

Zero Trust 기반 원격근무 보안강화

유용상, 장예나, 한원표



목차



연구배경



선행연구자료



연구 계획



기술 구현



방향성

1. 연구 배경 _ 기획발표



1. 연구 배경



- Zero Trust 보안 아키텍처 설계
 - NIST SP 800-207 기반 7원칙 반영
 - 동적 정책 기반 접근 제어 및 지속적인 인증 구조 설계
- MFA(다중 인증) 시스템 적용
 - OTP, 생체인식, WebAuthn 등 다양한 인증 기술 비교 및 적용 실험
 - MFA 적용 전/후 보안성 및 사용자 경험 분석
- 이상 행위 탐지 시스템 구축
 - 머신러닝 기반 로그인 패턴 분석
 - 비정상 접속 탐지 및 자동 차단 시스템 구현
- 보안 게이트웨이 도입
 - Cloudflare Access, Gateway, Browser Isolation 활용
 - AWS Gateway 및 WAF, GuardDuty로 트래픽 제어 및 위협 탐지
- 테스트 및 최적화
 - Zero Trust 환경과 기존 환경 보안성 비교
 - 침투 테스트 및 성능 측정 기반 보안 정책 개선

기획발표

> 기획발표 PPT 자료

1. 연구 배경 _ 관련 기술



MFA 종류	특징	물리 계층 보안	특징
TOTP	Google Authenticator 등에서 생성되는 6자리 코드	지리적 제약	특정 장소에서만 원격 로그인 가능하도록 제한
Push 인증	로그인 시 등록된 기기로 승인 요청 전송	출입 로그 연동	출입카드 인증을 통해 원격 로그인 가능 시간 한정, 출입 기록과 로그 기록
WebAuthn	FIDO2 기반의 지문, 얼굴 인식	허가된 디바이스 인증	TPM 활용해서 등록된 특정 하드웨어만 접속 허용
SMS/Email	2단계 인증 코드 전송	화상 기반 위치 인증	WebCam으로 사용자의 실제 환경 확인 후 승인
		보안 키 보관	WebAuthn용 보안키 지정된 장소에만 보관

1. MFA 기술중 TOTP와 Push 인증 기법 구현

2. 물리 보안 계층 개념 추가

- 지리적, 시간적 의심되거나 허가되지 않은 디바이스에서 접속 시 경고 표시



2. 선행 연구자료

1. 사용자 환경에 따른 적응형 인증

- 비밀번호 없는 사용자 환경에서 제로 트러스트 기반 복합 생체 인증 플랫폼 제안
- 원격 2차 사용자 인증을 위한 FIDO 복합인증
- 인증수단 이용률로 결과 분석

(출처: “암호 없는 사용자의 2차 인증용 복합생체 기반의 FIDO 플랫폼” 강민구 2022)

2. MFA 기술 진화와 미래 전망

- SFA -> 2FA -> MFA 까지 흐름과 그에 따른 전망 예측
- Shamir's Secret Sharing 기반의 MFA 프레임워크 제안
- FAR/FRR 기반으로 사용자 경험, 보안 문제 성능 평가

(출처: “Multi-Factor Authentication: A Survey” Aleksandr Ometov 2017)

(표 3) 복합 생체인증 수단에 의한 사용자 이용률 분석

(Table 3) Analysis of usage rate by multiple biometrics

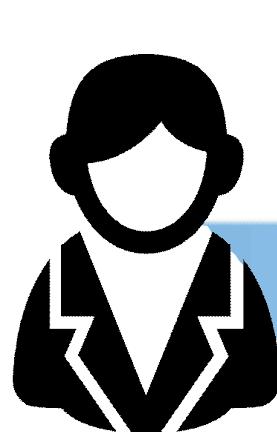
적용분야별 인증수단 이용률		Customer Identity and Access Management (CIAM), Privileged Access Management (PAM), Multi-factor Authentication (MFA)				Source: Gartner Source: Gartner Source: Gartner	
인증수단	적용분야	PDI(인증, MFA 및 대체로그인)	Remote Access (SAA, VPM-S)	PAM (특수권한 관리)	MFA	범주	대고객 관리 (Customer IAM)
IDaaS	SIEM	95.0% (인증) 98.0% (MFA)	98.0% (SAA, VPM-S)	99.0%	99.0%	99.0%	99.0%
	Cloud	98.0% (인증) 99.0% (MFA)	99.0% (SAA, VPM-S)	99.0%	99.0%	99.0%	99.0%
	HR	99.0% (인증) 99.5% (MFA)	99.5% (SAA, VPM-S)	99.5%	99.5%	99.5%	99.5%
	ITSM	99.5% (인증) 99.8% (MFA)	99.8% (SAA, VPM-S)	99.8%	99.8%	99.8%	99.8%
	DevOps	99.8% (인증) 99.9% (MFA)	99.9% (SAA, VPM-S)	99.9%	99.9%	99.9%	99.9%
	Finance	99.9% (인증) 99.95% (MFA)	99.95% (SAA, VPM-S)	99.95%	99.95%	99.95%	99.95%
OT	Time-Sensitive	99.9% (인증) 99.95% (MFA)	99.95% (SAA, VPM-S)	99.95%	99.95%	99.95%	99.95%
	Industrial	99.95% (인증) 99.99% (MFA)	99.99% (SAA, VPM-S)	99.99%	99.99%	99.99%	99.99%
	Healthcare	99.99% (인증) 99.995% (MFA)	99.995% (SAA, VPM-S)	99.995%	99.995%	99.995%	99.995%
	Manufacturing	99.995% (인증) 99.999% (MFA)	99.999% (SAA, VPM-S)	99.999%	99.999%	99.999%	99.999%
B2B2C (B2B)	CRM	99.99% (인증) 99.995% (MFA)	99.995% (SAA, VPM-S)	99.995%	99.995%	99.995%	99.995%
	Marketing	99.995% (인증) 99.999% (MFA)	99.999% (SAA, VPM-S)	99.999%	99.999%	99.999%	99.999%
	Sales	99.999% (인증) 99.9995% (MFA)	99.9995% (SAA, VPM-S)	99.9995%	99.9995%	99.9995%	99.9995%
	Customer Service	99.9995% (인증) 99.9999% (MFA)	99.9999% (SAA, VPM-S)	99.9999%	99.9999%	99.9999%	99.9999%

3. 연구 계획

테스트 시나리오 1

외부망 : 재택근무 중 민감 시스템 접근

- 원격 보안 시나리오
- 회사 외부에서 민감한 재무 시스템에 접속



직원 B

자택근무

자택 PC로 접속 시도

ZTNA 게이트웨이

- 장비 상태, IP, 위치 확인
- TPM 미등록 기기 차단
- 해외 접속은 추가 인증 요구

Push 인증 또는 OTP 입력

이상 없는 경우
시스템 접근 허용

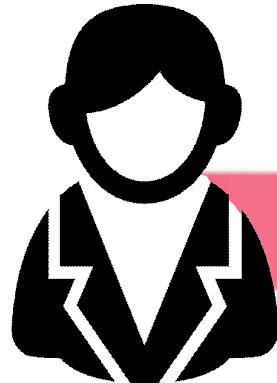
3. 연구 계획

테스트 시나리오 2



비인가 기기로 무단 접속 시도

- 이상 행위 탐지 시나리오
- 퇴직자의 이전 계정 정보가 유출되어 외부에서 로그인 시도



퇴직한
직원 C



접속 시도

- ZTNA 게이트웨이
- 등록되지 않은
기기접속 차단

위치, IP에 이상 있을
경우 관리자에게 전송



로그인 실패 기록
저장 및 분석



4. 기술 구현

MFA 추가 연구

TOTP의 주요 보안 취약점

비밀 키 유출 위험

클라이언트와 서버 모두에게 비밀 키를 저장하므로

서버에 대한 공격이 곧 시스템 내 모든 사용자의 모든 2차 인증 요소 노출로 이어질 수 있음

(출처: "T/Key: Second-Factor Authentication From Secure Hash Chains" [Dmitry Kogan](#), [Nathan Manohar](#), [Dan Boneh](#), 2017)

TOTP, Push 동시 사용
효율성 저하



Push 주 인증
TOTP 보조 인증

4. 기술 구현



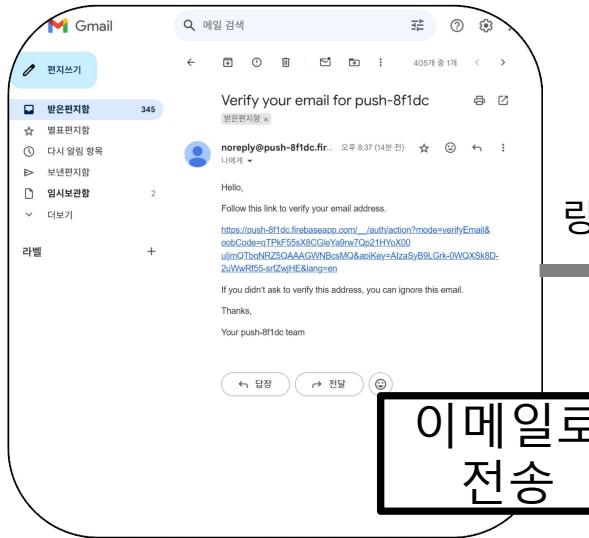
회원가입

push-says.onrender.com 내용:
인증 메일이 전송되었습니다. 메일함을 확인해주세요.

확인

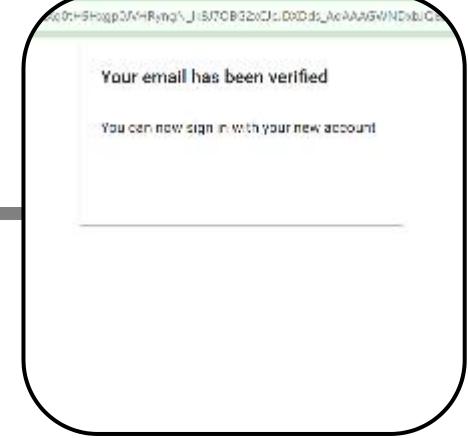
회원가입

Sign up



링크 클릭

이메일로
전송



로그인

push-says.onrender.com의 메시지
로그인 승인됨

확인

22ment.omori@gmail.com

.....

로그인 요청

로그인 승인 여부 선택

승인 거절

login

관리자 로그인 요청 리스트

- 요청: 22ment.omori@gmail.com (approved)

admin

push

Authentication

사용자

로그인 방법 템플릿 사용량 설정 Extensions

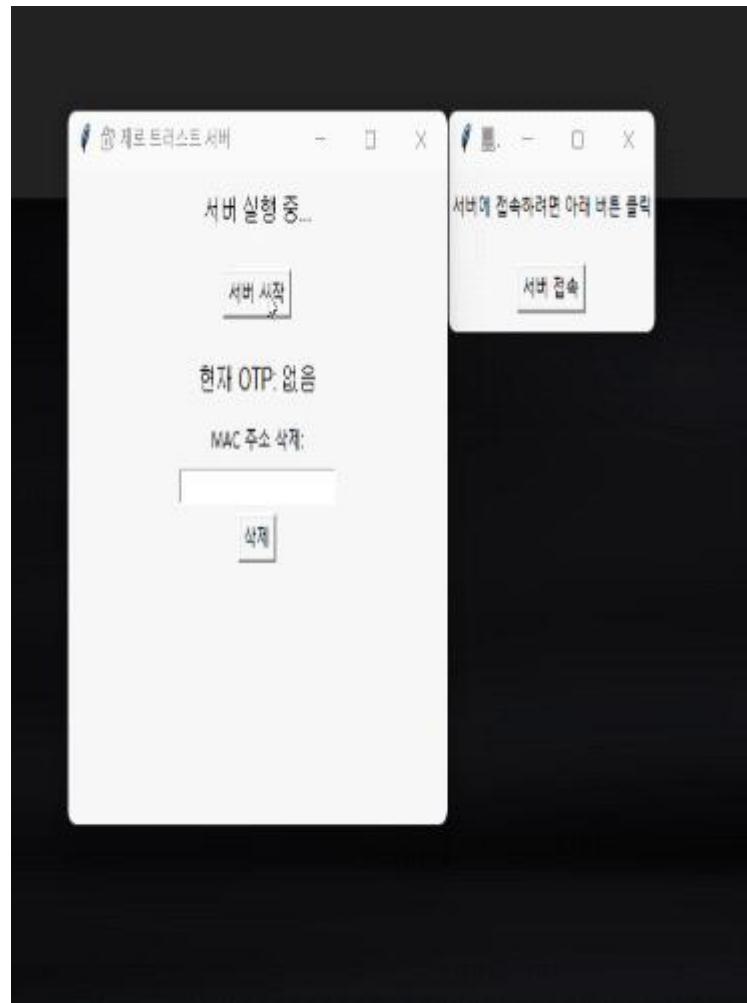
2025년 8월 25일에 Firebase 동적 링크가 종료되면 다음 인증 기능(모바일 앱의 이메일 링크 인증 및 웹 앱의 Cordova OAuth 지원)이 더 이상 작동하지 않습니다.

식별자	제공 업체	생성한 날짜	로그인한 날짜	사용자 UID
22ment.om...	...	2025. 4...	2025. 4...	oQldQEiSjqaza...
20211884@...	...	2025. 4...	2025. 4...	2rDwnVvP7gYM...
20201792@...	...	2025. 4...	2025. 4...	aQOLZosRocf2x...
hwp537244...	...	2025. 4...	2025. 4...	oDZBIRCx24Sw...

페이지당 행 수: 50 1 ~ 4 of 4

Push 인증 계정 로그

4. 기술 구현



비인가 기기 접속 거부



접속 허용 후 OTP 오류 3번



OTP인증 후 접속 성공



4. 기술 구현

The screenshot shows a web browser window with two tabs open:

- 로그인**: This tab displays a login form with fields for "이메일 입력" (Email input) and "비밀번호 입력" (Password input), and a "로그인 요청" (Login Request) button.
- 관리자 로그인 습관 모니터링**: This tab displays a title "관리자 로그인 요청 리스트" (Administrator Login Request List).

The URL for both tabs is `push-says.onrender.com/login`.

4. 기술 구현

이메일 인증, WebSocket 기반 실시간 인증 처리,
Firebase 사용자 인증을 설정하기 위한 초기 준비

```
object > server.py > ...
1  from flask import Flask, request, jsonify, render_template, redirect
2  from flask_socketio import SocketIO, emit
3  import firebase_admin
4  from firebase_admin import credentials, auth as firebase_auth
5  from smtp_utils import send_verification_email
6  import uuid, time, random, os
7  import json
8
9  app = Flask(__name__)
10 app.config['SECRET_KEY'] = 'secret!'
11 socketio = SocketIO(app)
12
13 firebase_json = os.environ.get("FIREBASE_CREDENTIALS")
14 if not firebase_json:
15     raise ValueError("FIREBASE_CREDENTIALS 환경변수가 설정되어 있지 않습니다.")
16
17 cred_dict = json.loads(firebase_json)
18 cred = credentials.Certificate(cred_dict)
19 firebase_admin.initialize_app(cred)
20
21 login_requests = {} # Push 인증 요청 저장
```



4. 기술 구현

html 사이트 페이지를 서버 가동 파이썬코드에 연결

```
24 @app.route('/')
25 def index():
26     return redirect('/login')
27
28 @app.route('/login')
29 def login_page():
30     return render_template('login.html')
31
32 @app.route('/success')
33 def success():
34     return render_template('success.html')
35
36 @app.route('/signup')
37 def signup_page():
38     return render_template('signup.html')
39
40 @app.route('/admin')
41 def admin():
42     return render_template('admin.html')
43
```

4. 기술 구현



```
59
60
61 @app.route('/request-login', methods=['POST'])
62 def request_login():
63     token = request.json.get('token')
64     try:
65         decoded = firebase_auth.verify_id_token(token)
66         email = decoded['email']
67         request_id = str(uuid.uuid4())
68         login_requests[request_id] = {
69             'email': email,
70             'status': 'pending',
71             'timestamp': time.time()
72         }
73         socketio.emit('login_request', {'request_id': request_id, 'email': email})
74         return jsonify({'request_id': request_id})
75     except Exception as e:
76         return jsonify({'error': str(e)}), 401
77
78 @app.route('/confirm-login', methods=['POST'])
79 def confirm_login():
80     data = request.json
81     request_id = data['request_id']
82     status = data['status']
83     if request_id in login_requests:
84         login_requests[request_id]['status'] = status
85         return jsonify({'result': 'ok'})
86     return jsonify({'result': 'fail'}), 400
87
88 @app.route('/check-status/<request_id>')
89 def check_status(request_id):
90     req = login_requests.get(request_id, {})
91     return jsonify({'status': req.get('status', 'unknown')})
92
93 if __name__ == '__main__':
94     port = int(os.environ.get("PORT", 5000))
95     socketio.run(app, debug=False, host='0.0.0.0', port=port)
```

사용자가 로그인 요청을 보내면 `request_id`가 생성되고,
승인자가 해당 요청을 처리하면 그 결과가 클라
이언트에 전달되는 **실시간 MFA 인증 흐름**

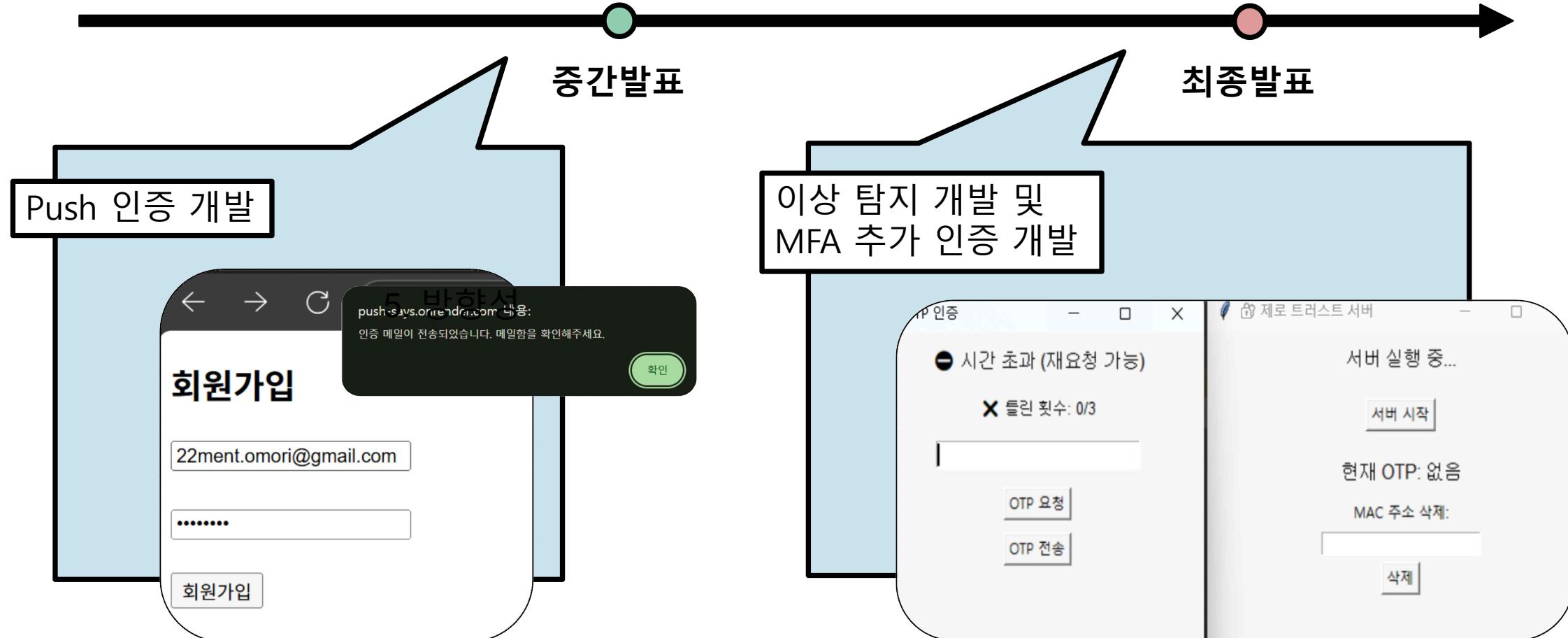
4. 기술 구현



html 코드 - Firebase 인증 키 연결

```
project > templates > <> signup.html > {} "signup.html" > ⏺ html > ⏺ head > ⏺ script > ⏺ firebaseConfig > ⏺ app
  1   <!DOCTYPE html>
  2   <html>
  3     <head>
  4       <meta charset="UTF-8">
  5       <title>회원가입 (Firebase 이메일 인증)</title>
  6       <script type="module">
  7         import { initializeApp } from "https://www.gstatic.com/firebasejs/9.22.2/
  8         import {
  9           getAuth,
 10           createUserWithEmailAndPassword,
 11           sendEmailVerification
 12         } from "https://www.gstatic.com/firebasejs/9.22.2/firebase-auth.js";
 13
 14         const firebaseConfig = {
 15           apiKey: "AIzaSyB9LGrk-0WQXSk8D-2uWwRf55-srfZwjHE",
 16           authDomain: "push-8f1dc.firebaseio.com",
 17           projectId: "push-8f1dc",
 18           storageBucket: "push-8f1dc.appspot.com",
 19           messagingSenderId: "323708834561",
 20           appId: "1:323708834561:web:48a59d3535778b67ff4100",
 21           measurementId: "G-9FXX5CDVH7"
 22         };
 23   
```

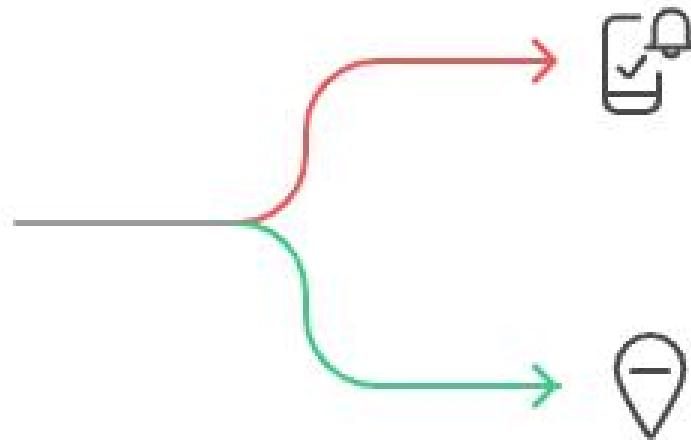
5. 방향성





5. 방향성


push 인증 요청
완료 후 위치 탐지



위치 이상 감지

추가 보안을 위해 OTP 인증을 요청
하고 푸시 알림을 보냅니다.

위치 이상 없음

로그인 프로세스를 완료합니다.



질의응답

