków	74
Spis tabel	76
Załączniki	77

Rozdział 1

Wstęp

Internet, od momentu powstania, zawsze był obszarem dynamicznego rozwoju i wymiany informacji, a zarazem nieustannie narażonym na nowe rodzaje zagrożeń. W początkowej fazie użytkownicy najczęściej stykali się z klasycznymi wirusami komputerowymi – programami zdolnymi do automatycznego kopowania się i infekowania kolejnych urządzeń, często w celu niszczenia danych lub destabilizowania systemu. Jednak wraz z rozwojem technologii ewoluowały w coraz bardziej wyrafinowane formy, takie jak robaki czy konie trojańskie, rozprzestrzeniające się za pomocą poczty elektronicznej, dyskietek, pendrive'ów i stron internetowych. W odpowiedzi na narastające zagrożenie, zaczęto opracowywać oprogramowanie antywirusowe, które miało na celu wykrywanie, izolowanie i usuwanie szkodliwych plików. Dzięki ciągłemu uaktualnianiu baz sygnatur wirusów antywirusy stały się bardzo skuteczne w ograniczaniu ryzyka infekcji, wykrywając nawet nowe i nieznane jeszcze warianty złośliwego oprogramowania. Ich popularność i rozwój sprawiły, że cyberprzestępcy zaczęli poszukiwać alternatywnych metod ataku.

Wkrótce stało się jasne, że najsłabszymogniwem w łańcuchu bezpieczeństwa jest człowiek. W obliczu rosnącej efektywności programów antywirusowych, coraz większe znaczenie zyskały techniki socjotechniczne, polegające na manipulowaniu użytkownikiem w celu wyłudzenia danych lub zachęcenia go do wykonania niebezpiecznych czynności. Cyberprzestępcy zauważyl, że wystarczy nakłonić użytkownika do wypełnienia formularza, wątpliwej autentyczności bądź kliknięcia w podejrzany link, by uzyskać dostęp do jego danych, niejednokrotnie z pominięciem nawet zaawansowanych mechanizmów bezpieczeństwa. W rezultacie główny kierunek działań atakujących przeniósł się na metody eksplotacji ludzkiej podatności na błędy i brak należytej uwagi, a tego rodzaju aktywność określa się mianem ataków phishingowych.

Zgodnie z danymi APWG (Anti-Phishing Working Group), w roku 2024 liczba zgłaszanych

ataków phishingowych utrzymywała się na stosunkowo stałym poziomie, w przedziale od 270 do 370 tysięcy incydentów miesięcznie. Zgodnie z najnowszymi analizami, ponad 60% wszystkich celów ataków phishingowych dotyczy trzech kluczowych obszarów: mediów społeczeństwowych, serwisów mailowych oraz instytucji finansowych. Jednocześnie zauważono, że banki oraz pokrewne instytucje finansowe stają się coraz bardziej celem dla cyberprzestępco, głównie z uwagi na rosnącą popularność dwuskładnikowego uwierzytelniania. Wprowadzane mechanizmy, takie jak kody wysypane na urządzenia mobilne, znacznie utrudniają skuteczne przeprowadzenie ataku i wyłudzenie wrażliwych danych. Zaawansowane systemy filtrowania poczty są coraz powszechniej stosowane, a wymagania dotyczące poprawnego wysyłania wiadomości elektronicznych stale rosną. W efekcie oszuści napotykają coraz większe trudności w dostarczaniu fałszywych e-maili do skrynek odbiorczych potencjalnych ofiar. Mechanizmy takie jak uwierzytelnianie nadawców, analiza reputacji adresów IP czy rozbudowane filtry antyspamowe sprawiają, że klasyczny phishing oparty na wysyłce masowych wiadomości staje się coraz mniej skuteczny [9].

1.1 Cel pracy

Celem niniejszej pracy jest implementacja wielowarstwowego uwierzytelnienia w postaci specyfikacji FIDO 2 oraz jego analiza działania w serwisie usługi poczty elektronicznej. Został zaimplementowany serwis udostępniający usługi poczty elektronicznej, do którego dostęp autoryzowany jest poprzez wielowarstwowy mechanizm uwierzytelniania. Szczegółowej analizie została poddana odporność proponowanego rozwiązania na ataki typu phishing, które stanowią jedno z najpowszechniejszych zagrożeń w kontekście wyłudzeń uwierzytelnienia. Przeprowadzona została dokładna analiza przepływu informacji pomiędzy klientem a serwerem z uwzględnieniem wszystkich warstw modelu odniesienia łączonych systemów otwartych. Na przykładzie opracowanego rozwiązania zostały wyznaczone dodatkowe czynniki poświadczania użytkownika, a następnie porównane ze standardowymi metodami autoryzacji opartymi na sekrecie.

W wyniku przeprowadzonej analizy zostało wykazane, w jaki sposób proponowane rozwiązanie skutecznie chroni przed atakami phishingowymi, eliminując jednowarstwową zależność od haseł na rzecz zaawansowanych mechanizmów weryfikacji tożsamości wielowarstwowej struktury programowo sprzętowej.

Rozdział 2

Przegląd literatury

2.1 Analiza metod autentyfikacji

W świecie cyfrowym, komunikując się z systemami i usługami, użytkownicy często muszą potwierdzić swoją tożsamość, aby uzyskać dostęp do zasobów i chronionych informacji. Procesem w którym dokonujemy sprawdzenia czy podmiot jest tym za kogo się podaje nazywamy uwierzytelnianiem.

Proces można rozłożyć na trzy zasadnicze etapy:

- Identyfikacja - podmiot (np. użytkownik) jednoznacznie określa, kim jest, przekazując systemowi unikalną wartość, taką jak nazwa użytkownika, adres e-mail czy numer identyfikacyjny.
- Uwierzytelnienie (autentykacja, autentyfikacja) – polega na potwierdzeniu, że zadeklarowana tożsamość jest prawdziwa. W tym celu system weryfikuje dodatkowe dane (np. hasło, kod jednorazowy), co pozwala upewnić się, że osoba lub urządzenie faktycznie jest tym, za kogo się podaje.
- Autoryzacja – mechanizm, w ramach którego system rozstrzyga, do jakich operacji i zasobów dana tożsamość ma uprawnienia. Proces ten uwzględnia zarówno możliwość wykonywania konkretnych działań (np. odczyt, zapis, modyfikacja), jak i dostęp do określonych elementów systemu, takich jak pliki, bazy danych czy usługi.

2.2 Podział mechanizmów

Mechanizmy autoryzacji różnią się przede wszystkim tym, na czym opierają proces weryfikacji tożsamości użytkownika. W klasycznym podejściu wyróżnia się trzy główne grupy czynników weryfikacji:

- Coś, co wiesz - ten rodzaj weryfikacji opiera się na informacji, którą użytkownik zna i która jest unikalna dla niego. Typowe przykłady to:
 - hasła - najczęściej stosowany mechanizm uwierzytelniania, złożoność i długość, wpływa na ich poziom bezpieczeństwa,
 - PIN-y - krótkie kody numeryczne, używane np. w bankomatach czy smartfonach.

Jednym z głównych atutów metod uwierzytelniania opartych na czynniku „coś, co wiesz” jest ich łatwość wdrożenia oraz niski koszt eksploatacji. Rozwiązania takie jak hasła czy kody PIN nie wymagają bowiem zaawansowanej infrastruktury ani specjalistycznych urządzeń, co przekłada się na minimalne bariery wejścia dla użytkowników i administratorów systemu. Niemniej jednak, ze względu na ograniczoną odporność tego rodzaju mechanizmów na ataki czy wycieki informacji, nie stanowią one samodzielnie wystarczającej formy zabezpieczenia w kontekście bardziej wymagających zastosowań, gdzie priorytetem jest wysoki poziom ochrony danych.

- Coś, co masz - weryfikacja w tej kategorii opiera się na posiadaniu fizycznego przedmiotu lub urządzenia które służy do potwierdzenia tożsamości użytkownika. Przykłady obejmują:
 - Tokeny sprzętowe - fizyczne urządzenia, które odpowiadają za generowanie i przechowywanie unikalnych kluczy bądź kodów weryfikacyjnych.
 - Aplikacje mobilne - generują jednorazowe kody lub wysyłają powiadomienia push w celu zatwierdzenia logowania.
 - Kody SMS - jednorazowe hasła wysyłane na numer telefonu użytkownika.

Weryfikacja oparta na czynniku „coś, co masz” charakteryzuje się wyższym poziomem bezpieczeństwa w porównaniu do metod opartych na wiedzy. Główną zaletą tego podejścia jest konieczność posiadania przez użytkownika fizycznego lub wirtualnego elementu uwierzytelniającego, takiego jak token sprzętowy, aplikacja mobilna generująca jednorazowe kody. Dzięki temu skutecznie utrudnia się dostęp osobom trzecim, które nie dysponują wymaganym urządzeniem.

- Coś, czym jesteś - opierają się na unikalnych cechach użytkownika, które nie mogą być łatwo podrobione Do tej grupy zalicza się:
 - odciski palców - wykorzystywane w smartfonach, laptopach i systemach dostępu fizycznego,
 - rozpoznanie twarzy - technologie używane m.in. w telefonach komórkowych oraz systemach monitoringu,
 - skanowanie tęczówki oka - stosowane w systemach wymagających bardzo wysokiego poziomu bezpieczeństwa.

Weryfikacja oparta na czynniku „coś, czym jesteś” wykorzystuje unikalne cechy fizyczne lub behawioralne użytkownika do potwierdzenia jego tożsamości. Zaletą tego podejścia jest wysoki poziom bezpieczeństwa, wynikający z trudności w podrobieniu tych cech, które są unikalne dla każdego użytkownika. Dzięki swojej naturze weryfikacja biometryczna eliminuje potrzebę zapamiętywania haseł lub noszenia fizycznych urządzeń uwierzytelniających. To sprawia, że jest nie tylko bezpieczna, ale także wygodna w użyciu.

2.3 Metody autentyfikacji

W celu potwierdzenia tożsamości użytkownika w systemy informatycznych, przez lata opracowano szereg różnych metod uwierzytelniania.

2.3.1 Hasła

Jednym z najpowszechniej stosowanych sposobów weryfikacji tożsamości użytkownika w systemach informatycznych. Mechanizm polega na wprowadzeniu dwóch podstawowych danych – loginu (często w postaci nazwy użytkownika lub adresu e-mail) oraz hasła. Wprowadzenie nazwy użytkownika pozwala jednoznacznie wskazać konto, które dana osoba reprezentuje w systemie (czynnik identyfikujący). Natomiast podanie właściwego hasła potwierdza, że ta osoba jest uprawnionym posiadaczem tego konta i ma prawo korzystać z przypisanych mu zasobów i usług.

Po pomyślnej weryfikacji serwer generuje unikalny identyfikator sesji i zapisuje go w bazie danych razem z odpowiednimi danymi użytkownika (np. ID użytkownika, czas trwania sesji, czy bieżące dane sesyjne). Następnie identyfikator ten jest wysyłany do przeglądarki w postaci ciasteczka. Ciasteczko może być podpisane lub zaszyfrowane za pomocą sekretu serwera, co zapewnia, że nie można go zmodyfikować ani podrobić. Użycie ciasteczek o flagach takich jak

HttpOnly oraz *Secure* znacząco zwiększa bezpieczeństwo systemu. Flaga *HttpOnly* zapobiega dostępowi do ciasteczek z poziomu skryptów uruchomionych na stronie, natomiast *Secure* - wymaga zabezpieczonego kanału komunikacyjnego.

Przy każdym kolejnym żądaniu przeglądarka załącza ciasteczko z identyfikatorem sesji. Serwer weryfikuje to ciasteczko, sprawdzając jego ważność i autentyczność, a następnie odczytuje dane sesji związane z identyfikatorem. Dzięki temu aplikacja wie, kto wykonuje daną akcję. Mechanizm ten pozwala serwerowi na centralne zarządzanie sesjami, w tym ich unieważnianie (np. podczas wylogowania) lub przedłużenie (np. przy każdym zapytaniu).

Ważnym aspektem bezpieczeństwa jest ochrona kanału komunikacyjnego, za pośrednictwem którego hasło jest przekazywane. Jeżeli uwierzytelnianie odbywa się poprzez nieszyfrowane protokoły, istnieje duże ryzyko przechwycenia danych logowania, ponieważ komunikacja może zostać podsłuchana, a treść pakietów może zostać odczytana przez osoby trzecie. Dlatego też w większości współczesnych rozwiązań stosuje się szyfrowane połączenia, takie jak HTTPS, co znacząco utrudnia potencjalne ataki. Pomimo prostoty i uniwersalności, metoda oparta wyłącznie na haśle niesie ze sobą pewne ograniczenia – w szczególności wymaga bezpiecznego zarządzania hasłami po stronie serwera, jak również konieczność zabezpieczenia kanału transmisji danych.

Hasła należą do kategorii „coś, co wiesz” i podlegają wyłącznej wymianie między użytkownikiem a systemem autoryzacyjnym. Ich skuteczność w ochronie danych wynika z założenia, że tylko uprawniona osoba zna hasło i może je przekazać w trakcie procesu logowania. Aby jednak zapewnić możliwie najwyższy poziom bezpieczeństwa, hasło powinno spełniać pewne kryteria opisane w punkcie 2.5.1.

W praktyce użytkownicy są w stanie zapamiętać zwykle około ośmiu różnych haseł [23]. Konieczność tworzenia skomplikowanych i rozbudowanych kombinacji sprawia, że często szuka się uproszczeń, poprzez stosowanie słów lub powtarzania tego samego hasła w różnych serwisach, co otwiera pole do nadużyć. Łatwe hasła mogą zostać odgadnięte na przykład za pomocą ataków typu *brute force* (systematyczne sprawdzanie wszystkich możliwych kombinacji znaków) lub ataków *słownikowych* (przeszukiwanie list popularnych słów, z uwzględnieniem najczęstszych modyfikacji, takich jak dodanie cyfr czy znaków specjalnych). Co więcej, czynnik ludzki bywa słabym ogniwem w łańcuchu bezpieczeństwa. Użytkownicy mogą paść ofiarą phishingu, czyli techniki w której cyberprzestępcy próbują wyłudzić poufne dane użytkownika. Oszuści najczęściej tworzą fałszywe strony internetowe, które do złudzenia przypominają oryginalne witryny znanych instytucji, takich jak banki, platformy e-commerce czy serwisy e-mailowe. Użytkownik, odwiedzając taką stronę, może nieświadomie wprowadzić swoje dane, przekazując je wprost w ręce oszustów.

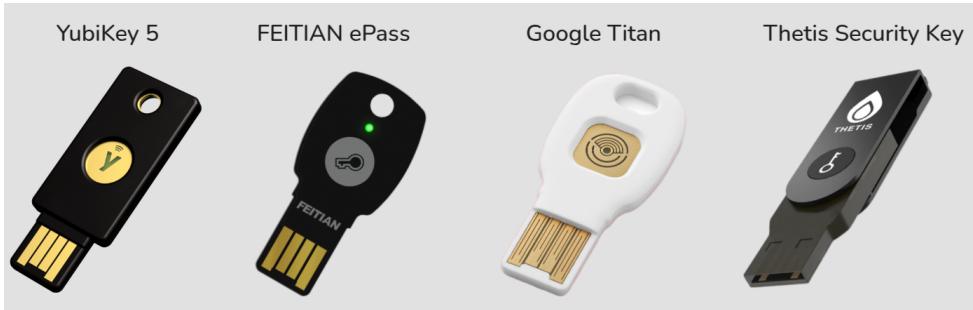
Innym zagrożeniem są ataki *social engineering*, podczas których napastnik manipuluje ofiarą, często udając administratora lub zaufaną osobę, aby nakłonić ją do dobrowolnego przekazania hasła. W efekcie nawet najbardziej złożone hasło może zostać ujawnione, jeśli jego właściciel nie zachowa odpowiedniej ostrożności.

Istnieje także problem bezpiecznego przechowywania haseł. Jeśli baza danych zawierająca hasła które są zapisane w postaci tekstu jawnego i zostanie skompromitowana, atakujący uzyskają natychmiastowy dostęp do wszystkich kont użytkowników bez konieczności dodatkowych działań. Dlatego przechowywanie haseł w formie zaszyfrowanej jest absolutnie niezbędne. Aby zwiększyć bezpieczeństwo przechowywania haseł, stosuje się techniki takie jak *haszowanie* i *solenie*. **Haszowanie** to proces, w którym dowolnie długie hasło jest zamieniane na ciąg znaków o stałej długości (skrót), z którego nie da się łatwo odzyskać pierwotnego hasła. Dodanie soli – losowej unikalnej wartości do hasła przed jego haszowaniem, znaczaco utrudnia ataki typu *rainbow table*, gdzie atakujący wykorzystują wcześniej przygotowane tablice haszy dla popularnych haseł. Kolejnym problemem jest zarządzanie i aktualizacja funkcji mieszających. W miarę rozwoju technologii, starsze funkcje, takie jak MD5 czy SHA-1, stają się podatne na **kolizje** (czyli sytuacje, w których różne dane wejściowe prowadzą do wygenerowania tego samego skrótu) oraz inne ataki kryptograficzne, co wymusza migrację do bardziej bezpiecznych algorytmów.

Hasła są najczęściej stosowaną formą uwierzytelniania w różnych obszarach cyfrowych. Używa się ich w serwisach mediów społecznościowych, gdzie użytkownicy logują się do platform takich jak Facebook, Instagram czy X. Chronią dostęp do gier online, umożliwiając graczom logowanie się do swoich kont na platformach takich jak Steam, Origin. A sklepach internetowych, hasła zabezpieczają konta klientów, chroniąc ich informacje o zamówieniach oraz dane osobowe. Pomimo szerokiego zastosowania, hasła często stają się słabym ogniwem w systemach bezpieczeństwa ze względu na możliwość użycia prostych, łatwych do odgadnięcia kombinacji, a także podatność na ataki typu brute force czy phishing.

2.3.2 Klucze bezpieczeństwa

Uwierzytelnianie za pomocą klucza bezpieczeństwa to metoda weryfikacji tożsamości użytkownika, która opiera się na fizycznym urządzeniu zwanym kluczem bezpieczeństwa. Klucz bezpieczeństwa może działać jako dodatkowy lub główny składnik uwierzytelnienia, w zależności od tego, jak został zaprojektowany system uwierzytelniania.



Rysunek 2.1: Klucze bezpieczeństwa

Swoje działanie opiera na **kryptografii asymetrycznej**, co oznacza, że podczas procesu rejestracji generowana jest unikalna para kluczy: prywatny i publiczny. Klucz prywatny, który pozostaje ściśle tajny, jest przechowywany wyłącznie w urządzeniu klucza bezpieczeństwa i nigdy nie opuszcza jego pamięci, co czyni go szczególnie odpornym na próby przechwycenia, służy do generowania **podpisu cyfrowego**, czyli unikalnego ciągu znaków potwierdzającego autentyczność i integralność danych. Natomiast klucz publiczny jest przesyłany do serwera i umożliwia weryfikację tego podpisu, potwierdzając, że operacja została wykonana przez posiadacza odpowiedniego klucza prywatnego.

Warto podkreślić, że przedstawiony mechanizm różni się od tradycyjnego podejścia, w którym klucz publiczny używany jest do szyfrowania danych, a klucz prywatny do ich odszyfrowania. W omawianym systemie, opartym na podpisie cyfrowym, operacja podpisywania odbywa się za pomocą klucza prywatnego, a proces weryfikacji – przy użyciu klucza publicznego. Takie rozwiązanie nie tylko umożliwia potwierdzenie autentyczności przesyłanych informacji, ale również zapewnia integralność danych, stanowiąc skuteczną metodę zabezpieczania komunikacji w systemach informatycznych.

Proces uwierzytelniania rozpoczyna się od wygenerowania przez serwer unikalnego wyzwania, które jest przesyłane do klucza bezpieczeństwa za pośrednictwem przeglądarki lub innego klienta. Klucz prywatny w urządzeniu podpisuje to wyzwanie, tworząc kryptograficznie zabezpiezioną odpowiedź. Odpowiedź ta, wraz z identyfikatorem klucza publicznego, jest przesyłana z powrotem na serwer. Dzięki podpisowi wygenerowanemu przez klucz prywatny, serwer może jednoznacznie zweryfikować tożsamość użytkownika, używając klucza publicznego zapisanego podczas rejestracji.

Klucze bezpieczeństwa oferują silną autoryzację, eliminując konieczność zapamiętywania skomplikowanych haseł. W procesie uwierzytelniania, po wprowadzeniu danych logowania, użytkownik jest proszony o potwierdzenie swojej tożsamości przy użyciu klucza bezpieczeństwa, który można podłączyć do portu USB, zbliżyć do urządzenia obsługującego NFC lub połączyć

przez Bluetooth. W niektórych przypadkach użytkownik może być poproszony o dotknienie klucza lub wprowadzenie kodu PIN w celu aktywacji. Kompatybilność z wieloma platformami i usługami internetowymi, które wspierają standardy FIDO2 czy Webauthn pozwala na korzystanie z jednego klucza na różnych kontach czy serwisach, co znacząco upraszcza zarządzanie procesami autoryzacji [23].

Mechanizm autentykacji oparty na kluczu bezpieczeństwa ma również swoje ograniczenia. Wysoki koszt zakupu klucza może stanowić barierę dla użytkowników indywidualnych oraz małych firm. Utrata jedynego klucza prowadzi do procesu odzyskiwania konta, co w zależności od wewnętrznych struktur i procedur firmowych, w przypadkach może być czasochłonne. Dlatego zaleca się posiadanie **przynajmniej dwóch** takich urządzeń, aby zapewnić ciągłość dostępu w przypadku zgubienia lub uszkodzenia jednego z kluczy.

Klucze bezpieczeństwa zaliczają się do kategorii „coś, co masz”, co oznacza, że każdorazowa autoryzacja wymaga fizycznego przedstawienia urządzenia - noszenie takiego urządzenia, zwłaszcza podczas podróży czy w sytuacjach wymagających mobilności, może stanowić pewien dyskomfort i wymagać od użytkownika dodatkowej uwagi. Obecnie nowoczesne rozwiązania w zakresie autoryzacji są wdrażane przede wszystkim przez największe platformy i usługi internetowe, które dysponują wystarczającymi zasobami oraz skalą działania umożliwiającą szybkie przyjęcie innowacyjnych technologii.

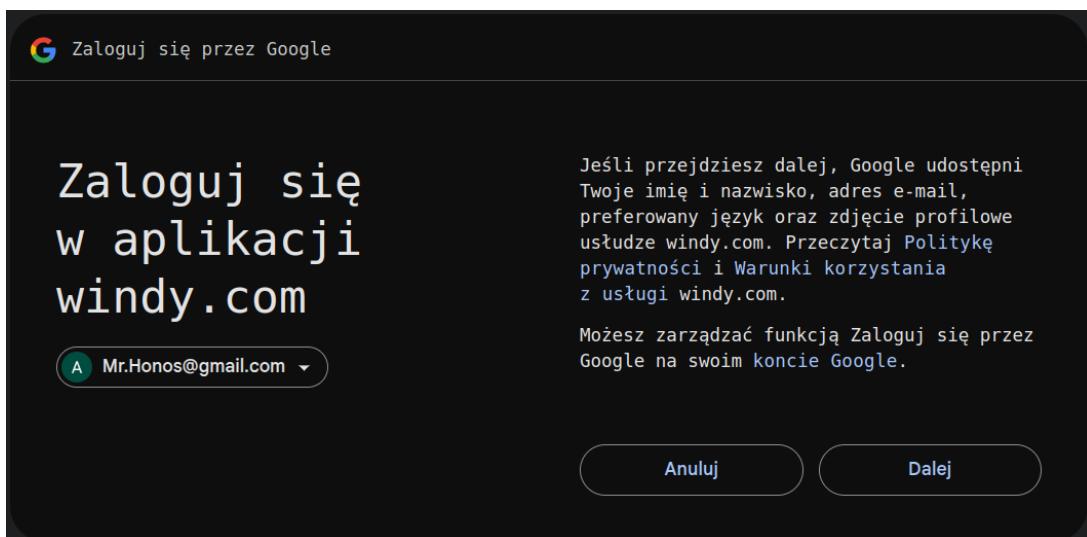
Standard FIDO2 znalazł szerokie zastosowanie w sektorach, gdzie wymagana jest wysoka ochrona danych i bezpieczeństwo, przykładem są serwisy rządowe - Login.gov (USA), mObywatel (gov.pl), RealMe (New Zealand Government) czy usługi finansowe - ING, MasterCard, Binance. Ponadto, jest powszechnie stosowaną metodą logowania do systemów operacyjnych - Windows, Linux oraz macOS. Dodatkowo, klucze bezpieczeństwa zgodne z FIDO2 mogą być wykorzystywane do logowania do zdalnych maszyn poprzez protokół SSH, co znacznie zwiększa bezpieczeństwo dostępu do zdalnych systemów informatycznych. Zaobserwowano wdrażanie tego rozwiązania głównie w sektorach, w których wymogi dotyczące ochrony danych i kontroli dostępu są najwyższe, takich jak finanse czy administracja publiczna. W tych obszarach, gdzie bezpieczeństwo informacji stanowi priorytet ze względu na potencjalnie katastrofalne skutki włamań, technologia ta pełni kluczową rolę w budowaniu nowoczesnych systemów uwierzytelniania, skutecznie odpowiadając na rosnące wyzwania związane z cyberbezpieczeństwem w dynamicznym środowisku cyfrowym.

2.3.3 OAuth

OAuth (ang. Open Authorization) to standard umożliwiający delegowanie uprawnień dostępu do zasobów użytkownika innym aplikacjom, bez konieczności ujawniania jego poufnych

danych uwierzytelniających, takich jak hasła. W tej metodzie użytkownik może łatwo autoryzować dostęp aplikacji zewnętrznych do swoich danych, poprzez potwierdzenie żądania dostępu w oknie przeglądarki. Proces OAuth polega na wydaniu przez serwer autoryzacyjny specjalnego tokena dostępu, który aplikacja zewnętrzna może wykorzystać do uzyskania dostępu do chronionych zasobów użytkownika. Token ten jest skonfigurowany z precyzyjnymi uprawnieniami i posiada ograniczony czas ważności, co zapewnia, że aplikacja zewnętrzna może uzyskać dostęp tylko do tych danych, które są niezbędne do wykonania określonego zadania, i to jedynie przez określony czas. OAuth jest szeroko stosowany w integracji różnych usług internetowych, umożliwiając użytkownikom logowanie się do aplikacji za pomocą swoich kont Google, Facebook czy X.

Na Rysunku 2.1 przedstawiono przykładowe okienko logowania opartego na protokole OAuth. Interfejs ten jasno komunikuje, jakie dane osobowe zostaną udostępnione aplikacji windy.com, jeżeli użytkownik zdecyduje się na autoryzację za pośrednictwem usługi Google. Użytkownik jest informowany o zamiarze udostępnienia imienia i nazwiska, adresu e-mail, preferowanego języka oraz zdjęcia profilowego. Decyzja o przekazaniu tych danych należy wyłącznie do użytkownika, co umożliwia mu świadome i przejrzyste udostępnianie informacji osobowych.



Rysunek 2.2: Okno logowania do serwisu windy.com przy pomocy konta Google

OAuth oferuje liczne korzyści, które przyczyniają się do jego szerokiego zastosowania w systemach informatycznych. Jednym z najważniejszych atutów OAuth jest zwiększone bezpieczeństwo, ponieważ eliminuje konieczność udostępniania haseł użytkowników aplikacjom trzecim. Zamiast tego wykorzystuje tokeny dostępu, które są ograniczone czasowo i specyficzne dla danej aplikacji oraz zakresu dostępu. Dzięki temu, nawet jeśli token zostanie przechwycony, je-

go użyteczność jest ograniczona, co zmniejsza ryzyko nieautoryzowanego dostępu do zasobów użytkownika [23].

OAuth znacznie ułatwia integrację między usługami, umożliwiając użytkownikom logowanie się do wielu aplikacji za pomocą jednego konta, na przykład konta Google, Facebook czy X. To nie tylko upraszcza proces logowania dla użytkowników, ale także redukuje liczbę kont, które muszą być zarządzane, co przekłada się na lepszą organizację i mniej problemów związanych z zarządzaniem tożsamością po stronie serwera. Ponadto, ten rodzaj autoryzacji może być stosowany w różnych typach aplikacji – zarówno webowych, mobilnych, jak i desktopowych – oraz w różnorodnych scenariuszach użycia. A w rezultacie, przyczynia się do poprawy doświadczenia użytkownika, oferując szybsze i wygodniejsze logowanie się do nowych aplikacji bez potrzeby tworzenia kolejnych kont i pamiętania wielu różnych haseł.

Mimo że OAuth jest powszechnie stosowanym, wiąże się z kilkoma istotnymi wadami, które warto uwzględnić podczas jego wdrażania. Protokół ten wymaga dokładnego zrozumienia jego mechanizmów oraz precyzyjnej konfiguracji, co może stanowić wyzwanie dla deweloperów, zwłaszcza tych mniej zaznajomionych z zaawansowanymi aspektami bezpieczeństwa. Określanie i zarządzanie zakresami dostępu może być trudne, zwłaszcza w dużych systemach z wieloma aplikacjami i różnorodnymi wymaganiami dotyczącymi dostępu. Niewłaściwe zarządzanie uprawnieniami może prowadzić do nadmiernego przyznawania dostępu lub zbyt restrykcyjnych ograniczeń, co może wpływać na funkcjonalność aplikacji oraz bezpieczeństwo danych. Konieczność precyzyjnego definiowania zakresów wymaga ciągłego monitorowania i aktualizacji polityk dostępu.

Ponadto, wprowadzana jest zależność od serwera autoryzacji – awaria tego serwera może uniemożliwić użytkownikom logowanie się do aplikacji, co negatywnie wpływa na dostępność usług czy reputację organizacji.

OAuth jest stosowany w sytuacjach, gdy do funkcjonowania aplikacji nie jest potrzebny pełny dostęp do danych użytkownika. Zamiast tego, umożliwia aplikacjom uzyskanie jedynie wybranych uprawnień do korzystania z określonych informacji przechowywanych na serwerze autoryzacyjnym. Zdarza się, że po udzieleniu uprawnień, aplikacja wyświetla formularz, w którym użytkownik jest proszony o podanie dodatkowych informacji, takich jak preferencje dotyczące mody, zainteresowania czy ulubiony kolor. Przykładem zastosowania OAuth są serwisy pogodowe takie jak Windy, Ventusky oraz fora internetowe np. Reddit, Blogger, gdzie podstawowe dane są wystarczające do funkcjonowania aplikacji, a dodatkowe informacje mogą być zbierane w sposób dobrowolny [4].

2.3.4 Biometria

Biometria jest jednym z najgorętszych trendów w dziedzinie bezpieczeństwa, szczególnie w kontekście rosnącej liczby urządzeń mobilnych wyposażonych w kamery czy czytniki linii papilarnych. W odróżnieniu od tradycyjnych metod uwierzytelniania opartych na tym, co użytkownik zna („coś, co wiesz”), takich jak hasła, czy na tym, co posiada („coś, co masz”), biometria wykorzystuje naturalne cechy człowieka. Opiera się na unikalnych cechach fizycznych lub behawioralnych użytkownika, takich jak odciski palców, podpis, rozpoznawanie twarzy czy skanowanie tęczówki oka, co sprawia, że proces weryfikacji jest nie tylko efektywny, ale także znacznie trudniejszy do podrobienia przez osoby trzecie [2].



Rysunek 2.3: Proces skanowania odcisków palca [10]

Mechanizm autentykacji biometrycznej działa w kilku etapach. Pierwszym krokiem jest rejestracja, podczas której system zbiera wzorzec biometryczny użytkownika. W przypadku skanowania odcisku palca, na przykład, system rejestruje unikalny układ grzbietów papilarów. Następnie, podczas próby logowania, użytkownik przedstawia swoją cechę biometryczną, która jest skanowana i system dokonuje porównania z wcześniej zapisanym wzorcem, aby zweryfikować, czy osoba próbująca uzyskać dostęp jest rzeczywiście uprawnionym użytkownikiem. Jeśli zebrane dane biometryczne odpowiadają zapamiętanemu wzorcowi, system potwierdza tożsamość użytkownika, umożliwiając mu dostęp do chronionych zasobów.

Biometria rozwiązuje problem dzielenia się kontami, ponieważ autoryzacja jest bezpośrednio związana z unikalnymi cechami użytkownika, a nie z danymi - które można łatwo przekazać innym. Dzięki temu, tylko autentyczny właściciel cechy biometrycznej ma możliwość dostępu do konta.

Jednakże, autentykacja biometryczna napotyka również na pewne problemy i ograniczenia. Jednym z głównych wyzwań jest dostępność tej technologii dla osób z niepełnosprawnościami lub dysfunkcjami np. brak dloni. Dla tych użytkowników, tradycyjne metody biometryczne mogą być niedostępne, co ogranicza ich zastosowanie. Ponadto, skuteczność rozpoznawania danych biometrycznych może być problematyczna w sytuacjach prób fałszowania, takich jak użycie wydrukowanego zdjęcia twarzy. Istotne jest minimalizowanie wskaźników fałszywych akceptacji oraz fałszywych odrzuceń. Fałszywe akceptacje prowadzą do nieautoryzowanego dostępu, podczas gdy fałszywe odrzucenia (tj. odrzucenie uprawnionego podmiotu) powodują frustrację i niepokój.

Implementacja biometrii wymaga zastosowania specjalistycznego sprzętu, takiego jak skanerów linii papilarnych, czytników tęczówki, kamery, co wiąże się z dodatkowymi kosztami oraz koniecznością integracji z istniejącą infrastrukturą. Oznacza to konieczność inwestycji w dedykowane urządzenia, które muszą być zakupione, skonfigurowane i zintegrowane z aktualnymi systemami bezpieczeństwa. Takie inwestycje obejmują nie tylko zakup sprzętu, ale także koszty związane z jego instalacją oraz konfiguracją, co często wymaga wsparcia specjalistów.

Zagrożenie wyciekiem danych biometrycznych jest jednym z najpoważniejszych problemów związanych z autentyką biometryczną. Dane biometryczne są niezwykle wrażliwe, a ich kradzież lub nieautoryzowane wykorzystanie może prowadzić do poważnych naruszeń prywatności oraz bezpieczeństwa użytkowników. W przeciwnieństwie do haseł, dane biometryczne nie mogą być zmienione, co oznacza, że ich utrata ma długotrwałe konsekwencje. Stosowanie skonsolidowanych systemów autentykacyjnych nie jest z tego powodu korzystne, wystarczy skompromitować taki system, aby przechwycić wszystkie dane biometryczne użytkowników. Centralizacja tworzy pojedynczy punkt awarii, który może mieć katastrofalne skutki dla prywatności i bezpieczeństwa danych.

Biometryczne systemy autentykacji, dzięki swojej wysokiej skuteczności, są stosowane tam, gdzie wymagany jest wysoki stopień bezpieczeństwa. Wśród takich miejsc znajdują się FBI CIA, NASA, banki.

2.4 Poziomy zabezpieczeń

W systemach bezpieczeństwa wyróżnia się różne poziomy zabezpieczeń, które pomagają chronić dostęp do danych i usług. Każdy kolejny poziom wprowadza dodatkowe procedury, zwiększając tym samym poziom trudności w przełamaniu zabezpieczeń.

- jednopoziomowa - najbardziej podstawowy mechanizm uwierzytelniania, który zazwyczaj opiera się wyłącznie na sekrecie,

- dwuskładnikowe uwierzytelnianie (2FA) - mechanizm, który wymaga potwierdzenia tożsamości za pomocą dwóch niezależnych czynników,
- uwierzytelnianie wieloskładnikowe (MFA) - rozwija koncepcję 2FA, wprowadzając więcej niż dwa czynniki weryfikacji.

W obliczu postępu technologicznego i rosnącego ryzyka cyberzagrożeń, systemy uwierzytelniania oparte na wielu czynnikach znajdują coraz częstsze zastosowanie. Takie rozwiązania zwiększą ochronę danych przechowywanych na platformach, jednocześnie budując zaufanie użytkowników. W typowych implementacjach wykorzystuje się różnorodne metody weryfikacji, co skutkuje tym, że potencjalny atakujący musi pokonać kilka oddzielnych warstw zabezpieczeń, co znaczco utrudnia przeprowadzenie udanego ataku [7].

2.5 Techniczne porównanie działania mechanizmów autoryzacji

2.5.1 Autentykacja oparta na sekrecie

Należy do najczęściej stosowanych metod weryfikacji tożsamości użytkowników w systemach informatycznych. Mechanizm ten polega na przypisaniu danej osobie unikalnych informacji uwierzytelniających (tzw. sekretu), zazwyczaj w postaci hasła lub innego ciągu znaków. Rozwiązanie to, mimo swej powszechności, wiąże się jednak z szeregiem potencjalnych zagrożeń wynikających z możliwości przechwycenia, kradzieży bądź nieuprawnionego ujawnienia sekretów. Bezpieczeństwo przekazywanego sekretu w dużej mierze zależy od kanału komunikacyjnego, w którym następuje wymiana danych między klientem a serwerem. W przypadku korzystania z protokołów neszifrowanych, takich jak HTTP, ryzyko przechwycenia hasła jest wysokie, ponieważ możliwe jest odczytanie zawartości pakietów w sposób bezpośredni.

Hasła i ich zagrożenia

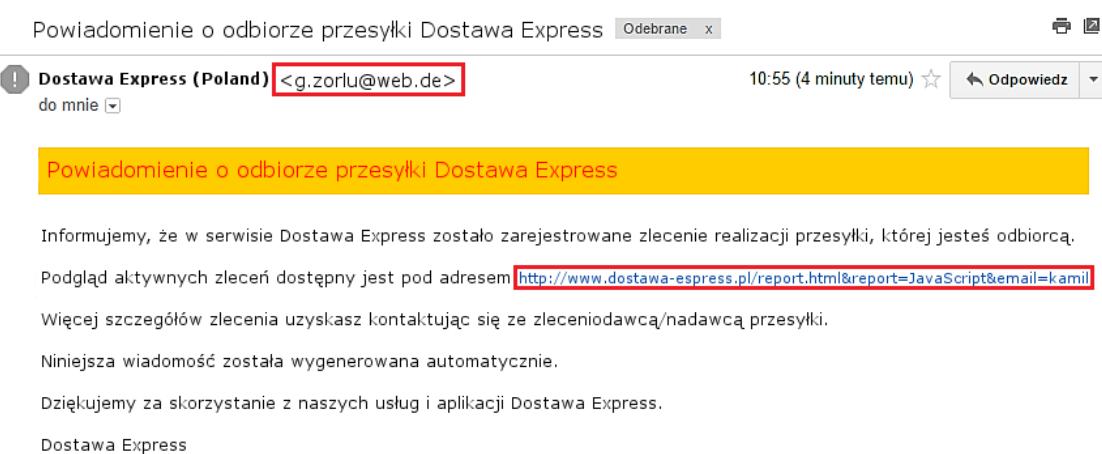
Pomimo, że hasła stanowią powszechnie stosowany środek zabezpieczający, ich używanie wiąże się z ryzykiem wystąpienia różnorodnych ataków [19] [11]:

- Atak siłowy (brute force) - polega na systematycznym sprawdzaniu wszystkich możliwych kombinacji znaków w celu odgadnięcia hasła. W praktyce atakujący korzysta z oprogramowania, które automatycznie generuje i testuje kolejne warianty haseł. Kluczową rolę odgrywa mechanizm haszowania. Systemy komputerowe często nie przechowują haseł

w postaci jawnnej, lecz jako ich skróty uzyskane przy użyciu funkcji haszujących, takich jak MD5, SHA-1 czy SHA-256. W przypadku ataku brute force, atakujący generuje różne kombinacje potencjalnych haseł, a następnie oblicza dla nich hasze, porównując uzyskane wyniki z haszem przechowywanym w systemie. Jeśli wygenerowany skrót odpowiada zapisanej wartości, oznacza to, że odgadnięto właściwe hasło. Prędkość działania funkcji haszujących ma kluczowe znaczenie dla skuteczności ataku brute force. Funkcje, które operują szybko, umożliwiają atakującemu przetestowanie bardzo dużej liczby kombinacji w krótkim czasie, co z kolei zwiększa ryzyko skutecznego przełamania zabezpieczeń. W odpowiedzi na zagrożenia związane z atakami brute-force, opracowano funkcje haszujące o zwiększonej złożoności obliczeniowej, które celowo wydłużają czas generowania skrótów. Takie podejście znaczco obniża efektywność masowych prób ataku, utrudniając potencjalnym intruzom przełamanie zabezpieczeń nawet przy wykorzystaniu znaczących zasobów obliczeniowych.

- Atak słownikowy (dictionary attack) - to rodzaj siłowego, w którym cyberprzestępca wykorzystuje gotowe zbiory najczęściej używanych fraz, słów i haseł (tzw. słowniki), aby systematycznie próbować odgadnąć właściwe dane uwierzytelniające. W przeciwieństwie do brute force, atak słownikowy ogranicza się do słów i ciągów znaków, które statystycznie występują najczęściej w hasłach wybieranych przez użytkowników. Najważniejszym czynnikiem wpływającym na skuteczność ataku słownikowego jest odpowiedni dobór słownika, gdyż to on zawiera potencjalne hasło. Właściwie skonstruowany zbiór słów i fraz, uwzględniający powszechnie używane kombinacje oraz warianty, podnosi szansę na trafienie właściwego hasła, stanowiąc tym samym kluczowy element sukcesu całego ataku.
- Sniffing - polega na pasywnym przechwytywaniu danych przesyłanych w sieciach komputerowych, której celem jest uzyskanie dostępu do informacji o charakterze poufnym, takich jak dane uwierzytelniające. Mechanizm ten opiera się na rejestracji ruchu sieciowego bez ingerencji w zawartość przesyłanych danych. Do tego celu wykorzystuje się narzędzia, takie jak Wireshark czy tcpdump, które umożliwiają jedynie obserwację i szczegółową analizę pakietów.
- Man in the middle - polega na aktywnym przechwytywaniu oraz modyfikowaniu komunikacji pomiędzy dwiema stronami (klientem i serwerem). W odróżnieniu od sniffingu, atak man in the middle charakteryzuje się aktywną ingerencją w przesyłane informacje co stanowi poważne zagrożenie dla poufności i integralności komunikacji.

- Phishing - forma oszustwa, w której użytkownik zostaje podstępnie nakłoniony do podania danych uwierzytelniających na spoprawowanych stronach internetowych. Typowy scenariusz ataku rozpoczyna się od otrzymania wiadomości e-mail rzekomo pochodzącej od zaufanego nadawcy, na przykład banku, firmy kurierskiej lub serwisu społecznościowego. W treści wiadomości znajduje się pilny komunikat wzbudzający niepokój — może to być ostrzeżenie o wygaśnięciu konta, podejrzany logowaniu z nieznanej adresu IP czy nieodebranej przesyłce, która za chwilę zostanie zwrocona do nadawcy. Ponadto, w wiadomości zamieszczony jest odnośnik prowadzący do strony będącej podobnej do oryginalnej witryny, co dodatkowo potęguje wrażenie autentyczności. Cyberprzestępca liczy na to, że użytkownik, działając w warunkach stresu, straci czujność i ujawni swoje dane logowania. Informacje te trafiają w ręce oszustów, którzy mogą je następnie wykorzystać do przejęcia konta, kradzieży środków finansowych lub innych nadużyć w przestrzeni internetowej.



Rysunek 2.4: Przykład phisingu – fałszywe powiadomienie o przesyłce [15]

- Ataki socjotechniczne - metoda wykorzystująca manipulację psychologiczną, której celem jest nakłonienie ofiary do ujawnienia poufnych informacji lub wykonania określonych działań, takich jak instalacja złośliwego oprogramowania. Zamiast wykorzystywać wyłącznie techniczne luki w zabezpieczeniach, atakujący odwołują się do ludzkiej podatności na stres, niewiedzę czy autorytet. Przykładowo, ofiara może otrzymać telefon od osoby podszywającej się pod pracownika działu IT, która pod pretekstem pilnej potrzeby żąda podania hasła, lub atakujący może wykorzystać media społecznościowe do zdobycia informacji o pracownikach, co pozwala na późniejsze przeprowadzenie bardziej złożonych ataków.

Dobre praktyki w stosowaniu haseł

Choć hasła pozostają najpopularniejszą metodą uwierzytelniania, w praktyce wiążę się z nimi szereg trudności i zagrożeń. Według badań ponad 70% ludzi przyznaje się do wielokrotnego używania tego samego hasła na różnych platformach i usługach [5]. Takie działanie znaczaco podnosi ryzyko zagrożenia bezpieczeństwa, ponieważ w przypadku przejęcia jednego z kont użytkownika, istnieje duże prawdopodobieństwo, że pozostałe konta korzystające z tego samego hasła również będą zagrożone. Aby zminimalizować ryzyko, kluczowe jest stosowanie zasad tworzenia silnych haseł, które stanowią pierwszą linię obrony przed nieautoryzowanym dostępem [6] [14].

Uwzględniając dzisiejszy poziom mocy obliczeniowej, hasło o wysokim poziomie bezpieczeństwa powinno spełniać określone kryteria:

- powyżej 16 znaków,
- zawiera małe i wielkie litery, liczby oraz symbole,
- nie zawiera słów słownikowych,
- znaczaco różni się od poprzednio stosowanych haseł.

W ramach niniejszego opracowania przeprowadzono proces łamania skrótu SHA-512, korzystając z platformy testowej o określonej specyfikacji opisanej w punkcie 2.5.1. Celem eksperymentu była ocena skuteczności powszechnie stosowanej zasady tworzenia haseł, zakładającej minimalną długość ośmiu znaków. W tym kontekście uwzględniono różne warianty zestawu znaków, aby sprawdzić, w jakim stopniu rozbudowanie puli dostępnych symboli przekłada się na wzrost poziomu zabezpieczeń. Analiza wyników pozwoliła określić realną efektywność najpopularniejszych wytycznych przy tworzeniu haseł, a także zweryfikować, czy stosowane praktyki w wystarczający sposób chronią przed atakami typu brute-force.

Platforma testowa:

- system operacyjny: Ubuntu Linux 24.04 LTS,
- procesor: Ryzen 5 1600,
- karta graficzna: GeForce GTX 1060 6GB,
- pamięć podręczna: 32GB 3200 GHz,
- płyta główna: MSI B450 Tomahawk,
- oprogramowanie: hashcat v6.2.6.

Na podstawie przedstawionej specyfikacji platformy osiągnięto szybkość generowania skrótów SHA512 na poziomie około **485MH/s**. Uzyskany wynik stanowi istotny benchmark, który będzie wykorzystywany jako punkt odniesienia przy dalszych analizach.

```
Session.....: hashcat
Status.....: Quit
Hash.Mode....: 1700 (SHA2-512)
Hash.Target...: hashes.txt
Time.Started...: Sun Feb 2 22:14:18 2025 (3 mins, 42 secs)
Time.Estimated.: Fri Jul 11 09:06:02 2025 (158 days, 9 hours)
Kernel.Feature.: Pure Kernel
Guess.Mask....: ?a?a?a?a?a?a?a [8]
Guess.Queue....: 1/1 (100.00%)
Speed.#1.....: 484.7 MH/s (10.78ms) @ Accel:8 Loops:256 Thr:256 Vec:1
Recovered.....: 1/2 (50.00%) Digests (total), 0/2 (0.00%) Digests (new)
Progress.....: 106984775680/6634204312890625 (0.00%)
Rejected.....: 0/106984775680 (0.00%)
Restore.Point...: 122880/7737809375 (0.00%)
Restore.Sub.#1..: Salt:0 Amplifier:79616-79872 Iteration:0-256
Candidate.Engine.: Device Generator
Candidates.#1...: 0Q1Dwurd -> < (t:312
Hardware.Mon.#1..: Temp: 68c Fan: 49% Util:100% Core:1999MHz Mem:3802MHz Bus:16
Started: Sun Feb 2 22:14:15 2025
Stopped: Sun Feb 2 22:18:02 2025
```

Rysunek 2.5: Proces zgadywania haseł realizowany z wykorzystaniem narzędzia hashcat.

Ilość wszystkich możliwych haseł dla danego zbioru znaków można wyliczyć za pomocą wzoru:

$$N = L^C \quad (2.1)$$

gdzie:

N - ilość możliwych kombinacji,

L - długość hasła,

C - ilość znaków w zbiorze,

Czas niezbędny do przetestowania pełnego zbioru potencjalnych kombinacji wyznaczono, korzystając z poniższej formuły:

$$t = \frac{N}{S}$$

gdzie:

t - czas sprawdzenia wszystkich możliwych kombinacji,

N - ilość możliwych kombinacji,

S - szybkość generowania kolejnych haseł,

W przypadku prezentowanego zestawu komputerowego wartość $S = 4.85 \times 10^8$.

Następnie zdefiniowano zbiory:

$$A = \{a, b, c, \dots, x, y, z\}$$

$$B = \{A, B, C, \dots, X, Y, Z\}$$

$$C = \{0, 1, 2, \dots, 7, 8, 9\}$$

$$D = \{ , , \#, \$, \%, ., *, +, -, /, <, =, >, ?, @, [, \], ; -\}$$

Tabela 2.1: Wpływ wielkości zbioru znaków i długości hasła na czas łamania

Zbiór	Długość zbioru	Długość hasła	Ilość kombinacji	Czas sprawdzenia
A	25	8	3.8×10^{11}	$784[\text{s}] \sim 10[\text{min}]$
		9	3.8×10^{12}	$7865[\text{s}] \sim 131[\text{min}]$
		16	2.3×10^{22}	$4.7 \times 10^{13}[\text{s}] \sim 1.4 \times 10^9[\text{dni}]$
$A \cup B$	50	8	3.9×10^{13}	$8.0 \times 10^5[\text{s}] \sim 23.2 [\text{dni}]$
		9	2.0×10^{15}	$4.0 \times 10^6[\text{s}] \sim 46.6 [\text{dni}]$
		16	1.5×10^{27}	$3.0 \times 10^{18}[\text{s}] \sim 8.7 \times 10^{13}[\text{dni}]$
$A \cup B \cup C$	60	8	1.7×10^{14}	$3.5 \times 10^6[\text{s}] \sim 41 [\text{dni}]$
		9	1.0×10^{16}	$2.0 \times 10^7[\text{s}] \sim 240[\text{dni}]$
		16	2.8×10^{28}	$5.7 \times 10^{19}[\text{s}] \sim 1.7 \times 10^{15}[\text{dni}]$
$A \cup B \cup C \cup D$	95	8	6.6×10^{18}	$1.4 \times 10^7[\text{s}] \sim 157[\text{dni}]$
		9	6.3×10^{19}	$1.3 \times 10^8[\text{s}] \sim 15040[\text{dni}]$
		16	4.4×10^{31}	$9.0 \times 10^{23}[\text{s}] \sim 1.0 \times 10^{18}[\text{dni}]$

Wyniki symulacji pokazują, że przy wykorzystaniu pełnej przestrzeni kombinacji, złamanie każdego możliwego hasła 8-znakowego zajmuje około 157 dni. To pokazuje, że przy stosunkowo krótkiej długości hasła liczba możliwych kombinacji jest na tyle ograniczona, że nawet przy nowoczesnych metodach brute-force atakujący są w stanie stosunkowo szybko przeprowadzić skuteczny atak. Z tabeli 2.1 można wywnioskować, że długość hasła ma fundamentalne znaczenie dla jego bezpieczeństwa, ponieważ przestrzeń kombinacji rośnie wykładniczo wraz z każdym dodatkowym znakiem. Zwiększenie długości hasła o jeden znak przy stałym zbiorze znaków powoduje pomnożenie liczby możliwych kombinacji przez rozmiar tego zbioru.

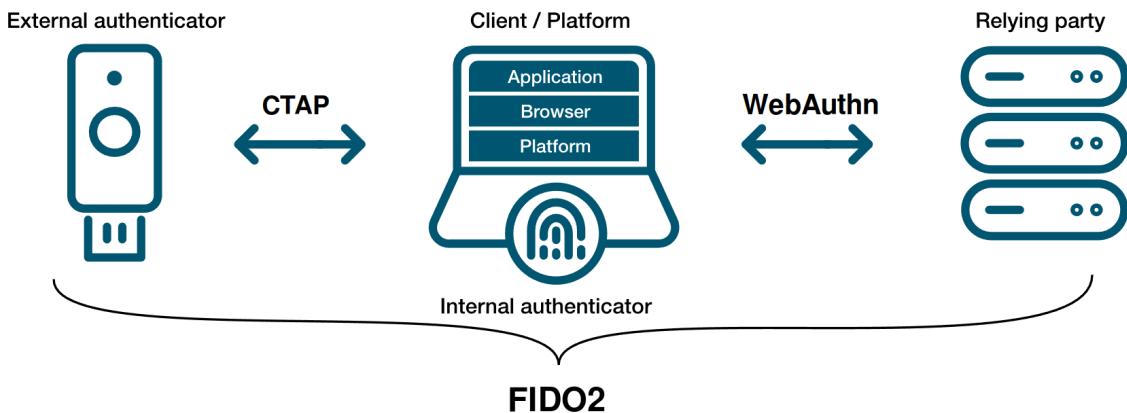
Kolejnym czynnikiem zabezpieczającym jest dobór odpowiedniej funkcji skrótu - te które działają bardzo szybko, umożliwiają atakującym sprawdzanie ogromnej liczby kombinacji w krótkim czasie, dlatego rekomenduje się stosowanie algorytmów o zwiększonej złożoności obliczeniowej, które celowo spowalniają proces haszowania. W procesie wyboru funkcji niezbędne jest uwzględnienie odporności na kolizje, czyli zdolności funkcji do generowania unikalnych skrótów dla różnych danych wejściowych, co stanowi kluczowy element ochrony przed atakami kryptograficznymi [1].

Należy pamiętać, że nawet najlepsze hasła nie gwarantują bezpieczeństwa, jeśli stanowią jedyny element zabezpieczenia. Dlatego nowoczesne systemy ochrony oparte są na wielopoziomowych strategiach, które utrudniają przełamanie zabezpieczeń i zmniejszają ryzyko naruszenia danych.

2.5.2 Autentykacja FIDO2

FIDO2 (Fast IDentity Online 2) to nowoczesny standard uwierzytelniania opracowany przez FIDO Alliance we współpracy z W3C (World Wide Web Consortium), który pozwala na bezpieczne logowanie bez użycia tradycyjnych haseł (passwordless). Opiera się na kryptografii asymetrycznej i oferuje użytkownikom bezpieczny, szybki i wygodny sposób uwierzytelniania w usługach internetowych, aplikacjach oraz na urządzeniach. Proces uwierzytelniania opiera się na współpracy dwóch głównych komponentów: WebAuthn i CTAP, oraz wykorzystaniu pary kluczy kryptograficznych (publicznego i prywatnego) [20].

- WebAuthn (Web Authentication) - to interfejs programistyczny zintegrowany z przeglądarkami internetowymi i platformami systemowymi, którego celem jest bezpieczne zarządzanie procesem uwierzytelniania użytkowników. Dzięki WebAuthn serwer może w bezpieczny sposób komunikować się z urządzeniem uwierzytelniającym (np. kluczem sprzętowym czy modułem biometrycznym). Umożliwia wykorzystanie kluczy kryptograficznych, co przekłada się na wyższy poziom ochrony w porównaniu do tradycyjnych metod opartych wyłącznie na sekrecie,
- CTAP (Client-to-Authenticator Protocol) - jest protokołem umożliwiającym komunikację między urządzeniem użytkownika (np. laptopem, smartfonem) a komponentem uwierzytelniającym (np. kluczem bezpieczeństwa, wbudowanym czytnikiem linii papilarnych). CTAP obsługuje różne typy połączeń, takie jak:
 - USB - fizyczne podłączenie klucza sprzętowego do urządzenia,
 - Bluetooth - bezprzewodowa komunikacja z autentykatorem,
 - NFC - komunikacja bezprzewodowa bliskiego zasięgu, używane głównie w urządzeniach mobilnych.



Rysunek 2.6: FIDO2 - Komponenty [24]

Urządzenia uwierzytelniające to specjalne jednostki kryptograficzne, które mogą być realizowane zarówno w formie rozwiązań sprzętowych (np. klucze bezpieczeństwa), jak i programowych. Ich głównym zadaniem jest rejestracja użytkowników w systemie oraz weryfikacja ich tożsamości podczas logowania. Działają one w oparciu o standard FIDO2, który eliminuje potrzebę używania tradycyjnych haseł, zastępując je bezpiecznymi mechanizmami kryptograficznymi.

Urządzenia autoryzujące można podzielić na dwie główne kategorie[18]:

- Autentykatory platformowe - stanowią środki uwierzytelniające osadzone bezpośrednio w urządzeniu użytkownika. Wykorzystują one zintegrowane mechanizmy zabezpieczeń, które przechowują klucz prywatny oraz realizują operacje kryptograficzne, umożliwiając proces uwierzytelniania bez konieczności stosowania zewnętrznych urządzeń. Dodatkowo, te rozwiązania często implementują biometryczne metody weryfikacji, takie jak skanowanie odcisku palca czy identyfikacja twarzy,
- Zewnętrzne urządzenia uwierzytelniające (en. roaming authenticators) - urządzenia, które użytkownik może swobodnie przenosić i wykorzystywać na różnych platformach oraz urządzeniach. Ich cechą jest to, że komunikują się z systemem poprzez interfejsy takie jak USB, NFC lub Bluetooth. Dzięki swojej mobilności umożliwiają uwierzytelnianie niezależnie od tego, czy dana maszyna posiada wbudowane mechanizmy zabezpieczające, co jest szczególnie istotne w środowiskach wieloplatformowych.

Na światowym rynku funkcjonuje wiele firm specjalizujących się w produkcji kluczy bezpieczeństwa, wśród których wyróżniają się marki takie jak Google, Thetis, Feitan czy Yubico 2.1.

Jednakże, na polskim rynku dominację sprawuje firma Yubico, będąca jedynym dystrybutorem tych urządzeń, co prowadzi do sytuacji zbliżonej do monopolu. W związku z tym dalsza część niniejszego opracowania koncentruje się wyłącznie na urządzeniach oferowanych przez Yubico [22].

Yubico oferuje cztery główne serie kluczy sprzętowych, które różnią się funkcjonalnością i przeznaczeniem:

- Security Key - najtańsza seria, stworzona z myślą o przeciętnym użytkowniku, obsługuje podstawowe standardy FIDO U2F i FIDO2,
- YubiKey 5 - najbardziej uniwersalna seria, umożliwiająca uwierzytelnianie przy użyciu różnych protokołów, takich jak FIDO2, U2F, OTP oraz smart card, co pozwala na szerokie zastosowanie w środowiskach osobistych i korporacyjnych,
- YubiKey 5 FIPS - dedykowana środowiskom o najwyższych wymaganiach bezpieczeństwa (np. instytucje rządowe i sektor finansowy), a normy FIPS (Federal Information Processing Standards) - gwarantują zgodność z międzynarodowymi standardami ochrony danych i kryptografii,
- YubiKey Bio - łączy tradycyjne metody uwierzytelniania z technologią biometryczną (np. wbudowany czytnik linii papilarnych). Oferuje te same funkcjonalności co Security Key Series, lecz z dodatkową warstwą zabezpieczenia opartą na weryfikacji biometrycznej.



Rysunek 2.7: Urządzenia uwierzytelniające YubiKey

Klucze YubiKey oferują szeroką gamę interfejsów komunikacyjnych takich jak USB-A, USB-C, Lightning oraz możliwość komunikacji przez NFC. W tabeli 2.2 dokonano porównania, wybierając klucz z każdej serii wyposażony w standardowe złącze USB-A, jednak należy pamiętać,

że podane ceny oraz oferta produktów mogą ulec zmianie w przyszłości. Wybierając klucz należy zwrócić uwagę na maksymalną liczbę poświadczeń, jakie dany klucz może obsłużyć, co również zostało ukazane.

Tabela 2.2: Porównanie serii urządzeń YubiKey

seria	Łączność	ilość slotów FIDO2	średnia cena
Security Key	USB-A, NFC	100	130zł
YubiKey 5	USB-A, NFC	100	240zł
YubiKey Bio	USB-A	100	470zł
YubiKey 5 FIPS	USB-A, NFC	25	470zł

Cechami wspólnymi wszystkich serii YubiKey są wysoka wytrzymałość i niezawodność. Urządzenia te są odporne na działanie wody, zgniatanie oraz osadzanie się pyłu. Ponadto, brak konieczności stosowania baterii oraz eliminacja ruchomych elementów minimalizują ryzyko awarii, podnosząc ogólną trwałość.

Para kluczy kryptograficznych w FIDO2

Kryptografia asymetryczna, wykorzystywana w standardzie FIDO2, opiera się na dwóch powiązanych ze sobą kluczach kryptograficznych: kluczu publicznym i kluczu prywatnym. Każdy z nich pełni określoną funkcję, a ich wzajemna zależność umożliwia bezpieczne uwierzytelnianie użytkownika.

Klucz publiczny jest przechowywany na serwerze uwierzytelniającym i służy do sprawdzania, czy podpis przesłany przez użytkownika został wygenerowany odpowiednim kluczem prywatnym. Dzięki swojej jawności, klucz publiczny nie wymaga szczególnej ochrony, ponieważ jego ujawnienie osobom nieuprawnionym nie umożliwia przeprowadzenia fałszywego uwierzytelnienia. Może być on zatem bezpiecznie przesyłany w sieci oraz przechowywany w bazie danych serwera.

Klucz prywatny jest przechowywany wyłącznie na urządzeniu użytkownika lub w autentykatorze, dzięki czemu pozostaje ściśle chroniony przed nieautoryzowanym dostępem. W praktyce oznacza to, że klucz ten nigdy nie opuszcza lokalnego środowiska i nie jest narażony na przechwycenie w trakcie transmisji. W zależności od urządzenia, dane mogą być przechowywane w specjalnym module sprzętowym lub bezpośrednio w pamięci klucza bezpieczeństwa [16].

Proces rejestracji i autoryzacji w FIDO2

Oba procesy opierają się na ścisłej współpracy trzech kluczowych elementów: przeglądarki, serwera oraz dedykowanego urządzenia uwierzytelniającego.

Ceremonia rejestracji rejestracja to pierwszy etap, w którym użytkownik tworzy konto w systemie lub dodaje nowy klucz uwierzytelniający do już istniejącego konta

1. Inicjacja rejestracji

Proces rozpoczyna się, gdy użytkownik wybiera opcję rejestracji klucza w na stronie internetowej. Serwer przesyła do przeglądarki prośbę o rozpoczęcie rejestracji. Prośba zawiera szczegóły dotyczące wymagań uwierzytelnienia, takich jak preferowane metody i polityki bezpieczeństwa,

2. Generowanie pary kluczy

Przeglądarka komunikuje się z urządzeniem uwierzytelniającym za pomocą protokołu CTAP [8]. W odpowiedzi autentykator generuje:

- klucz prywatny - który jest przechowywany lokalnie na urządzeniu w bezpiecznym środowisku,
- klucz publiczny - który zostaje zwrócony do przeglądarki,

3. Przesyłanie i zapis klucza publicznego

Klucz publiczny, identyfikator użytkownika oraz inne metadane są przesyłane do serwera. Serwer zapisuje klucz publiczny w swojej bazie danych i przypisuje go do konta użytkownika,

4. Potwierdzenie zakończenia rejestracji

Po zakończeniu rejestracji serwer informuje użytkownika o pomyślnym dodaniu urządzenia uwierzytelniającego. Od tego momentu użytkownik może korzystać z FIDO2 do logowania w systemie.

Ceremonia autentyfikacji

Autentyfikacja to proces, w którym użytkownik loguje się na swoje konto za pomocą zarejestrowanego urządzenia uwierzytelniającego. W tym procesie klucz prywatny i klucz publiczny są wykorzystywane do bezpiecznego potwierdzenia tożsamości.

- Inicjacja autoryzacji

Użytkownik wybiera opcję logowania w systemie. Serwer generuje **wyzwanie** (challenge), czyli unikalny jednorazowy ciąg znaków, który jest wysyłany do przeglądarki,

- Przekazanie wyzwania do autentykatora

Przeglądarka przesyła wyzwanie do urządzenia uwierzytelniającego za pośrednictwem protokołu CTAP. Użytkownik zostaje poproszony o potwierdzenie swojej tożsamości za pomocą lokalnego mechanizmu uwierzytelniania, takiego jak wprowadzenie PIN-u, odcisk palca czy skan twarzy,

- Podpisanie wyzwania

Po pomyślnej weryfikacji użytkownika autentykator używa swojego klucza prywatnego do podpisania wyzwania. Podpis zawiera unikalne dane potwierdzające, że operacja pochodzi od zarejestrowanego użytkownika,

- Weryfikacja podpisu przez serwer

Podpisane wyzwanie jest przesyłane z powrotem na serwer. Serwer wykorzystuje wcześniej zapisany klucz publiczny, aby zweryfikować podpis. Jeśli podpis jest poprawny, oznacza to, że użytkownik jest właścicielem klucza prywatnego, a zatem uwierzytelnienie przebiega pomyślnie,

- Zatwierdzenie autoryzacji

Po weryfikacji serwer umożliwia dostęp do konta lub zasobów użytkownika.

Rozdział 3

Specyfikacja FIDO2

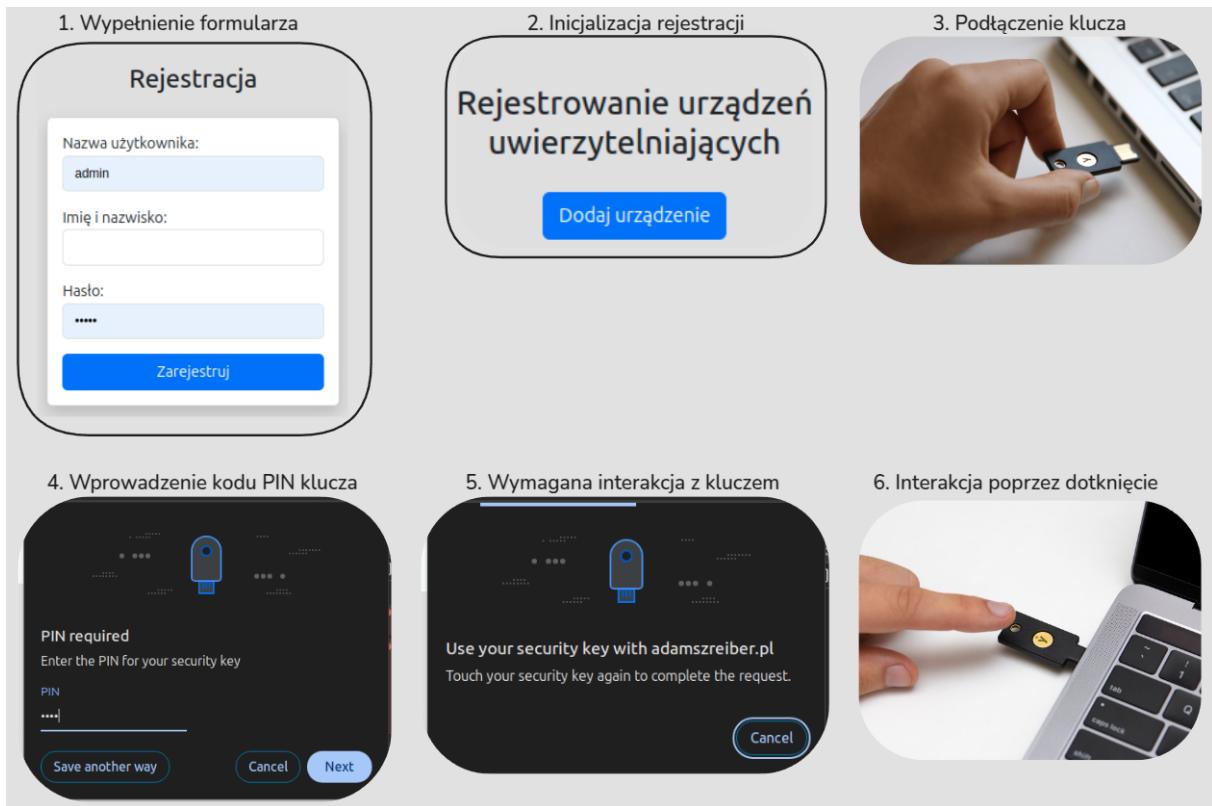
FIDO2 jest stosunkowo nowa technologia z obszaru bezpieczeństwa, która dopiero zaczyna zyskiwać uwagę w literaturze specjalistycznej. Ze względu na jej świeży charakter, liczba publikacji naukowych czy praktycznych raportów opisujących szczegółowo tę technologię jest ograniczona. W rezultacie badania wymagały sięgnięcia zarówno po klasyczne bazy naukowe, jak i dokumentacje techniczne - które łącznie stanowiły kompleksowy zasób wiedzy. Aby zapewnić wysoki poziom merytoryczny pracy, każda z pozyskanych publikacji została bardzo skrupulatnie przeanalizowana pod kątem jej przydatności. Celem było nie tylko odfiltrowanie treści pozbawionych wartości naukowej, ale również wyodrębnienie konkretnych wskazówek, rozwiązań i praktyk, które mogą zostać wykorzystane do wyjaśnienia działania, wdrożenia opisywanej technologii. Dzięki szczegółowej analizie możliwe było zbudowanie solidnej podstawy teoretycznej, niezbędnej do implementacji i weryfikacji założeń technologii w kolejnych częściach pracy.

Dokumentacja techniczna opracowana przez organizację W3C, pełni rolę głównego punktu odniesienia zarówno w kontekście podejścia deweloperskiego, jak i aspektów użytkowych. Zawarte w niej wytyczne precyzują sposób wdrażania omawianej technologii, zakłada, że proces ten powinien przebiegać w określonych etapach, gwarantując zgodność z przyjętymi standardami. Z perspektywy użytkownika rejestracja nowego urządzenia autoryzującego - jakim np. jest klucz bezpieczeństwa niesie za sobą następujące kroki:

- użytkownik przechodzi na stronę, tworzy nowe konto,
- przechodzi do sekcji "Zarejestruj klucz bezpieczeństwa",
- strona internetowa prosi użytkownika o włożenie klucza do portu USB,

- klucz miga dając tym samym znak do kliknięcia w fizyczny przycisk,
- strona internetowa wyświetla wiadomość "Zakończono rejestrowanie".

Poniżej przedstawiono wizualizację tego procesu:

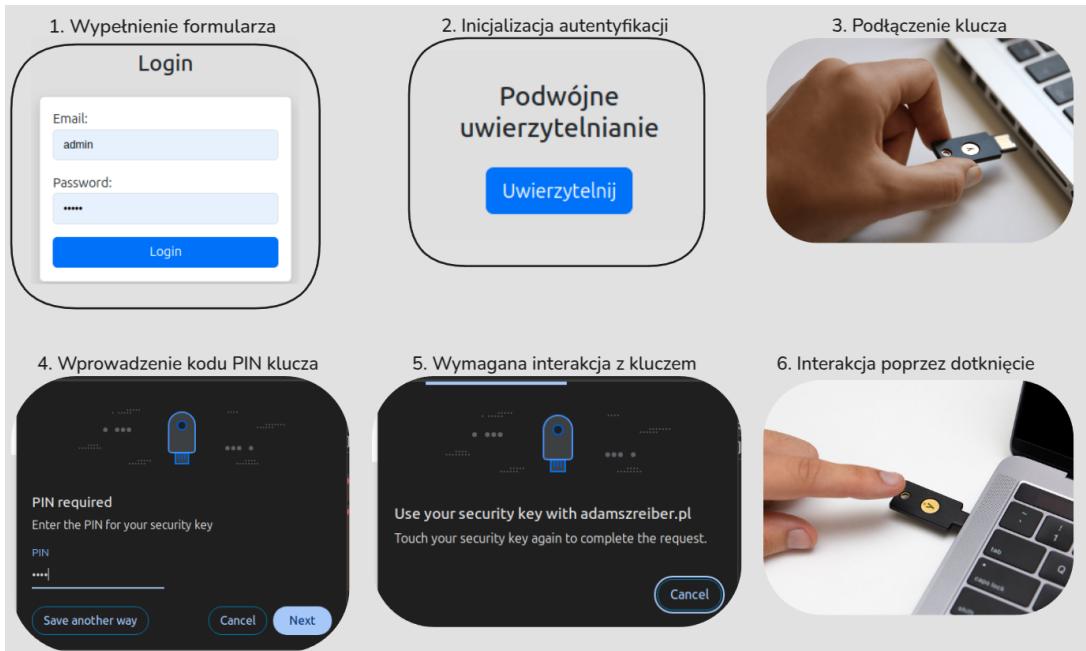


Rysunek 3.1: Proces rejestracji z perspektywy użytkownika.

W procesie uwierzytelniania można wyróżnić następujące czynności:

- użytkownik przechodzi na stronę, loguje się do systemu,
- inicjuje akcję autoryzacji,
- strona internetowa prosi użytkownika o włożenie klucza do portu USB,
- klucz miga dając tym samym znak do kliknięcia w fizyczny przycisk,
- strona internetowa wyświetla wiadomość "Użytkownik zalogowany".

Proces ten został zobrazowany poniżej:



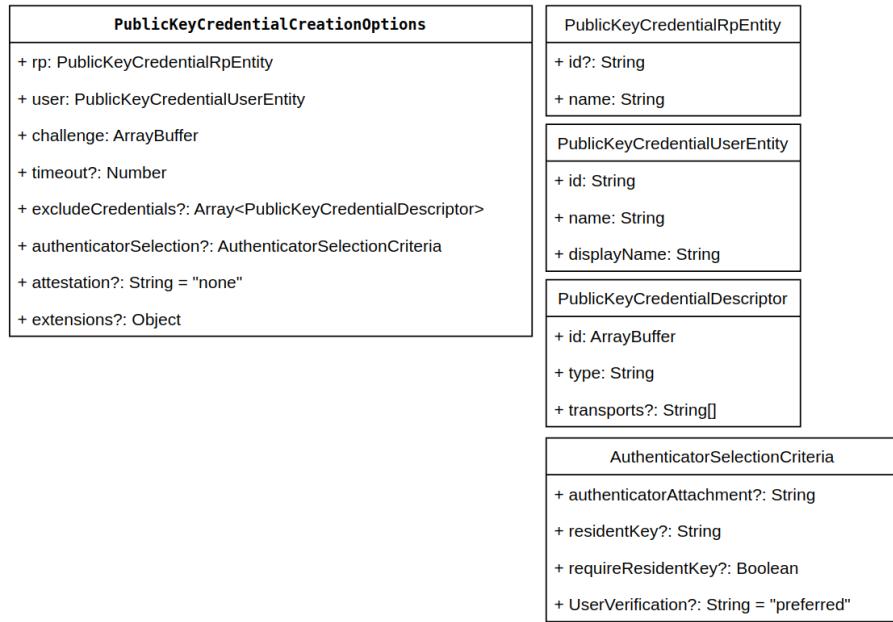
Rysunek 3.2: Procesu uwierzytelniania z perspektywy użytkownika

Z funkcjonalnego punktu widzenia rejestracja nowego klucza i autoryzacja za pomocą klucza już istniejącego przebiegają niemal identycznie. Choć użytkownik uczestniczy w dwóch odmiennych procesach, to zestaw czynności, które musi wykonać, się nie zmienia.

3.1 Rejestracja

Zgoła inaczej wygląda to „za kulisami” interfejsu użytkownika, czyli w warstwie komunikacji klient – serwer. Rejestracja urządzenia autoryzującego przebiega według ustalonego schematu, który rozpoczyna się, gdy użytkownik odwiedza stronę zawierającą niezbędną skrypt. W odpowiedzi na to zdarzenie przeglądarka nawiązuje komunikację z serwerem, inicjując procedurę rejestracyjną. Niezależnie od tego, czy użytkownik jest już autoryzowany w serwisie, czy dopiero rozpoczyna procedurę tworzenia konta, kolejne etapy przebiegają według zaplanowanej sekwencji:

1. Żądanie wygenerowania opcji rejestracyjnych - skrypt w przeglądarce wysyła zapytanie do serwera, prosząc o utworzenie struktury PublicKeyCredentialCreationOptions,



Rysunek 3.3: Struktura danych - PublicKeyCredentialCreationOptions

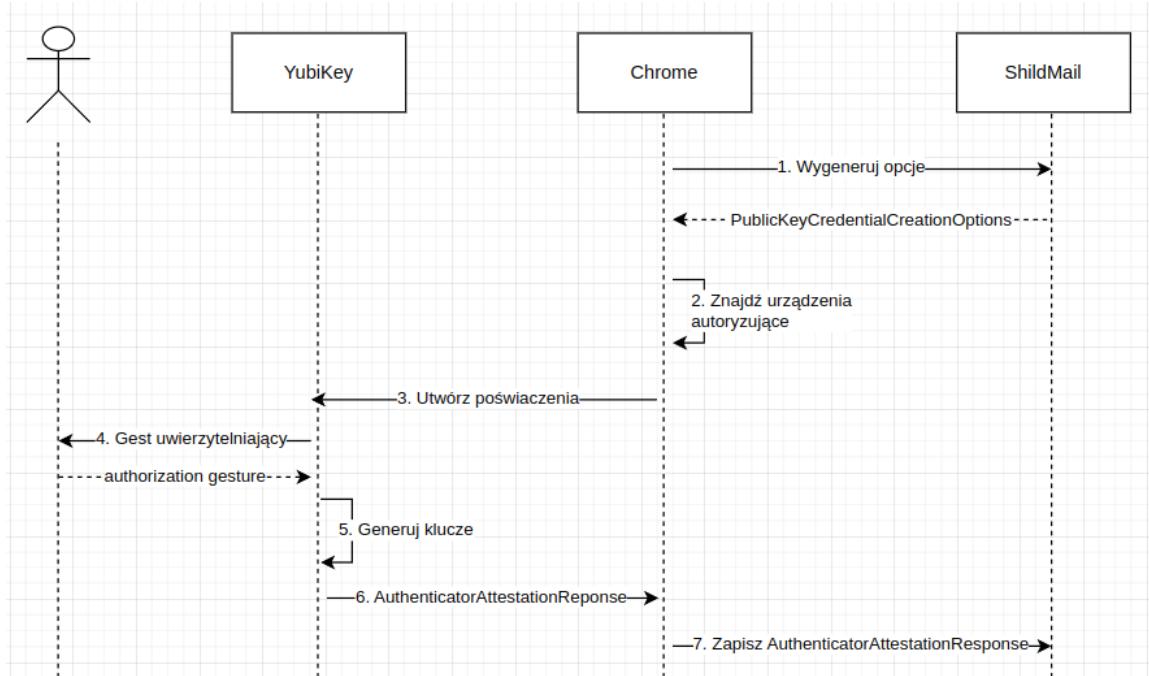
2. Przeglądarka identyfikuje dostępne urządzenia autoryzujące. Przekazuje struktury do urządzenia autoryzującego - uzyskane z serwera informacje są przesyłane do warstwy odpowiedzialnej za komunikację z kluczem bezpieczeństwa,
3. Klient nawiązuje komunikację z wybranym urządzeniem,
4. Sygnalizacja potrzebnej akcji użytkownika - klucz bezpieczeństwa daje o sobie znać, np. poprzez miganie diody, wzywając do tzw. gestu uwierzytelniającego, czyli fizycznego potwierdzenia (dotknięcia przycisku, wprowadzenia PIN-u, użycia czytnika linii papilarnych itp.). Ma to zapewnić, że użytkownik świadomie inicjuje proces tworzenia poświadczania,
5. Utworzenie poświadczania i wygenerowanie odpowiedzi - po potwierdzeniu przez użytkownika, klucz generuje nowe dane uwierzytelniające (m.in. parę kluczy kryptograficznych),
6. Następnie tworzona jest struktura AuthenticatorAttestationResponse, zawierająca informacje niezbędne do weryfikacji autentyczności i poprawności poświadczania i przekazywana do przeglądarki,

AuthenticatorAttestationResponse	CollectedClientData
+ attestationObject: ArrayBuffer	+ type: "webauthn.create" "webauthn.get"
+ clientDataJSON: ArrayBuffer<CollectedClientData>	+ challange: String
	+ origin: String
	+ crossOrigin?: Boolean
	+ tokenBinding?: TokenBinding
	TokenBinding
	+ status?: String
	+ id: String

Rysunek 3.4: Struktura danych - AuthneticatorAttestationResponse

7. Przeglądarka wysyła AuthenticatorAttestationResponse do serwera w celu ich zapisania.

Przebieg tego procesu można prześledzić na diagramie komunikacji przedstawionym w rysunku 3.5, który ilustruje wymianę komunikatów między przeglądarką, serwerem a urządzeniem autoryzującym w poszczególnych etapach.



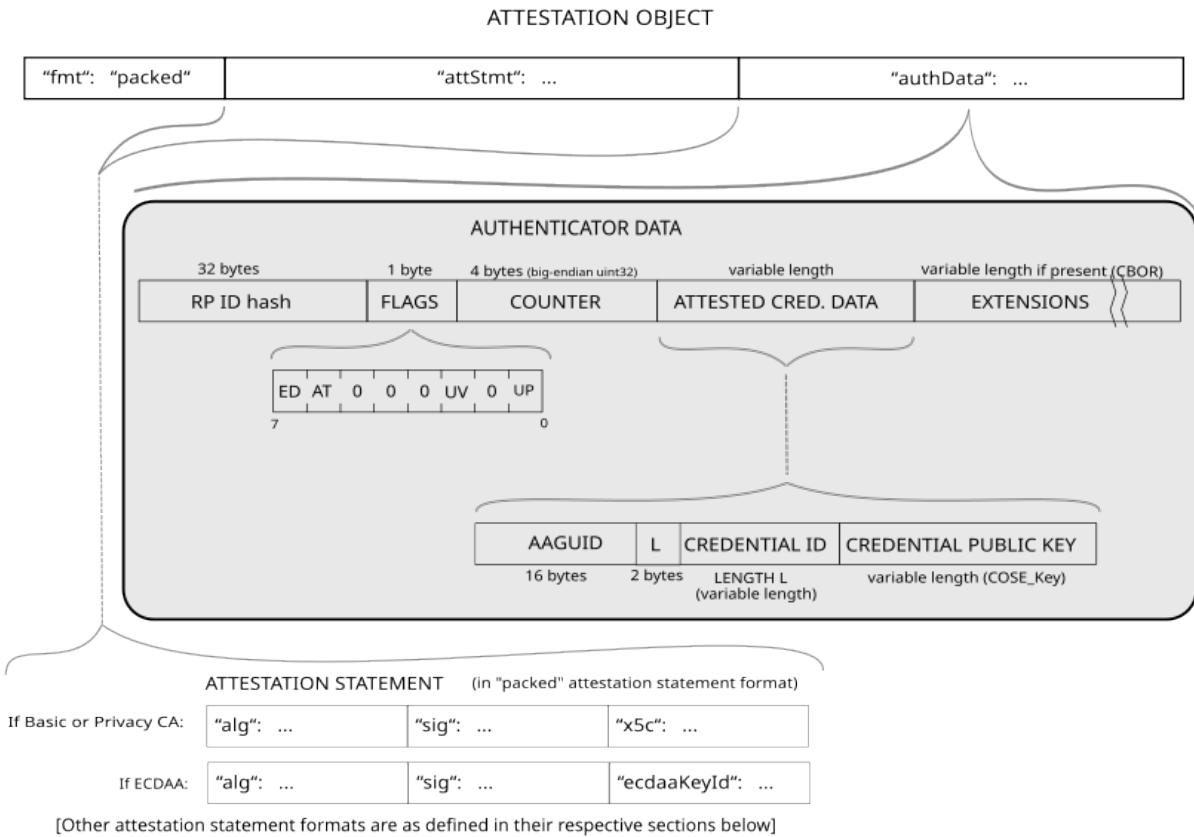
Rysunek 3.5: Proces rejestracji klucza

Po uzyskaniu zarejestrowanego poświadczenia w systemie, użytkownik może przeprowadzić proces autoryzacji, który – choć z pozoru podobny do rejestracji – opiera się na weryfikacji posiadanej już klucza bezpieczeństwa.

Podczas procesu rejestracji klucza FIDO2, przeglądarka generuje i przekazuje do serwera dwie kluczowe struktury danych: clientDataJSON oraz attestationObject. Pole clientDataJSON zawiera informacje kontekstowe związane z operacją rejestracji, takie jak źródło żądania (origin), wyzwanie wysłane przez serwer oraz typ operacji. Jest to ciąg znaków w formacie JSON, zakodowany w Base64URL, który jest tworzony przez przeglądarkę na podstawie danych dostarczonych przez serwer oraz kontekstu operacji.

Z kolei pole attestationObject to binarna struktura, zakodowana w formacie CBOR (Concise Binary Object Representation), która została szczegółowo przedstawiona na rysunku 3.6. Składa się ona z danych uwierzytelniających (**authData**) oraz zaświadczenia potwierdzającego autentyczność urządzenia (**attStmt**). Dane uwierzytelniające to kluczowy element, który składa się z:

- Skrót identyfikatora (RP ID Hash) - skrót identyfikatora strony (Relying Party), który potwierdza, że operacja została wykonana dla właściwej domeny,
- Flagi (flags) - określają stan operacji, np. czy użytkownik został uwierzytelniony (flagi UP i UV),
- Licznik podpisów (counter) - który zwiększa się przy każdej operacji uwierzytelniania, pomaga wykryć próby ponownego wykorzystania przechwyconych komunikatów uwierzytelniających (reply attack),
- Identyfikator (AAGUID) - unikalny identyfikator urządzenia uwierzytelniającego,
- Id poświadczenia (Credential ID)- Unikalny identyfikator poświadczenia, który jest używany do powiązania klucza publicznego z kontem użytkownika,
- Klucz publiczny (Credential Public Key) - wygenerowany przez urządzenie uwierzytelniające, który jest używany do weryfikacji podpisów podczas uwierzytelniania.



Rysunek 3.6: Dane opisujące urządzenie autentyfikujące [20]

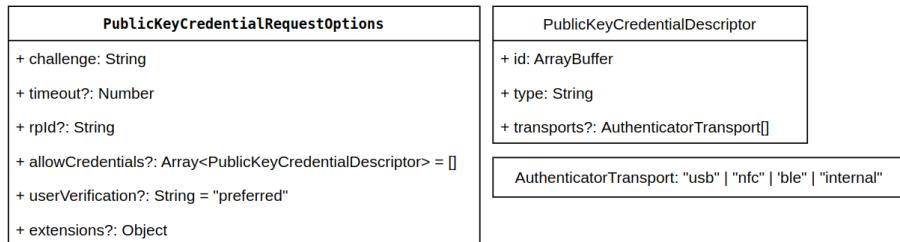
Struktura **attestationObject** jest generowana przez urządzenie uwierzytelniające i przekazywana do przeglądarki. Klucz publiczny, jest osadzony wewnątrz `attestationObject` w polu `authData`. Po pomyślnej weryfikacji danych przez serwer, klucz publiczny jest zapisywany w bazie danych serwera i powiązany z kontem użytkownika. Dzięki temu może w przyszłości wykorzystać ten klucz do weryfikacji podpisów podczas procesu uwierzytelniania.

3.2 Autentykacja

Autentykacja oparta na specyfikacji FIDO inicjowana jest dopiero po pomyślnym przejściu etapu logowania przez użytkownika. Na tym etapie kluczową rolę odgrywa serwer, który weryfikuje tożsamość osoby próbującej uzyskać dostęp i automatycznie ogranicza listę akceptowanych urządzeń autoryzujących (`allowedCredentials`) do tych, które zostały wcześniej zarejestrowane i powiązane z danym kontem.

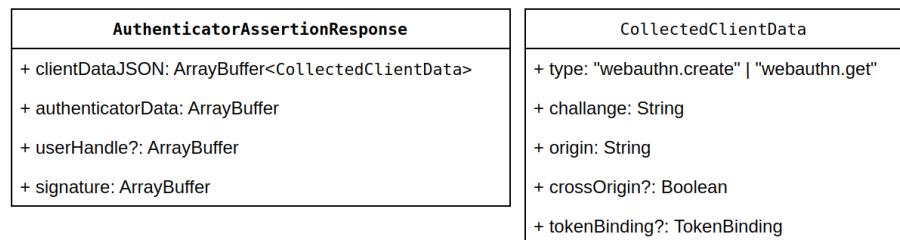
Przebieg całego procesu można zarysować w kilku kluczowych krokach:

1. Żądanie wygenerowania opcji autentykacji - skrypt w przeglądarce wysyła zapytanie do serwera, prosząc o utworzenie struktury PublicKeyCredentialRequestOptions



Rysunek 3.7: Struktura danych - PublicKeyCredentialRequestOptions

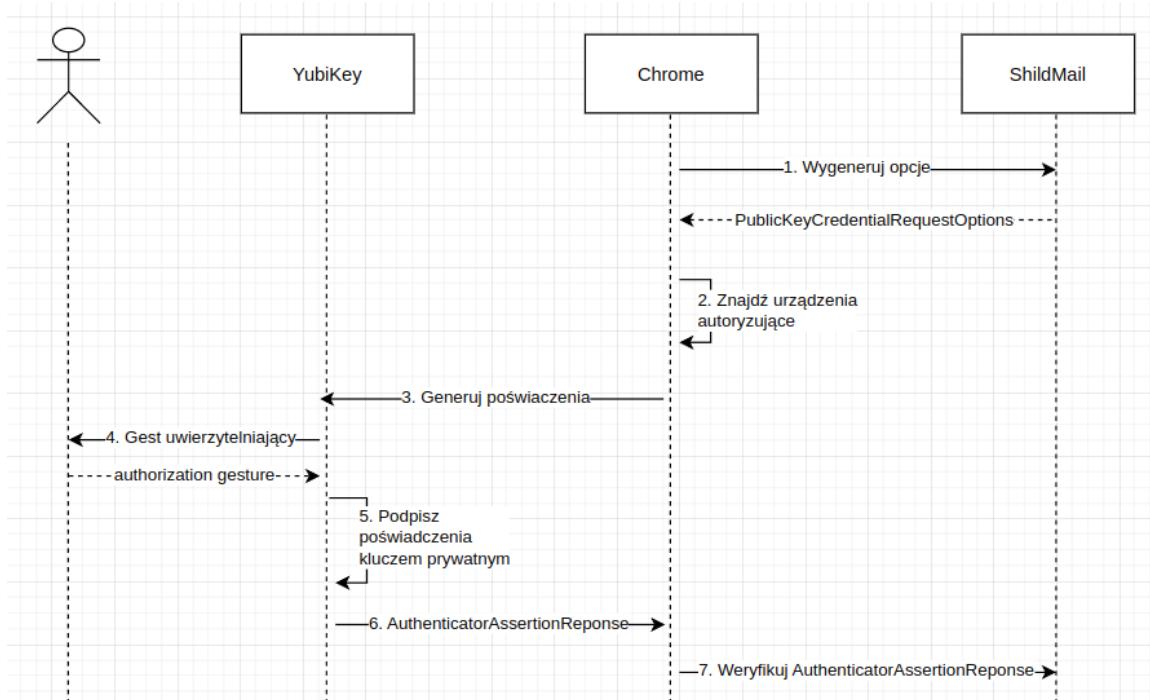
2. Na tym etapie przeglądarka identyfikuje dostępne urządzenia autoryzujące. Przekazuje struktury do urządzenia autoryzującego - uzyskane z serwera informacje są przesyłane do warstwy odpowiedzialnej za komunikację z kluczem bezpieczeństwa.
3. Klient (przeglądarka) nawiązuje komunikację z wybranym urządzeniem.
4. Sygnalizacja potrzebnej akcji użytkownika - klucz bezpieczeństwa daje o sobie znać, np. poprzez miganie diody, wzywając do tzw. „authorization gesture”, czyli fizycznego potwierdzenia (dotknięcie przycisku, wprowadzenia PIN-u, użycia czytnika linii papilarnych itp.).
5. Wykonanie podpisu i utworzenie odpowiedzi - Po uzyskaniu niezbędnego potwierdzenia od użytkownika, urządzenie przystępuje do wykorzystania istniejącej pary kluczy kryptograficznych.
6. Następnie generowana jest struktura AuthenticatorAssertionResponse, która zawiera dane niezbędne do weryfikacji autentyczności oraz poprawności poświadczenia. Struktura ta jest następnie przekazywana do przeglądarki.



Rysunek 3.8: Struktura danych - AuthenticatorAssertionResponse

7. Przeglądarka, transmituje otrzymaną strukturę z do serwera. Serwer dokonuje analizy otrzymanych danych, weryfikuje podpis cyfrowy oraz potwierdza zgodność. Jeżeli weryfikacja przebiegnie pomyślnie, użytkownik zostaje uwierzytelniony.

Proces autoryzacji można w pełni prześledzić, analizując diagram komunikacji ukazany na rysunku 3.9 który prezentuje zarówno podmioty biorące udział w wymianie danych (takie jak użytkownik, serwer czy urządzenie uwierzytelniające), jak i szczegółowe działania inicjowane w ramach systemu autoryzacji.

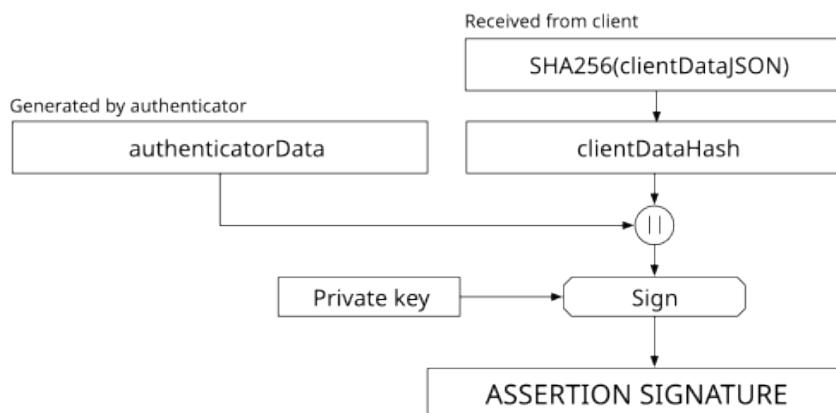


Rysunek 3.9: Proces autoryzacji użytkownika

Podczas procesu autentykacji, przeglądarka generuje i przekazuje do serwera dwie kluczowe struktury danych: **clientDataJSON** oraz **authenticatorData** (często nazywane authData). Pole **clientDataJSON** zawiera informacje kontekstowe związane z operacją uwierzytelniania, takie jak źródło żądania, wyzwanie wysłane przez serwer oraz typ operacji (webauthn.get). Jest to ciąg znaków w formacie JSON, zakodowany w Base64URL, który jest tworzony przez przeglądarkę na podstawie danych dostarczonych przez serwer oraz kontekstu operacji.

Pole **userHandle** jest opcjonalnym elementem, który służy do identyfikacji użytkownika. Jest to ciąg bajtów (zwykle zakodowany w Base64URL), który jest przekazywany z serwera do urządzenia uwierzytelniającego podczas rejestracji, a następnie może być zwrócony przez urządzenie podczas uwierzytelniania. Może to być np. unikalny identyfikator użytkownika (ID) z bazy danych.

Ostatnia wartość to **sygnature** która jest generowana przez urządzenie uwierzytelniające. Proces tworzenia sygnatury ukazano na rysunku 3.10 i składa się z kilku kluczowych etapów. Po pierwsze, przeglądarka przekazuje do urządzenia clientDataJSON, z którego obliczany jest skrót (hash), zazwyczaj przy użyciu algorytmu SHA-256, zwany clientDataHash. Następnie, urządzenie uwierzytelniające dokonuje konkatenacji dwóch elementów: authenticatorData oraz clientDataHash. Powstały w ten sposób ciąg bajtów jest podpisywany kluczem prywatnym.



Rysunek 3.10: Generowanie sygnatury (podpisu kryptograficznego) [20]

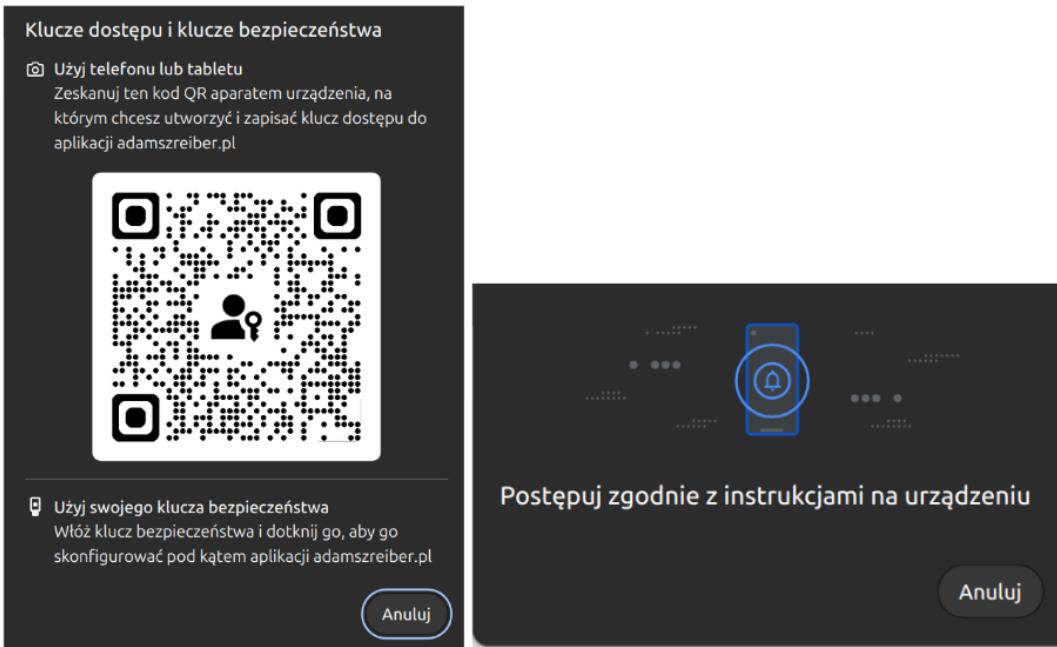
Sygnatura zostaje zwrócona przeglądarce, która przesyła ją do serwera. Serwer, mając publiczny klucz użytkownika, weryfikuje poprawność podpisu na podstawie tych samych danych, sprawdza spójność innych parametrów i tym samym potwierdza autentyczność oraz integralność całego procesu uwierzytelniania.

Dokumentacja pełni rolę kluczowego źródła informacji i standaryzacji, na którym opiera się wdrażanie wielu rozwiązań technologicznych. Jej dokładne zrozumienie i dogłębiańska analiza mechanizmów opisanych w poszczególnych specyfikacjach pozwala uniknąć potencjalnych błędów, a jednocześnie stanowi gwarancję, że wprowadzana implementacja jest zgodna z założeniami przyjętymi w ramach oficjalnie ustalonych standardów.

W kontekście weryfikacji poprawności implementacji oraz dalszych badań, warto uwzględnić perspektywę akademicką, która może istotnie wzbogacić praktyczne ujęcie tematu. Z tego powodu pomocna okazuje się analiza pracy zatytułowanej Fast Identity Online 2: Authentication Technique [3]. Opracowanie oparto głównie na przeglądzie dokumentacji oraz innych publikacji naukowych, skupiając się jednak przede wszystkim na ujęciu teoretycznym i nie obejmując praktycznych testów implementacyjnych.

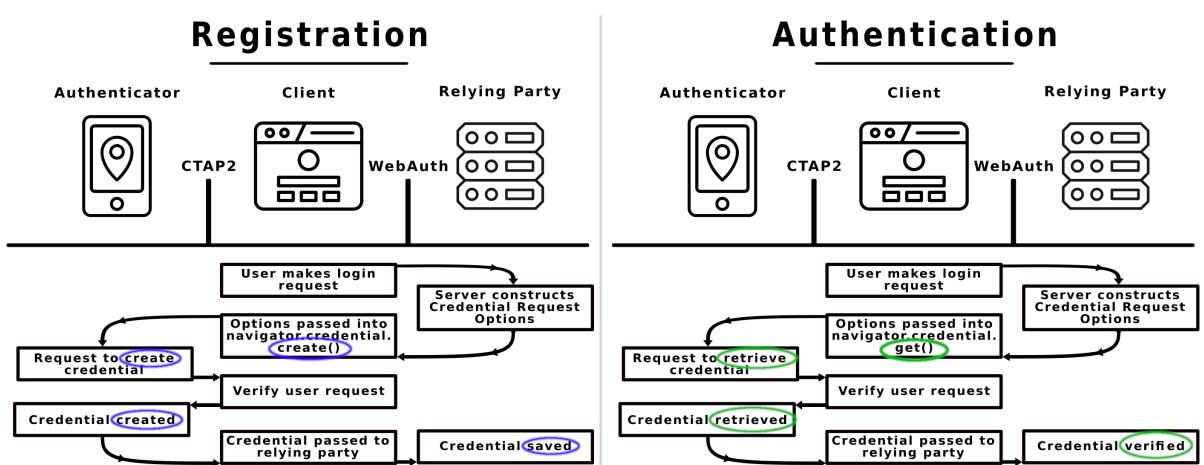
Przebadano w owej publikacji proces rejestracji oraz logowania w kontekście użytkownika który korzysta z przeglądarki oraz urządzenia mobilnego pełniącego rolę modułu uwierzytel-

niającego, w bezprzewodowym kanale komunikacyjnym jakim jest Bluetooth. Urządzenia autoryzujące w postaci smartfonów wnoszą nową perspektywę do specyfikacji FIDO2, głównie za sprawą ich powszechniej dostępności w codziennym życiu. Dzięki temu zamiast wymagać dedykowanych, specjalistycznych kluczy sprzętowych, może opierać się na posiadanych już przez użytkowników urządzeniach mobilnych. Choć podstawowy schemat rejestracji i uwierzytelniania pozostaje zgodny z opisany w dokumentacji, w owym przypadku dochodzi dodatkowy element interakcji między przeglądarką a smartfonem. Po stronie przeglądarki generowany jest kod QR wyświetlony w natywnym oknie. Użytkownik skanuje ten kod za pomocą telefonu, co inicjuje proces uwierzytelniania na urządzeniu mobilnym. W zależności od skonfigurowanej w telefonie metody zabezpieczenia – może to być skan odcisku palca, Face ID lub wprowadzenie numeru PIN – użytkownik zostaje poproszony o potwierdzenie swojej tożsamości. Po pomyślnym zakończeniu tego etapu smartfon automatycznie przekazuje informacje z powrotem do przeglądarki, która potwierdza uwierzytelnienie i umożliwia dalsze korzystanie z serwisu. Taki model, oparty na mechanizmie skanowania kodu QR i weryfikacji, nieznacznie różni się od standardowego schematu WebAuthn wyłącznie koniecznością nawiązania dodatkowego połączenia pomiędzy przeglądarką a telefonem, jednak nie narusza zasad i struktur opisanych w specyfikacji.



Rysunek 3.11: Autentyfikacja Bluetooth: instrukcje w przeglądarce

Graficzne podsumowanie przebiegu procesu rejestracji i weryfikacji tożsamości, zostało przedstawione na ilustracjach, a różnice występujące w każdym z kroków zaznaczono kolorem – co ułatwi zrozumienie kluczowych etapów oraz mechanizmów działania omawianych rozwiązań.



Rysunek 3.12: Schemat rejestracji i autoryzacji w publikacji [3]

Podczas rejestracji używana jest funkcja `credential.create`, co sugeruje, że proces ten polega na tworzeniu nowego poświadczenia. Natomiast w przypadku logowania wykorzystywana jest funkcja `credential.get`, która odpowiada za wyszukanie istniejącego poświadczenia. Autentykator, gdy otrzymana odpowiednią komendę, rozpoczyna wykonywanie wskazanego mechanizmu

i żąda weryfikacji ze strony użytkownika. Po weryfikacji, autentykator wykonuje otrzymane polecenie, odpowiednio tworząc nowe poświadczenie lub wyszukując istniejące. Następnie, poświadczenia te są przekazywane do przeglądarki, a dalej kierowane do serwera, gdzie są rejestrowane lub poddawane weryfikacji.

W wymienionej publikacji szczególną uwagę poświęca się analizę zabezpieczeń przed atakami phishingowymi, m.in. poprzez weryfikację zgodności domen czy zastosowanie licznika logowań, który pozwala rozpoznać ewentualne próby podszywania się pod uprawnione urządzenie.

Zestawienie dwóch różnych perspektyw – praktycznego podejścia opartego na dokumentacji oraz ujęcia teoretycznego przedstawionego w publikacji Fast Identity Online 2: Authentication Technique – pozwoliło na uzyskanie szerszego obrazu badanego zagadnienia. W efekcie połączenie doświadczeń płynących z praktycznej integracji protokołów i mechanizmów autoryzacyjnych z akademicką analizą teoretycznych założeń zapewniło solidną podstawę do wnioskowania oraz dalszych eksperymentów.

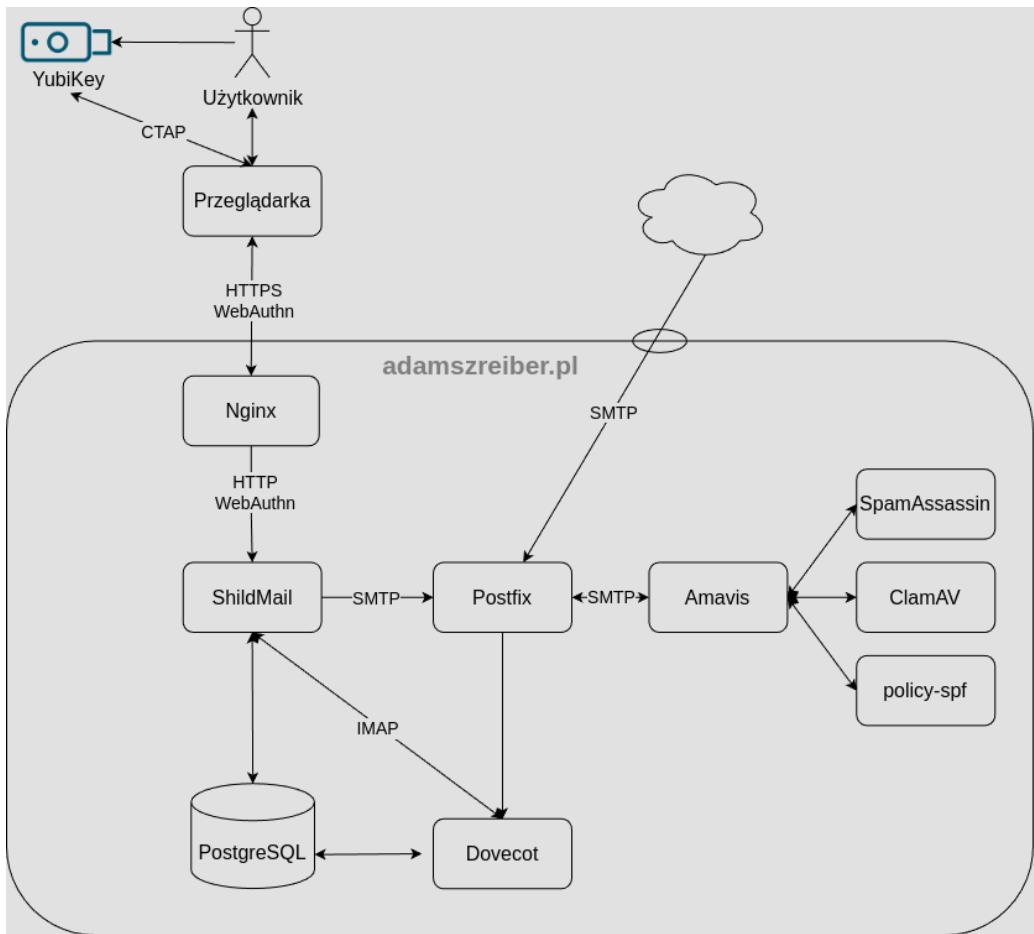
Rozdział 4

Implementacja platformy badawczej

W ramach badań nad opracowano projekt usługi poczty elektronicznej, którą na potrzeby projektu nazwano **ShieldMail**. Aplikacja została zaprojektowana z uwzględnieniem zaawansowanych mechanizmów bezpieczeństwa, w tym uwierzytelniania dwuskładnikowego. Pierwszy etap opiera się na tradycyjnej metodzie uwierzytelniania za pomocą sekretnych danych - hasło, natomiast jako drugi zaimplementowano nowoczesną technologię FIDO2. ShieldMail stanowi praktyczne środowisko testowe, umożliwiające ocenę skuteczności połączenia tych mechanizmów w kontekście bezpieczeństwa systemów informatycznych.

4.1 Architektura projektu

W celu realizacji projektu wynajęto serwer chmurowy, który dostępny jest pod domeną **adamszreiber.pl**. Do stworzenia serwisu e-mail wykorzystano otwartoźródłowe komponenty. Otwartoźródłowy charakter komponentów umożliwia ich transparentną analizę - redukuje to ryzyko ukrytych podatności i pozwala na dostosowanie ich do specyficznych wymagań projektu.



Rysunek 4.1: Architektura systemu Shildmail

Serwis e-mail zbudowano od podstaw, zaczynając od instalacji systemu operacyjnego Linux Ubuntu 24.04 LTS, a następnie integracji poszczególnych elementów, takich jak Postfix, Dovecot, Amavis, SpamAssassin, ClamAV oraz mechanizmy weryfikacji autentyczności nadawcy (DKIM, SPF). Każdy z tych elementów wymagał szczególowej konfiguracji, aby poprawnie współpracował w obrębie całego ekosystemu. Wdrożenie obejmowało między innymi dostosowanie ustawień serwera pocztowego, filtrowanie wiadomości, skanowanie antywirusowe oraz konfigurację odpowiednich rekordów w systemie nazw domen (DNS). Dzięki temu uzyskano w pełni funkcjonalny serwis e-mail, zgodny z wymaganiami tematu pracy, a także skonfigurowany w taki sposób, by umożliwiał komunikację z innymi serwerami pocztowymi i wymianę wiadomości w obu kierunkach.

Postfix to otwartoźródłowy serwer pocztowy (MTA – Mail Transfer Agent), który obsługuje wysyłanie i odbieranie wiadomości e-mail za pośrednictwem protokołu SMTP (Simple Mail Transfer Protocol). Przyjmuje wiadomości e-mail od klientów pocztowych oraz innych serwerów pocztowych. Decyduje, jak i gdzie dostarczyć wiadomość na podstawie konfiguracji

domen i rekordów DNS. Postfix współpracuje z dodatkowymi komponentami, tj: integracja z filtrami i analizatorami.

Dovecot - oprogramowanie z otwartym kodem źródłowym, serwer IMAP i POP3, przeznaczony do obsługi skrzynek pocztowych. Jego głównym celem jest zapewnienie bezpiecznego, wydajnego i niezawodnego **dostępu do wiadomości e-mail** przechowywanych na serwerze.

Amavis - pełni rolę interfejsu pośredniczącego pomiędzy serwerem pocztowym Postfix, a narzędziami filtrującymi. Po odebraniu wiadomości od MTA, Amavis przetwarza je, dodając odpowiednie nagłówki oraz metadane, co umożliwia wielopoziomową analizę wiadomości.

SpamAssassin - narzędzie do analizy i filtrowania spamu, stworzone z myślą o identyfikacji i klasyfikacji niechcianych wiadomości e-mail. Działa jako system oceniający każdą wiadomość na podstawie różnych reguł, heurystyk, sygnatur i analiz statystycznych, przypisując jej wynik punktowy określający, czy wiadomość jest spamem

ClamAV - zaawansowane oprogramowanie antywirusowe, którego głównym celem jest efektywne wykrywanie i usuwanie zagrożeń w różnych środowiskach informatycznych. Zostało zaprojektowane tak, aby sprostać szerokiemu wachlarzowi zastosowań, dzięki czemu doskonale sprawdza się w systemach pocztowych, serwerach plików oraz innych infrastrukturach wymagających szczegółowej analizy pod kątem obecności złośliwego oprogramowania.

python-policyd-spf serwer polityki dla Postfix napisany w języku Python, którego celem jest potwierdzenie, że nadawca wiadomości jest upoważniony do wysyłania e-maili w imieniu określonej domeny (polityka SPF). Jest to narzędzie zgodne z mechanizmem polityki Postfix, które działa jako zewnętrzny proces, analizujący wiadomości w czasie rzeczywistym i pomagający w ochronie przed podszywaniem się pod domeny nadawców.

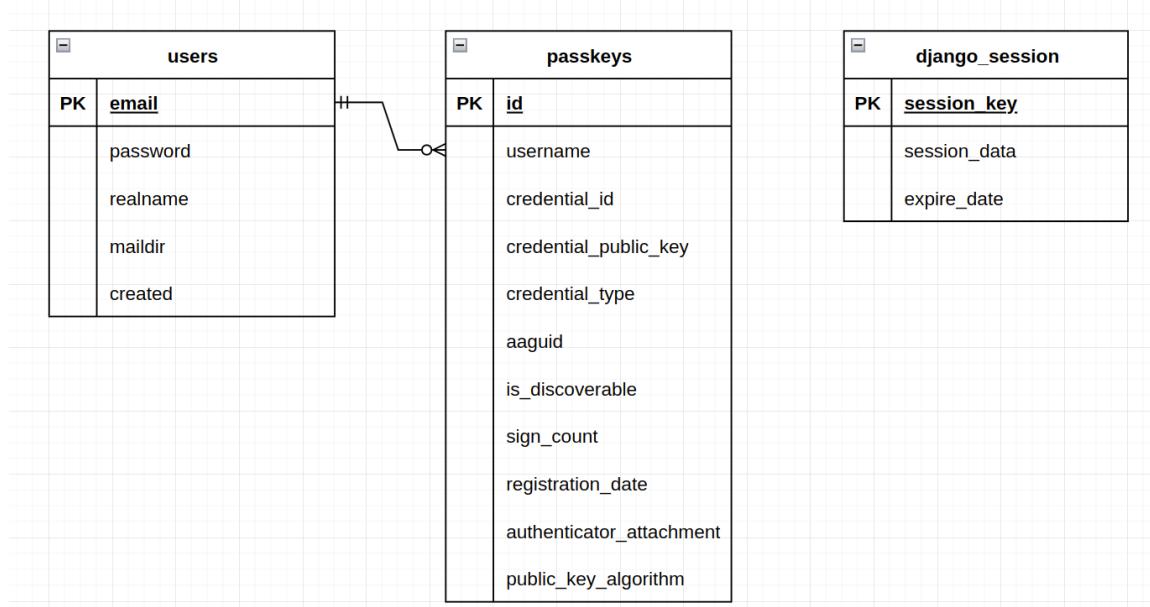
PostgreSQL to zaawansowany system zarządzania relacyjnymi bazami danych, który jest dostępny jako oprogramowanie otwartoźródłowe. Wspiera większość standardów SQL, oraz niestandardowe typy, takie jak dane binarne, co czyni go idealnym wyborem dla aplikacji Shild-Mail.

Nginx to wszechstronny serwer HTTP, który znajduje szerokie zastosowanie w zapewnianiu bezpieczeństwa i wydajności w środowiskach sieciowych. Jednym z najważniejszych jego zastosowań jest działanie jako brama, która nie tylko przekazuje żądania do aplikacji serwera-wych, ale również zapewnia szyfrowanie komunikacji.

Shildmail - centralna aplikacja systemu pocztowego, pełniąca rolę głównego węzła komunikacyjnego, napisana w języku Python przy użyciu frameworka Django. Jej zadaniem jest zapewnienie użytkownikom bezpiecznego i łatwego dostępu do skrzynek pocztowych oraz możliwości wysyłania wiadomości e-mail. Aplikacja obsługuje uwierzytelnianie dwuskładnikowe, które opiera się na tradycyjnej metodzie bazującej na sekrecie, a także na nowoczesnym roz-

wiązaniu wykorzystującym specyfikację FIDO2. Do implementacji tej specyfikacji wykorzystano bibliotekę **Pythonic WebAuthn** [12] stworzoną przez Duo Labs.

Komunikacja użytkownika z aplikacją odbywa się za pośrednictwem przeglądarki internetowej, gdzie ShieldMail udostępnia interfejs użytkownika. Aplikacja pozwalała na odbieranie i wysyłanie wiadomościami e-mail. Ponadto, interfejs ten zawiera skrypt inicjujący procedury rejestracji i autentykacji FIDO2, który implementuje rozwiążanie oparte na bibliotece **simplewebauthn** [13]. Proces uwierzytelniania w aplikacji bazuje na mechanizmie sesji, przechowywanej po stronie przeglądarki w plikach cookie. Po zalogowaniu użytkownika, aplikacja centralna przejmuje rolę pośrednika między nim a serwerami pocztowymi, obsługującymi protokoły SMTP i IMAP. Wszystkie dane użytkowników, w tym informacje uwierzytelniające, sesje oraz klucze bezpieczeństwa, są przechowywane w relacyjnej bazie danych PostgreSQL.



Rysunek 4.2: Schemat bazy danych aplikacji Shildmail

Przeglądarka - Przeglądarka internetowa pełni rolę interfejsu użytkownika w komunikacji z serwisem ShieldMail, umożliwiając bezpośredni dostęp do funkcji zarządzania pocztą elektroniczną. Bierze udział także w autentykacji – jej zadaniem jest komunikacja z kluczem bezpieczeństwa, w celu przeprowadzenia procesu uwierzytelniania

Przeglądarka	Wspiera specyfikację FIDO	Minimalna wersja oprogramowania
Chrome	Tak	67
Firefox	Tak	60
Brave	Tak	od początku
Safari	Tak	13
Opera	Tak	54
Edge	Tak	18

Tabela 4.1: Przeglądarki wspierające specyfikację FIDO [21]

4.2 Ochrona przed spamem i wirusami

Po otrzymaniu wiadomości e-mail, serwer pocztowy Postfix, zgodnie z ustalonymi zasadami konfiguracji, przekazuje ją do zewnętrznego modułu filtrującego Amavis. Integracja ta realizowana jest przy użyciu protokołu SMTP, co umożliwia przeprowadzenie szczegółowej analizy treści wiadomości w celu wykrycia spamu, wirusów oraz innych potencjalnych zagrożeń.

Amavis jest niejako bramą do systemów analizy treści. Wiadomość mail która trafia poddawana jest analizie trzem odrębnym systemom. Przesyłana jest do **ClamAV** który sprawdza na podstawie swojej bazy sygnatur wirusów oraz heurystyk, załączniki i treść wiadomości pod kątem obecności wirusów, trojanów, ransomware oraz innych złośliwych plików. W rezultacie jeśli ClamAV wykryje złośliwe oprogramowanie, Amavis może oznaczyć wiadomość jako niebezpieczną lub ją odrzucić.

Następnie Wiadomość jest analizowana przez **SpamAssassin** który ocenia ją pod kątem cech spamu. Dokładnie analizuje zawartość wiadomości e-mail, wyszukując charakterystyczne dla spamu słowa kluczowe. Przeszukuje nagłówki wiadomości w celu wykrycia manipulacji. Sprawdza obecność podejrzanych linków i załączników, które mogą stanowić zagrożenie. Odpowiedź zawiera wynik punktowy który określa czy wiadomość jest spamem.

Ostatnim etapem weryfikacji jest, **polityka SPF** która pozwala na określenie, jakie serwery są uprawnione do wysyłania wiadomości w imieniu danej domeny. Dzięki temu można zweryfikować autentyczność nadawcy i zapobiegać fałszerstwom.

Po zakończeniu analiz, Amavis podejmuje decyzję na podstawie zebranych wyników. Gdy wiadomość zostanie uznana za bezpieczną, dostarczana jest do odbiorcy. W przypadku stwierdzenia, że wiadomość to spam, może albo ją odrzucić, albo oznaczyć w nagłówkach jako spam i dostarczyć do odbiorcy, do folderu „Spam”. Jeśli zaś wiadomość zawiera złośliwe oprogramowanie, jest natychmiast odrzucana lub przenoszona do kwarantanny, aby zapobiec potencjalnym zagrożeniom.

Ochrona poczty elektronicznej to złożony proces, w którym współpracuje wiele elementów, by zapewnić obronę przed różnego rodzaju zagrożeniami. Każdy z nich wnosi unikalny wkład w ocenę bezpieczeństwa wiadomości, a ich współdziałanie pozwala na podejmowanie trafnych decyzji. Kluczowe jest to, że system nie podejmuje decyzji na podstawie pojedynczego testu, lecz zbiera wyniki z wszystkich analiz i na ich podstawie podejmuje ostateczną decyzję.

4.3 Tworzenie konta w ShildMail

Proces rejestracji rozpoczyna się od wypełnienia formularza rejestracyjnego który wymaga wprowadzenia nazwy użytkownika, imienia i nazwiska oraz hasła. Po zatwierdzeniu formularza dane przesypane są na serwer, gdzie przeprowadzana jest walidacja, obejmująca sprawdzenie unikalności adresu e-mail. Po pomyślnej weryfikacji danych hasło użytkownika jest poddane procesowi haszowania za pomocą bezpiecznego algorytmu SHA-512 [17], który tworzy skrót kryptograficzny, tym samym rozwiązuje problem przechowywania hasła w formie jawniej. Następnie tworzony jest nowy rekord w bazie danych, który zawiera imię i nazwisko, adres e-mail, skrót SHA-512 oraz datę rejestracji.

4.4 Rejestracja urządzeń autoryzacyjnych

Po zarejestrowaniu nowego konta serwis automatycznie przekieruje użytkownika na stronę, na której należy dodać drugi czynnik uwierzytelniający. W tym kroku dodano ograniczenie czasowe - użytkownik ma jedną minutę na skonfigurowanie drugiego czynnika uwierzytelniającego. Po upływie czasu konto zostanie usunięte, a cały proces rejestracji trzeba będzie przejść od początku. Takie rozwiązanie ma na celu zapobieganie tworzeniu wielu nieaktywnych kont, które nigdy nie zostaną użyte, co mogłoby niepotrzebnie obciążać zasoby serwera oraz bazę danych. Mechanizm 2FA wykonano przy użyciu biblioteki simplewebauthn [13].

Cały proces sprowadza się do kroków:

- Inicjalizacja rejestracji urządzenia poprzez przycisk “Dodaj urządzenie”.
- Przeglądarka wysyła żądanie do serwera o wygenerowanie opcji rejestracji.
- Serwer generuje unikalne wyzwanie oraz inne niezbędne dane (np. informacje o użytkowniku, wymagania kryptograficzne) i zwraca je do przeglądarki jako obiekt PublicKeyCredentialCreationOptions.

Listing 4.1: Generowanie opcji rejestracji - PublicKeyCredentialCreationOptions

```
# Tworzenie opcji rejestracji klucza
registration_options = generate_registration_options(
    rp_id=WEBAUTH_RP_ID, # adamszreiber.pl
    rp_name=WEBAUTH_RP_NAME, # Shild Mail
    user_name=passkey_registration.email, # email uzytkownika
)
# Zapisanie challange do pamieci
passkey_registration.challange =
    registration_options.challenge.hex()
# Przekszta cenie obiektu PublicKeyCredentialCreationOptions do
# formatu JSON
reg_options_json =
    json.loads(options_to_json(registration_options))
# Informuje czy klucz moze byc wyszukiwany i uzywany w kontekscie
# logowania
reg_options_json["extensions"] = { "credProps": True }
# Wyslanie opcji do klienta
return JsonResponse(reg_options_json)
```

- Aplikacja kliencka po otrzymaniu opcji, wywołuje metodę startRegistration() z biblioteki `@simplewebauthn/browser`. Przeglądarka wyzwala natywne okno dialogowe w którym użytkownik zostaje poproszony o fizyczne potwierdzenie tożsamości, np. przez dotknięcie klucza bezpieczeństwa, odcisk palca lub inne metody biometryczne.
- Przeglądarka generuje odpowiednie dane uwierzytelniające (AuthenticatorAttestationResponse), które zawierają między innymi klucz publiczny, podpis kryptograficzny, wyzwanie
- Dane wygenerowane przez przeglądarkę są wysyłane na serwer w celu weryfikacji.

Listing 4.2: Tworzenie nowych poświadczzeń

```
//Pobranie parametrow rejestracji z serwera
const registrationOptionsJSON = await
    loadCredentialRequestOptions()
//Inicjalizacja procesu rejestracji
const attestationResp = await
    SimpleWebAuthnBrowser.startRegistration({ optionsJSON:
        registrationOptionsJSON });
//Przeslanie do serwera AuthenticatorAttestationResponse
await sendRegistartionAuthenticatorResponse(attestationResp)
```

- Serwer sprawdza poprawność podpisu, wyzwania oraz innych parametrów, aby upewnić się, że proces rejestracji był prawidłowy, zapisuje go w bazie danych, a następnie wysyła odpowiedź o pomyślnym przeprowadzeniu rejestracji.

Listing 4.3: Weryfikacja i zapisanie poświadczeń

```
# przetworzenie danych i utworzenie obiektu RegistrationCredential
credential = parse_registration_credential_json(body_json)
# walidacja i utworzenie obiektu VerifiedRegistration
verification = verify_registration_response(
    credential=credential,
    expected_challenge=bytes.fromhex(passkey_registration.challange),
    expected_rp_id=WEBAUTH_RP_ID, # adamszreiber.pl
    expected_origin=WEBAUTH_ORIGIN, # https://adamszreiber.pl
    require_user_verification=False, # flag UV, UP
)
# Stworzenie nowego obiektu posiadczenn FIDO
new_passkey = WebAuthnCredential.objects.create(
    username=passkey_registration.email,
    credential_id=verification.credential_id,
    credential_public_key=verification.credential_public_key,
    credential_type=verification.credential_type,
    aaguid=verification.aaguid,
    transports=body_json['response'].get("transports", list),
    is_discoverable=True,
    sign_count=verification.sign_count,
    authenticator_attachment=body_json['authenticatorAttachment'],
    public_key_algorithm=body_json['response']
        .get("publicKeyAlgorithm", 0),
)
# Zapisanie obiektu posiadczend bazy danych
new_passkey.save()
# Usuniecie z pamieci obiektu pomocniczego PasskeysRegistration
memory_storage.pop(uuid)
# Potwierdzenie dodania klucza
return JsonResponse({'result':True, 'redirect':'/login'})
```

Mechanizm logowania

Formularz logowania opracowany na potrzeby omawianego rozwiązania wymaga od użytkownika wprowadzenia adresu e-mail oraz hasła. Adres e-mail, jako unikatowy identyfikator, pozwala na jednoznaczne rozpoznanie konta, do którego użytkownik próbuje uzyskać dostęp.

Po przesłaniu formularza logowania przez użytkownika, serwer odbiera wprowadzone dane, a następnie wyszukuje w bazie danych użytkownika przypisanego do tego adresu mail. Jeśli taki użytkownik istnieje, Django tworzy skrót SHA512 z hasła, wprowadzonego w formularzu i porównuje z skrótem przechowywanym w bazie. Jeżeli adres e-mail oraz hasło są poprawne, użytkownik zostaje zalogowany, a Django tworzy sesję użytkownika, zapisując odpowiednie dane identyfikujące na serwerze oraz w odpowiedzi wysyła przeglądarce nagłówek zawierający cookies (ciasteczko) o treści sessionid=XYZ; expires=Thu, 30 Jan 2025 14:35:34 GMT; HttpOnly; Max-Age=1209600; Path=/; SameSite=Lax. Ciasteczko nosi nazwę **sessionid** i posiada wartość "XYZ", która stanowi ciąg identyfikujący sesję użytkownika. Dzięki ustawieniu atrybutu HttpOnly nie może być ono odczytane przez skrypty na stronie, co chroni przed przejęciem sesji. Dodatkowo, atrybut SameSite ogranicza jego przesyłanie w żądaniach pochodzących z innych witryn. Sesja pozwala na śledzenie stanu zalogowania, dzięki czemu użytkownik może korzystać z chronionych zasobów i funkcji aplikacji bez konieczności ponownego podawania danych logowania w trakcie trwania sesji. W przypadku niepowodzenia, użytkownik otrzymuje komunikat o błędnych danych logowania.

Mechanizm 2FA przy pomocy FIDO2

Po pomyślnym zalogowaniu, system wymaga dodatkowego potwierdzenia tożsamości użytkownika poprzez uwierzytelnienie za pomocą drugiego czynnika uwierzytelniającego, które zostało wcześniej zarejestrowane.

Proces ten można przedstawić w następujących krokach:

- Inicjalizacja uwierzytelniania poprzez przycisk "Uwierzytelnij", przeglądarka wysyła żądanu do serwera w celu wygenerowania opcji logowania.
- Serwer tworzy obiekt PublicKeyCredentialRequestOptions, który zawiera:
 - wyzwanie,
 - listę zarejestrowanych kluczy użytkownika (allowCredentials),
 - informacje o domenie aplikacji,
 - limit czasu na wykonanie operacji,

Listing 4.4: Generowanie opcji logowania - PublicKeyCredentialRequestOptions

```
# Pobranie zarejestrowanych kluczy dla uzytkownika z bazy danych
credentials =
    WebAuthnCredential.objects.filter(username=request.user.email)
# Wygenerowanie opcji autentykacji przy pomocy klucza
authentication_options = generate_authentication_options(
    rp_id=WEBAUTH_RP_ID, # adamszreiber.pl
    # Dla znalezionych kluczy tworzony jest
    PublicKeyCredentialDescriptor
allow_credentials=[
    PublicKeyCredentialDescriptor(id=credential.credential_id)
    for credential in credentials
],
)
# Do sesji uzytkownika zostaje zapisany wygenerowany challenge
request.session['challenge'] =
    authentication_options.challenge.hex()
# przekształcenie typu PublicKeyCredentialRequestOptions na JSON
response_json = json.loads(options_to_json(authentication_options))
return JsonResponse(response_json) # Wyslanie danych do
przegladarki
```

- Przeglądarka po otrzymaniu opcji, wywołuje metodę startAuthentication() z biblioteki @simplewebauthn/browser. Zostaje otwarte natywne okno dialogowe, umożliwiające użytkownikowi interakcję z urządzeniem uwierzytelniającym.

Listing 4.5: Uwierzytelnianie za pomocą klucza

```
# Pobranie opcji autentyfikujących z serwera
authenticationOptionsJSON = await loadCredentialRequestOptions()
# Inicjalizacja autoautentykacji
const assertionResp = await
    SimpleWebAuthnBrowser.startAuthentication({
        optionsJSON: authenticationOptionsJSON
});
# Wyslanie response do serwera
sendAssertion(assertionResp)
```

- Użytkownik zostaje poproszony o fizyczne potwierdzenie tożsamości, np. przez dotknięcie klucza bezpieczeństwa, użycie odcisku palca, lub inną metodę biometryczną.

- Po pomyślnym potwierdzeniu, klucz generuje dane uwierzytelniające (AuthenticatorAssertionResponse), które zawierają:

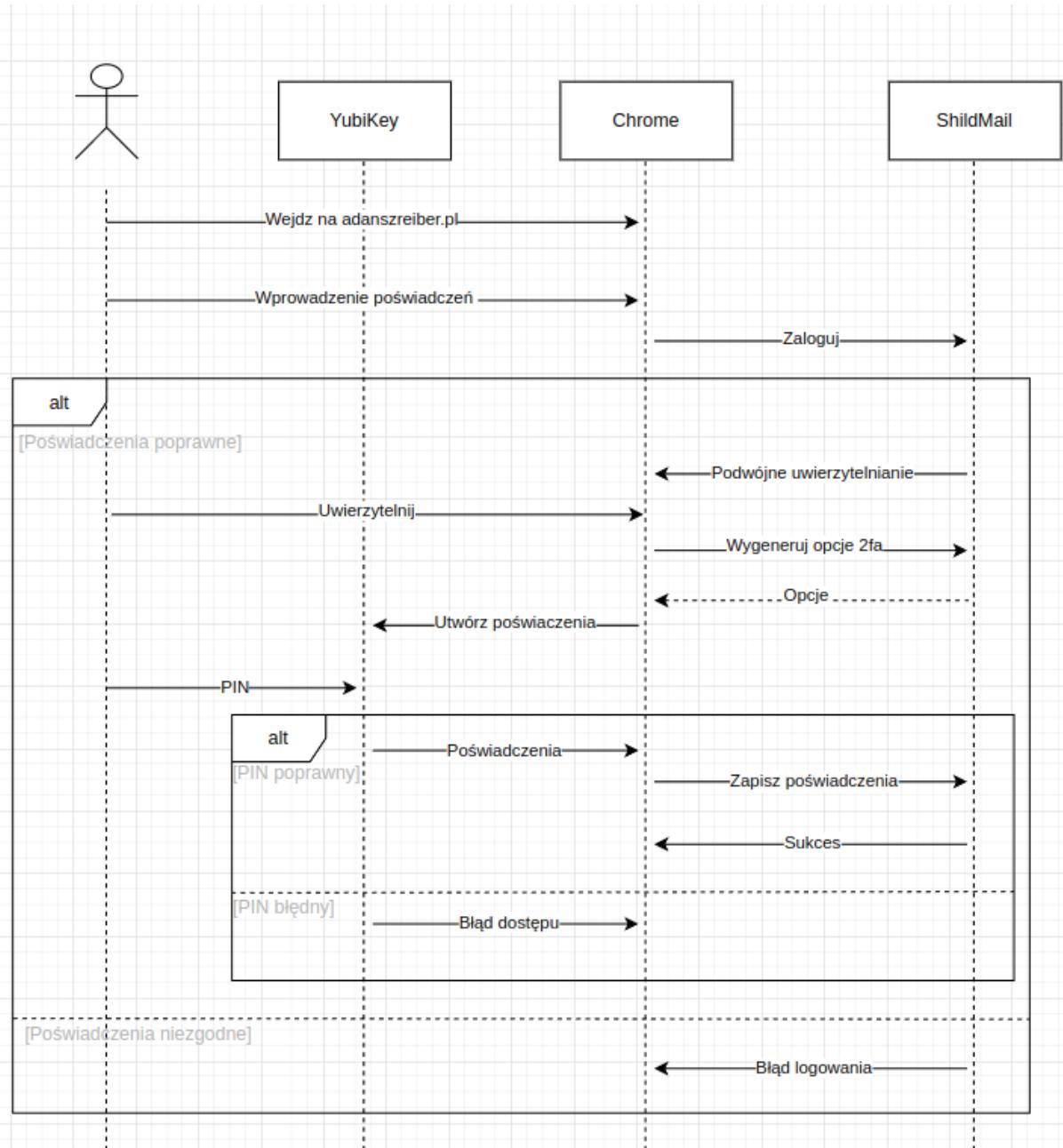
- podpisane wyzwanie,
- identyfikator klucza,

Następnie dane przesyłane do serwera w celu weryfikacji.

- Serwer otrzymuje odpowiedź, sprawdza, czy podpisane wyzwanie jest zgodne z wcześniejszym wygenerowanym, weryfikuje podpis kryptograficzny za pomocą klucza publicznego. Jeśli weryfikacja przebiegła pomyślnie, serwer potwierdza logowanie użytkownika i umożliwia dostęp do funkcjonalności serwisu.

Listing 4.6: Weryfikacja uwierzytelniania

```
# Sprawdzenie poprawnosci formatu wyslanych danych i utworzenie
# AuthenticationCredential
credential = parse_authentication_credential(body_json)
# Pobranie klucza ktory uzytkownik przedstawił w procesie
# autentyfikacji
db_credential = WebAuthnCredential.objects.filter(
    username=request.user.email,
    credential_id=credential.raw_id
).first()
# Odczyt wartosci 'challenge' z sesji.
challenge = request.session['challenge']
# Sprawdzenie poprawnosci odpowiedzi uwierzytelniającej
verification = verify_authentication_response(
    credential=credential,
    expected_challenge=bytes.fromhex(challenge),
    expected_rp_id=WEBAUTH_RP_ID, # adamszreiber.pl
    expected_origin=WEBAUTH_ORIGIN, # https://adamszreiber.pl
    credential_public_key=db_credential.credential_public_key,
    credential_current_sign_count=db_credential.sign_count,
)
# Aktualizacja wartosci licznika logowan w bazie danych
WebAuthnCredential.objects.update_sign_count(
    credential_id=credential.raw_id,
    count=verification.new_sign_count
)
# Zapisanie w sesji id klucza
request.session['credential_id'] = credential.id
return JsonResponse({'result':True, 'redirect':'/'})
```



Rysunek 4.3: Proces autentykacji przedstawiony na diagramie sekwencji

Rozdział 5

Analiza informacji uwierzytelniających w kanale komunikacyjnym.

Do przeprowadzenia analizy komunikacji oraz przepływu informacji uwierzytelniających wykorzystano program **Wireshark**, który pozwala na przechwytywanie i szczegółowe badanie ruchu sieciowego. Na rysunku 5.1, Wireshark przechwycił i dekodował komunikaty wymieniane podczas całego procesu autoryzacji – począwszy od wejścia na stronę rejestracji, aż do po-prawnego zalogowania się nowym kontem i przejścia na stronę główną. Dzięki analizie ruchu sieciowego możliwe będzie prześledzenie każdego etapu procesu uwierzytelniania i porównanie go z wymogami standardu.

Aby przeprowadzić techniczną analizę diagramu przepływu pakietów w Wiresharku w modelu **request-response**, należy dokładnie prześledzić komunikację między klientem a serwerem. Każda interakcja składa się z dwóch pakietów: zapytania (request) oraz odpowiedzi (response). Na przedstawionym rysunku 5.1 klientem jest **adam.pc**, a serwerem **adamszreiber.pl**.



Rysunek 5.1: Wireshark - komunikacja w procesie rejestracji i logowania

Proces rejestracji

Tabela 5.1: Konfrontacja wdrożonej rejestracji z standaryzacją

Kroki	ShildMail	Dokumentacja
1-2	Pobranie strony /register	Wejście na stronę
3-4	Wysłanie formularza rejestracji	Użytkownik może być zalogowany bądź może być w trakcie procesu tworzenia nowego konta
5-6	Pobranie strony /register-credentials	-
7-8	Pobranie struktury PublicKeyCredentialsCreationOptions	Pobranie challenge, oraz metadata w postaci struktury PublicKeyCredentialsCreationOptions
9-10	Wysłanie utworzonych poświadczeń w formie struktury AuthenticatorAttestationResponse	Wysłanie klucza publicznego na serwer wraz z dodatkowymi informacjami w formie struktury AuthenticatorAttestationResponse

Proces logowania

Tabela 5.2: Konfrontacja wdrożonego uwierzytelniania z standaryzacją

Kroki	ShildMail	Dokumentacja
11-12	Pobranie strony /login	Przejście na stronę
13-14	Wysłanie formularza logowania	Wprowadzenie danych które umożliwią wybór poświadczeń należących do użytkownika
15-16	Pobranie strony /2fa	-
17-18	Pobranie struktury PublicKeyCredentialsRequestOptios	Inicjalizacja uwierzytelniania - pobranie z serwera przyporządkowanych użytkownikowi kluczy i wartości challenge - struktura PublicKeyCredentialsRequestOptios
19-20	Wysłanie poświadczeń w formie struktury PublicKeyCredentials	Wysłanie poświadczeń do serwera
21-22	Pobranie strony /	Dowolne poinformowanie użytkownika o poprawnym uwierzytelnieniu

5.1 Weryfikacja implementacji

Autor stworzył domenę phishingową o nazwie adanszreiber.pl, która różni się od prawdziwej domeny adamszreiber.pl jedną literą – nastąpiła zmiana "m" na "n". Ta subtelna zmiana, ma na celu zasymulowanie typowego ataku phishingowego, w którym przestępcy wykorzystują podobnie wyglądające nazwy domen.

Weryfikacja procesu rejestracji opiera się na założeniu, iż użytkownik wykonuje proces tworzenia konta **przy użyciu nieprawidłowej domeny**. W standardowych warunkach, przy prawidłowej konfiguracji frameworka Django i odpowiednim ustawieniu parametru ALLOWED_HOSTS – który określa listę zaufanych domen – taka sytuacja nie byłaby możliwa. Mechanizm ten zablokowałby zapytanie już na etapie weryfikacji domen, uniemożliwiając dalsze przetwarzanie.

DisallowedHost at /

Invalid HTTP_HOST header: 'adanszreiber.pl'. You may need to add 'adanszreiber.pl' to ALLOWED_HOSTS.

Request Method: GET
Request URL: http://adanszreiber.pl/
Django Version: 5.1.4
Exception Type: DisallowedHost
Exception Value: Invalid HTTP_HOST header: 'adanszreiber.pl'. You may need to add 'adanszreiber.pl' to ALLOWED_HOSTS.

Rysunek 5.2: Błąd: połączenie z domeną nie ujętą w ALLOWED_HOSTS

Na potrzeby przeprowadzenia szczegółowej weryfikacji mechanizmu rejestracji, rozszerzono możliwości komunikacyjne systemu, umożliwiając obsługę żądań pochodzących z dowolnych domen.

W kontekście omawianego zagadnienia należy rozważyć następujące scenariusze:

1. Użytkownik przystępuje do procesu rejestracji, następnie zostaje przekierowany na stronę rejestracji klucza uwierzytelniającego. W momencie inicjalizacji tego etapu serwis wywołuje błąd **SecurityError**, wskazując, że użytkownik znajduje się na domenie niezgodnej z tą określoną w parametrach przekazanych w obiekcie PublicKeyCredentialCreationOptions.

The screenshot shows a browser's developer tools console. The user has typed 'window.location.host' and received the response 'adanszreiber.pl'. A red error box highlights an 'Uncaught (in promise)' error from 'index.umd.min.js:2'. The error message is: 'SecurityError: The RP ID "adamszreiber.pl" is invalid for this domain'. It then traces back through the code: 'at error (index.umd.min.js:2:4343)', 'at e.startAuthentication (index.umd.min.js:2:4671)', and 'at async device_auth (2fa:66:29)'. Below this, it says 'Caused by: SecurityError: The relying party ID is not a registrable domain suffix of, nor equal to the current domain. Subsequently, an attempt to fetch the .well-known/webauthn resource of the claimed RP ID failed.'

Rysunek 5.3: SecurityError - nieprawidłowa domena

2. Użytkownik nawiązuje interakcję z serwisem dostępnym pod adresem adanszreiber.pl. Inicjuje procedurę rejestracji w systemie dwuskładnikowego uwierzytelniania poprzez wysłanie odpowiedniego żądania do serwera. W odpowiedzi serwer generuje i przesyła zestaw parametrów służących do zainicjowania procesu rejestracji, jednakże – na skutek nieautoryzowanej ingerencji – wewnętrz tych parametrów modyfikacji ulega wartość **rp.id**, przypisana zostaje domena adanszreiber.pl. Urządzenie uwierzytelniające sprawdza otrzymane parametry i interpretuje jako prawidłowe, co skutkuje wygenerowaniem pary kluczy kryptograficznych oraz podpisaniem wyzwania za pomocą klucza prywatnego. Kolejnym etapem jest odesłanie do serwera podpisanej odpowiedzi, wraz z kluczem publicznym. Serwer, w procesie weryfikacji, najpierw rozszyfrowuje dane przy użyciu otrzymanego klucza publicznego, następnie dokonuje porównania domeny, dla której klucz został pierwotnie utworzony, z tą, którą skonfigurowano w opcjach serwerowych. W przedstawionej sytuacji, stwierdzona zostaje rozbieżność między powyższymi wartościami, występuje

błąd `InvalidRegistrationResponse`, kończący proces rejestracji urządzenia uwierzytelniającego.



Rysunek 5.4: Błąd serwerowy – rejestrowany klucz nie jest przeznaczony dla domeny *adamszreiber.pl*.

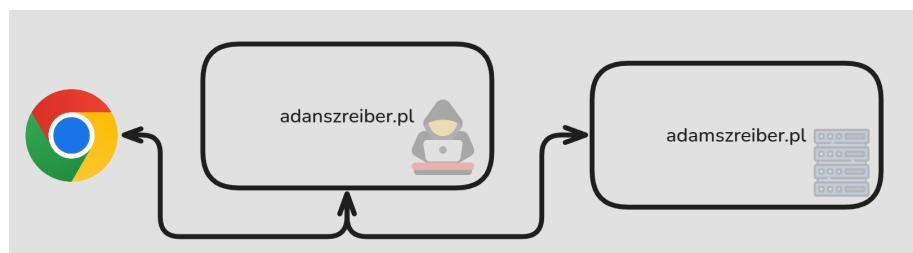
Aby zweryfikować poprawność implementacji mechanizmu uwierzytelniania, na wstępie należy przyjąć kilka kluczowych założeń.

1. Użytkownik zarejestrował się w systemie za pośrednictwem domeny *adamszreiber.pl*.
2. Użytkownik przeszedł pełen proces rejestracji i dodał drugi czynnik uwierzytelniający.

Takie założenia pozwalają na dokładne przetestowanie oraz ocenę skuteczności wdrożonego rozwiązania przed nieautoryzowanym dostępem.

Rozważone zostaną 2 przypadki:

1. Użytkownik trafia na witrynę *adanszreiber.pl*. Wprowadza swoje dane uwierzytelniające, następnie przystępuje do procedury dwuskładnikowego uwierzytelniania. Procedura zostaje przerwana, a system generuje komunikat błędu `SecurityError 3.9`. Użytkownik zostaje poinformowany, iż wystąpiła rozbieżność między domeną, na której się znajduje (*adanszreiber.pl*), a domeną przekazaną przez serwer w konfiguracji opcji uwierzytelniania.



Rysunek 5.5: Atak typu sniffing 2.5.1

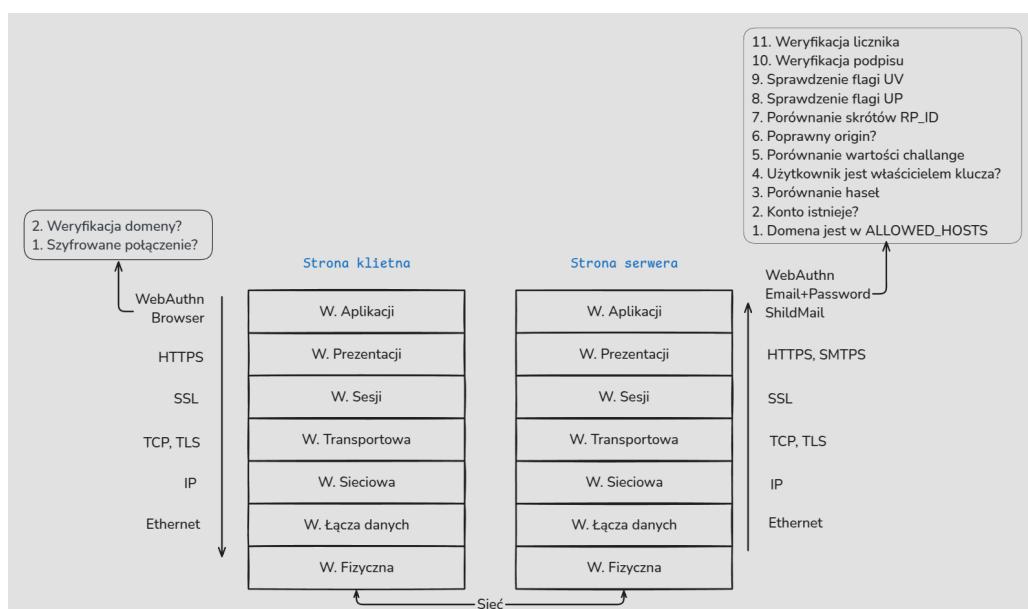
2. Użytkownik odwiedza stronę działającą pod adresem adanszreiber.pl i loguje się, wykorzystując swoje dane uwierzytelniające – login oraz hasło. Po udanym logowaniu następuje przejście do etapu uwierzytelniania dwuskładnikowego. W krytycznym momencie procesu, serwis pośredniczący, działający w roli atakującego, dokonuje modyfikacji przekazywanego obiektu opcji, ustawiając parametr **rp.id** na wartość odpowiadającą phisingowej domenie adanszreiber.pl. W wyniku tej manipulacji, mimo iż użytkownik otrzymuje zestaw parametrów, które pozornie są prawidłowe, urządzenie uwierzytelniające nie odnajduje klucza odpowiadającego phisingowej domenie 5.9.



Rysunek 5.6: Atak typu Man in the Middle 2.5.1

5.2 Analiza warstwowa zabezpieczeń

Bezpieczeństwo systemów informatycznych można rozpatrywać w oparciu o siedmiowarstwowy model ISO/OSI, który umożliwia logiczne rozdzielenie różnych aspektów transmisji danych. Każda warstwa modelu pełni inną funkcję w procesie przesyłania danych, co pozwala na wielopoziomowe planowanie i wdrażanie odpowiednich zabezpieczeń.



Rysunek 5.7: Schemat wielowarstwowych zabezpieczeń aplikacji ShildMail w modelu ISO/OSI

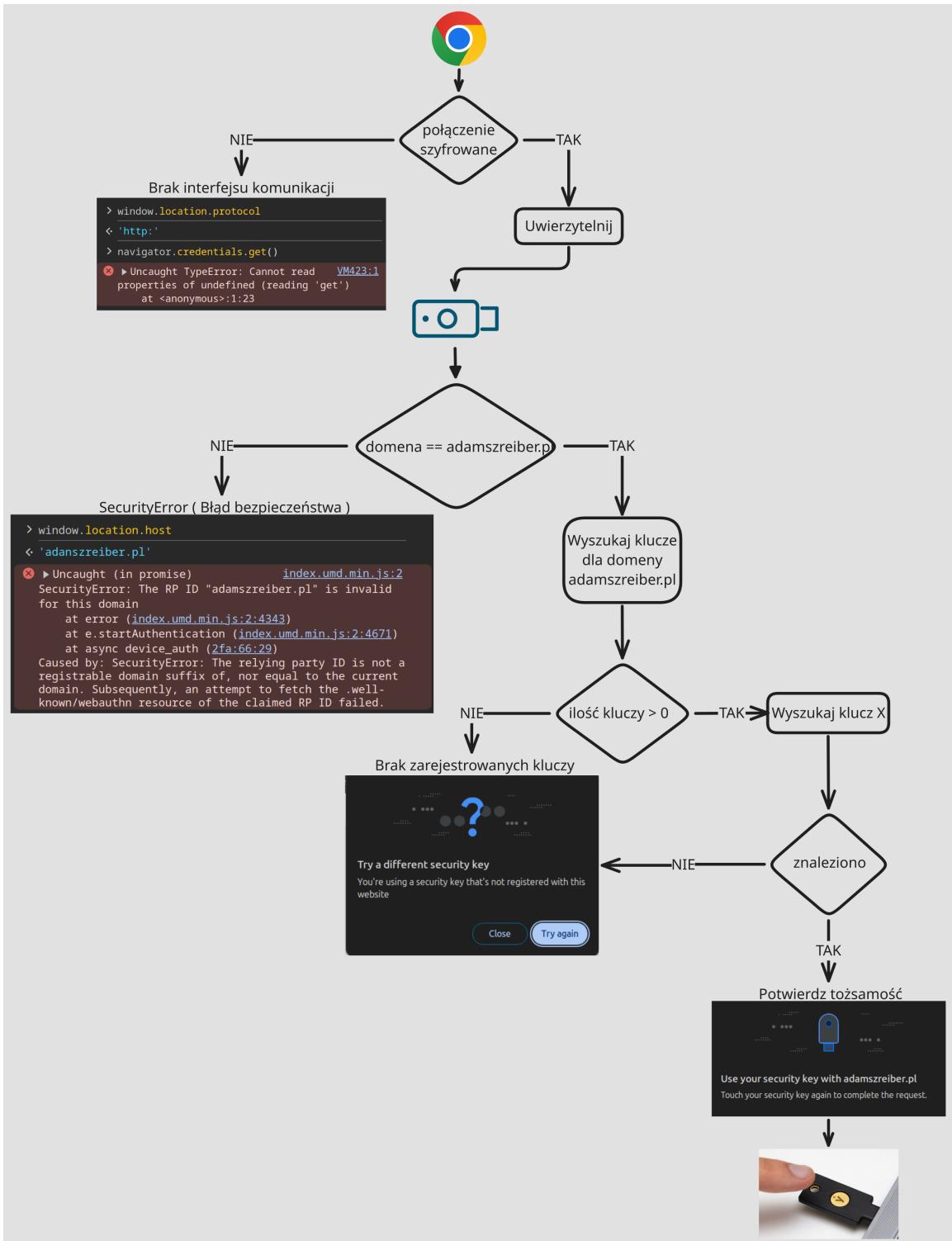
W modelu ISO/OSI dane inicjowane przez przeglądarkę w warstwie aplikacji (na urządzeniu końcowym użytkownika) przechodzą stopniowo przez niższe warstwy – w celu ich odpowiedniego przygotowania i zaadresowania do przesłania przez infrastrukturę sieciową. Następnie, w środowisku serwerowym, odebrane dane „pokonują” kolejne poziomy od dołu, aż w końcu trafiają do warstwy aplikacji, gdzie zostaną obsłużone przez procesy i usługi oferowane przez serwer.

Dostęp do urządzenia autoryzującego jest możliwy wyłącznie wtedy, gdy kanał komunikacyjny jest zabezpieczony szyfrowaniem, co pokazano na rysunku 5.9. W takim przypadku przeglądarka nie tylko ustanawia połączenie zgodne z protokołem TLS, ale również przeprowadza weryfikację certyfikatu SSL serwera z domeną aplikacji, co gwarantuje, że połączenie jest autentyczne i nie zostało poddane manipulacjom.



Rysunek 5.8: Certyfikat SSL - widok z poziomu przeglądarki

Przed rozpoczęciem procesu uwierzytelniania, urządzenie dokonuje weryfikacji domeny, na której aktualnie znajduje się użytkownik, porównując ją z domeną odczytaną z przekazanej struktury danych – PublicKeyCredentialRequestOptions. Schemat przebiegu weryfikacji danych po stronie przeglądarki został przedstawiony na diagramie 5.9, który ilustruje kolejne etapy tego procesu – od potwierdzenia, że kanał komunikacyjny jest szyfrowany, przez weryfikację zgodności domeny, aż po ostateczną autoryzację żądania.



Rysunek 5.9: FIDO2 - Mechanizm walidacji po stronie klienta

Po stronie serwerowej rozpoczynając analizę od dołu stosu - pierwsza warstwa która posiada zabezpieczenia to warstwa transportowa. Na tej warstwie umieszony został protokół TLS. Stanowi rozwinięcie wcześniejszego rozwiązania znanego jako SSL (Secure Sockets Lay-

er). Tworzy bezpieczny kanał komunikacyjny między klientem a serwerem. Szyfruje przesyłane dane, zapewnia integralność komunikacji i pozwala na uwierzytelnienie serwera. Efektem działania TLS jest ochrona przed przechwyceniem lub modyfikacją przesyłanych informacji, a także możliwość weryfikacji autentyczności serwera, z którym nawiązywane jest połączenie.

Source	Destination	Protocol	Info
adam.pc	adamszreiber.pl	TLSv1.3 Client Hello (SNI=adamszreiber.pl)	
adamszrei... adam.pc		TLSv1.3 Server Hello, Change Cipher Spec, Encrypted Extensions, Finished	
adam.pc	adamszreiber.pl	TLSv1.3 Change Cipher Spec, Finished	
adamszrei... adam.pc		TLSv1.3 New Session Ticket	

Rysunek 5.10: Proces ustanawiania szyfrowanego połączenia TLS w Wireshark

W warstwie prezentacji znajdują się klasyczne protokoły, takie jak HTTP czy SMTP, które nie oferują domyślnie ochrony przed podsłuchem i modyfikacjami danych, nie jest wystarczające z punktu widzenia bezpieczeństwa. Dlatego w procesie tworzenia serwisu ShildMail użyto ich bezpieczniejsze odpowiedniki – HTTPS i SMTPS – które wzbogacają komunikację o warstwę szyfrowanego połączenia. Autoryzacja oparta na standardzie FIDO2 jest dostępna wyłącznie przy użyciu szyfrowanych połączeń, które wykorzystują certyfikaty SSL.

Ostatni komponent zabezpieczający znajduje się w warstwie aplikacji. Na tej warstwie znajduje się wiele mechanizmów weryfikacji które przechodzą w określonej kolejności (z przedstawionej na grafice 5.7):

1. Domena jest w ALLOWED_HOSTS - na samym początku system weryfkuje, czy domena, z której pochodzi żądanie, znajduje się na liście dozwolonych hostów.
2. Istnieje konto - sprawdzana jest obecność konta powiązanego z podanym adresem e-mail w bazie danych.
3. Porównanie haseł - w tym etapie następuje porównanie skrótu hasła przechowywanego w systemie z wartością dostarczoną przez użytkownika.
4. Użytkownik jest właścicielem klucza? - krok polega na weryfikacji, czy klucz użyty w procesie uwierzytelniania jest przypisany do tego użytkownika.
5. Porównywanie wartości wyzwania - system porównuje wartość wyzwania przeslaną przez serwer z tą, która została zwrócona przez urządzenie autoryzujące, przy pomocy klucza publicznego przekazanego w procesie rejestracji.
6. Poprawny origin - Weryfikowana jest zgodność wartości origin, czyli źródła żądania, z RP origin zapisanym w ShildMail. Dzięki temu możliwe jest potwierdzenie, że żądanie pochodzi z autoryzowanego interfejsu.

7. Porównanie skrótów RP_ID - system oblicza skrót identyfikatora RP (RP_ID) i porównuje go z wartością przekazaną z urządzania autoryzującego. Krok ma na celu potwierdzenie, że żądanie jest skierowane do właściwego serwera, zgodnego z danymi zapisanymi w konfiguracji uwierzytelniania.
8. Sprawdzenie flagi UP - flaga *User Present*, informuje, czy użytkownik był fizycznie obecny podczas procesu uwierzytelniania.
9. Sprawdzenie flagi UV - faga *User Verified*, wskazuje, czy użytkownik przeszedł dodatkową weryfikację (np. PIN), co dodatkowo potwierdza jego tożsamość.
10. Weryfikacja podpisu - w tym etapie następuje kryptograficzna weryfikacja podpisu generowanego przez urządzenie autoryzujące, przy użyciu wcześniej zarejestrowanego klucza publicznego. Podpis ten obejmuje dane uwierzytelniające i stanowi gwarancję integralności oraz autentyczności przesyłanych informacji.
11. Weryfikacja licznika - system sprawdza licznik (signature counter) przypisany do urządzenia autoryzującego, co pozwala na wykrycie ewentualnych prób ponownego użycia starych danych uwierzytelniających lub klonowania urządzenia.

Rozdział 6

Wnioski

Przeprowadzona analiza oraz praktyczne badania nad wdrożeniem mechanizmów FIDO2 w aplikacji poczty internetowej wykazały, że wykorzystanie standardu realnie wpływa na wzrost poziomu bezpieczeństwa. Dzięki wdrożeniu mechanizmów weryfikacji domeny możliwość przeprowadzenia skutecznych ataków phishingowych zostaje całkowicie wyeliminowana. Nawet w sytuacji, gdy atakujący pozyska podstawowe dane logowania – nazwę użytkownika i hasło – nie uzyska dostępu do konta, gdyż procedura uwierzytelniania wymaga drugiego czynnika z grupy "coś co masz".

Weryfikacja poszczególnych etapów – zarówno podczas rejestracji, jak i przy samej autentyczacji – w odniesieniu do przeanalizowanej dokumentacji standardu FIDO2, utwierdziła autora w przekonaniu, że proces implementacji został przeprowadzony prawidłowo. Testy przypadków brzegowych wykazały, że w sytuacji wystąpienia niezgodności, system konsekwentnie generował błąd w oczekiwany momencie, co potwierdziło poprawność implementacji.

Zaobserwowano, iż niektórzy użytkownicy mogą wymagać dodatkowego szkolenia lub wsparcia, zwłaszcza w przypadku uwierzytelniania za pośrednictwem Bluetooth. Proces ten wymaga bowiem większej liczby kroków i może wydawać się mniej intuicyjny dla osób nieobeznanych z technologiami bezprzewodowymi.

Stwierdzono, że wdrożenie autoryzacji opartej na specyfikacji FIDO2 jest zadaniem czasochłonnym i kosztownym. Wymaga szerokiej wiedzy z zakresu specyfikacji oraz protokołów. Jednakże, biorąc pod uwagę znaczny wzrost poziomu ochrony zapewniany przez te mechanizmy, rozważenie ich implementacji może przynieść wymierne korzyści, zwłaszcza tam, gdzie phishing stanowi istotne zagrożenie dla poufności i integralności systemów.

6.0.1 Napotkane trudności

Podczas tworzenia serwisu ShildMail autor napotkał szereg problemów. Już na samym początku kluczowym wyzwaniem okazało się zrozumienie i uwzględnienie różnicy pomiędzy implementacją a integracją, co znaczco wpłynęło na cały proces projektowania i budowy aplikacji. Implementacja czyli tworzenie od podstaw funkcjonalności systemu, zgodnie z określonymi wymaganiami i specyfikacjami, wymaga pisania kodu, budowy nowych modułów oraz projektowania mechanizmów. Z kolei integracja polega na łączeniu istniejących systemów, komponentów lub usług w sposób, który umożliwia im współpracę. Temat pracy skupiał się na implementacji, co wymusiło na autorze przyswojenie obszernej wiedzy technicznej. Zdobyte kompetencje zostały następnie wykorzystane w praktycznej części opracowania przy implementacji serwera ShildMail, integrującego wszystkie kluczowe komponenty w spójną i funkcjonalną całość.

Pierwszym krokiem w implementacji było znalezienie odpowiedniego hostingu serwera, na którym miał działać system ShildMail. Jak się później okazało, wybór hostingu stał się poważnym wyzwaniem. Dostawcy usług hostingowych, w trosce o bezpieczeństwo internetowe i zapobieganie nadużyciom związanym ze spamem, często blokują porty odpowiedzialne za komunikację SMTP i SMTPS (port 25, 465). To uniemożliwia wysyłanie wiadomości e-mail z poziomu serwera. Rozważano skorzystanie z platformy DigitalOcean, która jest popularnym rozwiązaniem wśród programistów i oferuje elastyczne środowisko do budowy aplikacji. Niestety, okazało się, że platforma ta domyślnie blokuje porty, co stanowiło przeszkodę w realizacji projektu. Autor skontaktował się z działem wsparcia DigitalOcean, wyjaśniając, że projekt jest częścią badań do pracy dyplomowej i zapewniając, że nie ma złych intencji. Mimo starań i argumentów, prośba o odblokowanie portów została odrzucona. Następnie podjęto próbę z platformą Amazon AWS, która jak się później okazało stosuje podobną politykę bezpieczeństwa i blokuje porty SMTP. Kontakt z działem wsparcia oraz prośba o odblokowanie portów również nie przyniosły żadnych rezultatów, co zmusiło autora do poszukiwania innych rozwiązań. Ostatecznie zdecydowano się na skorzystanie z usług mniej rozpowszechnionego dostawcy jakim jest Hetzner.

Implementacja platformy pocztowej okazała się zadaniem niezwykle skomplikowanym, ze względu na złożoną strukturę systemu i konieczność integracji wielu różnorodnych komponentów. Aby stworzyć w pełni funkcjonalny i bezpieczny serwis pocztowy, niezbędne było zastosowanie takich narzędzi jak Postfix, Dovecot, Amavis, PostgreSQL, ClamAV, Policy-SPF oraz SpamAssassin. Każdy z tych elementów pełnił istotną rolę w całym ekosystemie i wymagał skomplikowanej konfiguracji, tak aby wszystkie działały jako jedna, spójna całość. Serwis musiał zostać odpowiednio skonfigurowany pod kątem rekordów DNS, które są kluczowe, aby

wiadomości wysyłane nie były oznaczane jako podejrzane lub spam przez inne serwery pocztowe. Następnie, moduły filtrujące pocztę po stronie serwera, musiały zostać zintegrowane z Postfix, aby skanować wszystkie przychodzące i wychodzące wiadomości pod kątem potencjalnych zagrożeń. Ostatecznie autorowi udało się zbudować system, który spełniał wszystkie stawiane mu wymagania.

Choć specyfikacja FIDO2 jest złożona i zawiera w sobie wiele protokołów, implementacja przebiegła sprawnie i efektywnie. Autor zawdzięcza to przede wszystkim dokładnej dokumentacji technicznej oraz licznie dostępnych przykładom wdrożeń. Istotnym czynnikiem ułatwiającym proces było również otwartoźródłowe podejście do technologii FIDO2, które umożliwiło szczegółową analizę kodu bibliotek implementujących ten standard. Dzięki możliwości wglądu w kod źródłowy autor zyskał głębsze zrozumienie działania protokołu. Pozwoliło to na skuteczną integrację FIDO2 z systemem ShildMail oraz rozwiązanie pojawiających się w trakcie implementacji problemów. Otwartość i przejrzystość ekosystemu FIDO2 znacząco przyczyniły się do osiągnięcia sukcesu w tym kluczowym etapie projektu.

6.0.2 Możliwości rozwoju

System ShildMail jako środowisko badawcze, posiada wiele możliwości rozwoju, w obszarze optymalizacji procesów rejestracji i logowania. Jednym z potencjalnych ulepszeń mogłoby być przeprowadzenie rejestracji i logowania w obszarze jednej witryny internetowej. Dzięki zastosowaniu takiego podejścia możliwe byłoby wyeliminowanie konieczności przejść pomiędzy stronami. W procesie rejestracji, po pomyślnym wprowadzeniu i przesłaniu danych do serwera, system mógłby natychmiast uruchomić natywny dialog przeglądarki służący do rejestracji klucza bezpieczeństwa w ramach mechanizmu 2FA. Takie rozwiązanie zapewniłoby bardziej płynny i intuicyjny przebieg procesu, umożliwiając użytkownikowi skonfigurowanie swoich kluczy zabezpieczających bezpośrednio po założeniu konta. Podobny schemat mógłby zostać zastosowany podczas logowania – po wprowadzeniu poświadczeń, natywny dialog przeglądarki byłby automatycznie wywoływany w celu weryfikacji tożsamości za pomocą klucza FIDO2.

Kolejnym krokiem w kierunku podniesienia poziomu bezpieczeństwa systemu ShildMail mógłby być integracja narzędzia Fail2Ban, które stanowi skuteczne rozwiązanie w ochronie przed różnymi formami ataków, takimi jak ataki słownikowe czy nieautoryzowane próby dostępu. Fail2Ban monitoruje logi serwera, identyfikując podejrzane zachowania, takie jak wielokrotne nieudane próby logowania, a następnie automatycznie blokuje adresy IP, z których pochodzą te próby.

Opracowany system koncentruje się przede wszystkim na zapewnieniu bezpiecznego dostępu do systemu i wdrożeniu zaawansowanych mechanizmów ochrony. Obszarem o znacznym

potencjalnie rozwojowym pozostaje interfejs użytkowników. W obecnym stanie brakuje takich funkcjonalności jak możliwość odświeżania skrzynki pocztowej w czasie rzeczywistym, obsługa załączników, tworzenie dodatkowych katalogów czy zarządzanie wiadomościami. Wdrożenie tych usprawnień niewątpliwie podniosłoby użyteczność, komfort i efektywność korzystania z systemu przez użytkowników.

6.0.3 Zdefiniowanie czynności do implementacji

Przed rozpoczęciem implementacji tak zaawansowanego mechanizmu uwierzytelniania, jakim jest FIDO2, niezbędne jest przeprowadzenie szczegółowej analizy systemowej w celu określenia, czy istnieje rzeczywista konieczność wdrożenia rozwiązania charakteryzującego się wysokim poziomem bezpieczeństwa. Warto rozważyć, czy alternatywne, a zarazem tańsze metody uwierzytelniania dwuskładnikowego takie jak o kody jednorazowe nie spełniłyby określonych wymagań bezpieczeństwa.

W kolejnym etapie należy przeprowadzić szczegółową analizę systemu pod kątem możliwości rozbudowy mechanizmów rejestracji i uwierzytelniania użytkowników, a także zaplanować zakres testów. W ramach analizy warto rozważyć również rezygnację z dotychczasowego systemu uwierzytelniania na rzecz rozwiązania opartego na passwordless authentication, które stanowi integralny element specyfikacji FIDO2.

Nieodzownym krokiem będzie analiza dostępnych bibliotek dla wybranego języka programowania, co znacząco przyspieszy i ułatwi proces implementacji. Wykorzystanie istniejących, sprawdzonych rozwiązań nie tylko zoptymalizuje czasochłonność procesu, ale również zmniejszy ryzyko wystąpienia błędów, które mogłyby pojawić się w przypadku próby samodzielnnej implementacji standardu FIDO2 od podstaw.

Przed rozpoczęciem implementacji niezbędnym etapem jest odpowiednie przygotowanie środowiska wdrożeniowego, które zapewni obsługę szyfrowanych połączeń HTTPS. Przeglądarki internetowe wspierające WebAuthn domyślnie uniemożliwiają wykonywanie operacji związanych z FIDO2 w przypadku działania aplikacji w środowisku niezabezpieczonym, takim jak protokół HTTP. W związku z tym, wdrożenie certyfikatu SSL/TLS oraz odpowiednia konfiguracja serwera w celu obsługi połączeń HTTPS stanowią warunek konieczny.

Implementacja standardu FIDO2 wymaga tworzenia kodu zarówno po stronie klienta, jak i serwera. Po stronie serwera niezbędne jest zapewnienie funkcjonalności związanych z obsługą kluczy kryptograficznych, przechowywaniem poświadczeń użytkownika oraz generowaniem wyzwań, które są kluczowe w procesie uwierzytelniania. Z kolei po stronie klienta należy zapewnić obsługę kluczy poprzez wykorzystanie interfejsu CTAP, zaimplementowanego w przeglądarkach internetowych.

Warto zainwestować w szeroki przekrój urządzeń autentyfikujących, pochodzących od różnych dostawców i oferujących odmienne interfejsy komunikacyjne, które są kluczowe dla prawidłowej implementacji systemu. Przeprowadzenie kompleksowej weryfikacji przy użyciu szerokiej gamy urządzeń pozwoli na identyfikację ewentualnych luk w zabezpieczeniach oraz identyfikację ewentualnych niezgodności.

Bibliografia

- [1] N.A Ahmad N. Maarop N.N.A Sjarif H. Abas A. Maetouq, S.M. Daud. Comparison of hash function algorithms against attacks: A review. *International Journal of Advanced Computer Science and Applications*, 9, 2018. https://thesai.org/Downloads/Volume9No8/Paper_13-Comparison_of_Hash_Function_Algorithms.pdf [Data uzyskania dostępu: 09.02.2024r.]. [str. 24]
- [2] R. Anderson. *A Guide to Building Dependable Distributed Systems*. WILEY, 2020. [str. 17]
- [3] Dr. Ganavi M Bhavana R M. Fast identity online 2 :authentication technique. *International Journal of Scientific Research in Engineering and Management (IJSREM)*, 2024. [str. 40, 42, 74]
- [4] R. Boyd. *Getting Started with OAuth 2.0*. O'REILLY, 2012. [str. 16]
- [5] A. Pinto S. Widup C.D. Hylander, P. Langlois. 2024 data breach investigations report. <https://www.verizon.com/business/resources/T50c/reports/2024-dbir-data-breach-investigations-report.pdf> [Data uzyskania dostępu: 09.02.2024r.]. [str. 22]
- [6] Cybersecurity and Infrastructure Security Agency. Phishing activity trends report. <https://www.cisa.gov/secure-our-world/require-strong-passwords> [Data uzyskania dostępu: 09.02.2024r.]. [str. 22]
- [7] Ravindra Das. *The Zero Trust Framework and Privileged Access Management*. CRC Press, 2024. [str. 19]
- [8] fido ALLIANCE. Client to authenticator protocol (ctap). <https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html> [Data uzyskania dostępu: 09.02.2024r.]. [str. 29]

- [9] Anti-Phising Working Group. Phishing activity trends report. https://docs.apwg.org/reports/apwg_trends_report_q3_2024.pdf [Data uzyskania dostępu: 09.02.2024r.]. [str. 7]
- [10] Digits Home. Access control system. <https://digits-home.com/solutions/access-control> [Data uzyskania dostępu: 09.02.2024r.]. [str. 17, 74]
- [11] T. Messerschmidt J. LeBlanc. *Identity and Data Security for Web Development*. O'REILLY, 2016. [str. 19]
- [12] Duo Labs. Pythonic webauthn. https://duo-labs.github.io/py_webauthn/ [Data uzyskania dostępu: 09.02.2024r.]. [str. 47]
- [13] Matthew Miller. Simplewebauthn. <https://simplewebauthn.dev/> [Data uzyskania dostępu: 09.02.2024r.]. [str. 47, 49]
- [14] M. Mazurek R. Shay T. Vidas L. Bauer N. Christin L. Faith J. López P. Gage Kelley, S. Komanduri. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. *IEEE Symposium on Security and Privacy*, 2012. [str. 22]
- [15] Santander Bank Polska. Przykład phisingu - email firmy transportowej. <https://www.santander.pl/klient-indywidualny/bankowosc-internetowa/bezpieczne-bankowanie/phishing> [Data uzyskania dostępu: 09.02.2024r.]. [str. 21, 74]
- [16] E. Gilman D. Barth R. Rais, C. Morillo. *Zero Trust Networks*. O'REILLY, 2016. [str. 28]
- [17] S. Reddy S. Kamepalli. Analysis of secure hash algorithm (sha-512) for encryption process. *Iconic Research and Engineering Journals (IRE Journals)*, 2020. [str. 49]
- [18] N. Field S. Machani. Fido alliance white paper. 2021. [str. 26]
- [19] U. Abah N. Faruk E. Alozie L. Imoize T. Adeniran, G. Jimoh. Vulnerability assessment studies of existing knowledge-based authentication systems: A systematic review. *Sule Lamido University Journal of Science Technology*, 2024. [str. 19]
- [20] W3C. Web authentication. <https://www.w3.org/TR/webauthn-2/> [Data uzyskania dostępu: 09.02.2024r.]. [str. 25, 37, 40, 74]

- [21] W3C. Webauthn - kompatybilność przeglądarki. https://developer.mozilla.org/en-US/docs/Web/API/Web.Authentication_API/WebAuthn_extensions#browser_compatibility [Data uzyskania dostępu: 09.02.2024r.]. [str. 48, 76]
- [22] W3C. Yubico store. <https://www.yubico.com/pl/store/compare> [Data uzyskania dostępu: 09.02.2024r.]. [str. 27]
- [23] Philip J. Windley. *Learning Digital Identity*. O'REILLY, 2023. [str. 11, 14, 16]
- [24] Yubico. Get started with web authentication. <https://www.yubico.com/authentication-standards/webauthn/> [Data uzyskania dostępu: 09.02.2024r.]. [str. 26, 74]

Spis rysunków

2.1	Klucze bezpieczeństwa	13
2.2	Okno logowania do serwisu windy.com przy pomocy konta Google	15
2.3	Proces skanowania odcisków palca [10]	17
2.4	Przykład phisingu – fałszywe powiadomienie o przesyłce [15]	21
2.5	Proces zgadywania haseł realizowany z wykorzystaniem narzędzia hashcat.	23
2.6	FIDO2 - Komponenty [24]	26
2.7	Urządzenia uwierzytelniające YubiKey	27
3.1	Proces rejestracji z perspektywy użytkownika.	32
3.2	Procesu uwierzytelniania z perspektywy użytkownika	33
3.3	Struktura danych - PublicKeyCredentialCreationOptions	34
3.4	Struktura danych - AuthneticatorAttestationResponse	35
3.5	Proces rejestracji klucza	35
3.6	Dane opisujące urządzenie autentyfikujące [20]	37
3.7	Struktura danych - PublicKeyCredentialRequestOptions	38
3.8	Struktura danych - AuthenticatorAssertionResponse	38
3.9	Proces autoryzacji użytkownika	39
3.10	Generowanie sygnatury (podpisu kryptograficznego) [20]	40
3.11	Autentyfikacja Bluetooth: instrukcje w przeglądarce	42
3.12	Schemat rejestracji i autoryzacji w publikacji [3]	42
4.1	Architektura systemu Shildmail	45
4.2	Schemat bazy danych aplikacji Shildmail	47
4.3	Proces autentykacji przedstawiony na diagramie sekwencji	55
5.1	Wireshark - komunikacja w procesie rejestracji i logowania	57
5.2	Błąd: połączenie z domeną nie ujętą w ALLOWED_HOSTS	58

5.3	SecurityError - nieprawidłowa domena	59
5.4	Błąd serwerowy - rejestrowany klucz nie jest przeznaczony dla domeny <i>adamszreiber.pl</i>	60
5.5	Atak typu sniffing 2.5.1	60
5.6	Atak typu Man in the Middle 2.5.1	61
5.7	Schemat wielowarstwowych zabezpieczeń aplikacji ShildMail w modelu ISO/OSI	61
5.8	Certyfikat SSL - widok z poziomu przeglądarki	62
5.9	FIDO2 - Mechanizm walidacji po stronie klienta	63
5.10	Proces ustanawiania szyfrowanego połączenia TLS w Wireshark	64

Spis tabel

2.1	Wpływ wielkości zbioru znaków i długości hasła na czas łamania	24
2.2	Porównanie serii urządzeń YubiKey	28
4.1	Przeglądarki wspierające specyfikację FIDO [21]	48
5.1	Konfrontacja wdrożonej rejestracji z standaryzacją	57
5.2	Konfrontacja wdrozonego uwierzytelniania z standaryzacją	58

Załączniki

Kody źródłowe aplikacji