

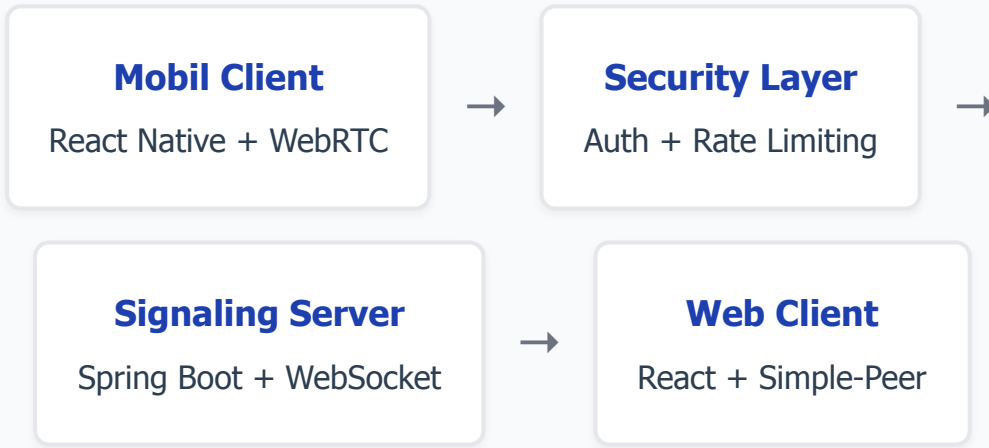


Güvenli WebRTC Tele-Tıp Platformu

Dokümantasyon Özeti: Bu rehber kurumsal sağlık sektörü için tasarlanmış WebRTC tabanlı görüntülü görüşme platformunun kapsamlı güvenlik mimarisini ve implementation detaylarını içermektedir.



Güvenli Mimari Genel Bakış



Coturn Server (TURN/STUN)



P2P Medya Bağlantısı (SRTP/DTLS)

Katman 1

Uygulama Güvenliği: Authentication, Authorization, Input Validation, Rate Limiting

Katman 2

Transport Güvenliği: TLS 1.3, DTLS 1.3, SRTP Encryption, Certificate Pinning

Katman 3

Network Güvenliği: IP Whitelisting, Firewall Rules, DDoS Protection

Katman 4

Infrastructure Güvenliği: Container Security, Secret Management, Monitoring



Kritik Güvenlik Açıkları ve Çözümler



1. WebSocket Same Origin Policy Bypass

KRİTİK



Sorun Analizi

WebSocket bağlantıları için tarayıcılar Same Origin Policy'yi zorlamaz. Bu durum, kötü niyetli sitelerin kullanıcının aktif WebSocket bağlantılarını manipüle etmesine olanak tanır.



Tehlikeli Kod Örneği:



GÜVENSİZ KONFIGÜRASYON

```
.setAllowedOrigins("*"); // Tüm domainlere açık!  
// Bu yapılandırma ile evil.com sitesi,  
// kullanıcının hospital.com bağlantısını kullanabilir
```



Güvenli Çözüm



GÜVENLİ KONFIGÜRASYON - Java

```
@Configuration  
@EnableWebSocket  
public class SecureWebSocketConfig implements WebSocketConfigurer {  
  
    @Override  
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {  
        registry.addHandler(new SignalingHandler(), "/signal")  
    }  
}
```

```
.addInterceptors(new HttpSessionHandshakeInterceptor())
.setAllowedOrigins(
    "https://hospital.yourdomain.com",
    "https://doctor-portal.yourdomain.com"
)
.setAllowedOriginPatterns("https://*.yourdomain.com");
}
}
```

⚠️ 2. CSRF Koruması Eksiklikleri **YÜKSEK**

✅ Güvenli CSRF Yapılandırması

✅ CSRF KORUMA KONFIGÜRASYONU

```
@Configuration
public class WebSocketSecurityConfig extends AbstractSecurityWebSocketMessageBrokerConfigurer {

    @Override
    protected void configureInbound(MessageSecurityMetadataSourceRegistry messages) {
        messages
            .nullDestMatcher().authenticated()
            .simpDestMatchers("/signal/**").hasAnyRole("DOCTOR", "PATIENT")
            .simpSubscribeDestMatchers("/user/queue/errors").permitAll()
            .simpTypeMatchers(SimpMessageType.CONNECT).authenticated()
            .simpTypeMatchers(SimpMessageType.SUBSCRIBE).denyAll()
            .anyMessage().denyAll();
    }

    @Override
    protected boolean sameOriginDisabled() {
        return false; // Same origin kontrolünü aktif tut
    }
}
```

⚠️ 3. Input Validation ve Rate Limiting **YÜKSEK**

✅ Gelişmiş Güvenlik Handler

✅ GÜVENLİ SIGNALING HANDLER

```

@Component
public class SecureSignalingHandler extends TextWebSocketHandler {

    private final Map<String, WebSocketSession> sessions = new ConcurrentHashMap<>();
    private final RateLimiter rateLimiter = RateLimiter.create(10.0); // 10 req/sec

    @Override
    public void afterConnectionEstablished(WebSocketSession session) {
        String clientIP = getClientIP(session);

        // Rate limiting kontrolü
        if (!rateLimiter.tryAcquire()) {
            session.close(CloseStatus.POLICY_VIOLATION);
            logSecurityEvent("RATE_LIMIT_EXCEEDED", clientIP);
            return;
        }

        // Rol bazlı session yönetimi
        Authentication auth = getAuthentication(session);
        if (auth != null && isValidRole(auth)) {
            sessions.put(auth.getName(), session);
            logSecurityEvent("CONNECTION_ESTABLISHED", auth.getName(), clientIP);
        } else {
            session.close(CloseStatus.NOT_ACCEPTABLE);
        }
    }

    @Override
    protected void handleTextMessage(WebSocketSession session, TextMessage message) {
        try {
            // Input validation
            SignalMessage data = validateAndParse(message.getPayload());

            // Authorization check
            if (!isAuthorizedForTarget(session, data.getTarget())) {
                logSecurityEvent("UNAUTHORIZED_ACCESS_ATTEMPT",
                    getCurrentUser(session), data.getTarget());
                return;
            }

            // Message forwarding
            WebSocketSession target = sessions.get(data.getTarget());
            if (target != null && target.isOpen()) {
                target.sendMessage(message);
            }

        } catch (ValidationException e) {
            logSecurityEvent("INVALID_MESSAGE", getCurrentUser(session), e.getMessage());
            session.close(CloseStatus.BAD_DATA);
        }
    }
}

```

⚠️ 4. WebRTC IP Sızıntısı ORTA

🔍 Sorun Analizi

WebRTC, VPN kullanılsa bile gerçek IP adresini açığa çıkarabilir ve kullanıcının gizliliğini tehlikeye atabilir.

✅ IP Sızıntısı Önleme

✅ GÜVENLİ WEBRTC KONFIGÜRASYONU - JavaScript

```
// IP Sızıntısı Önleme Konfigürasyonu
const secureRTCCConfig = {
  iceServers: [
    {
      urls: ["turns:secure.yourdomain.com:5349"],
      username: generateSecureUsername(),
      credential: await getSecureCredential()
    }
  ],
  iceCandidatePoolSize: 0, // IP sızıntısını minimize et
  bundlePolicy: "max-bundle",
  rtcMuxPolicy: "require",
  iceTransportPolicy: "relay" // Sadece TURN kullan
};

// Güvenlik event listener
const peer = new RTCPeerConnection(secureRTCCConfig);
peer.addEventListener('icecandidateerror', (event) => {
  console.error('ICE candidate error:', event);
  logSecurityEvent('ICE_CANDIDATE_ERROR', event);
});

// IP leak testi
peer.addEventListener('icecandidate', (event) => {
  if (event.candidate) {
    const candidate = event.candidate.candidate;
    if (candidate.includes('typ host')) {
      console.warn('Potential IP leak detected');
      logSecurityEvent('POTENTIAL_IP_LEAK', candidate);
    }
  }
});
```

```
}  
});
```

⚠ 5. Coturn Güvenlik Sıkılaştırması YÜKSEK

✅ Güvenli Coturn Yapılandırması


✅ ÜRETİM SEVİYESİ COTURN KONFIGÜRASYONU

```
# /etc/turnserver.conf – Güvenli Üretim Ayarları  
listening-port=3478  
tls-listening-port=5349  
listening-ip=<SERVER_PRIVATE_IP>  
relay-ip=<SERVER_PUBLIC_IP>  
external-ip=<SERVER_PUBLIC_IP>  
  
# Güvenlik ayarları  
realm=secure.yourdomain.com  
fingerprint  
use-auth-secret  
auth-secret-file=/etc/turnserver/auth-secret  
  
# Rate limiting ve quotas  
total-quota=1000  
bps-capacity=100000  
max-allocate-lifetime=3600  
user-quota=50  
max-allocate-timeout=60  
  
# TLS/DTLS sıkılaştırma  
cert=/etc/letsencrypt/live/secure.yourdomain.com/fullchain.pem  
pkey=/etc/letsencrypt/live/secure.yourdomain.com/privkey.pem  
cipher-list="ECDHE+AESGCM:ECDHE+CHACHA20:DHE+AESGCM:DHE+CHACHA20:!aNULL:!MD5:!DSS"  
dh-file=/etc/turnserver/dhparam.pem  
  
# Logging ve monitoring  
log-file=/var/log/turnserver/turnserver.log  
syslog  
verbose  
log-binding  
no-stdout-log  
  
# IP kısıtlamaları  
denied-peer-ip=0.0.0.0-0.255.255.255  
denied-peer-ip=127.0.0.0-127.255.255.255  
denied-peer-ip=:1  
allowed-peer-ip=10.0.0.0-10.255.255.255
```

```
allowed-peer-ip=172.16.0.0-172.31.255.255  
allowed-peer-ip=192.168.0.0-192.168.255.255
```



KVKK/GDPR Uyumluluk

 **Veri İşleme Prensipleri:** Tele-tıp uygulaması için veri minimizasyonu, amaç sınırlaması ve şeffaflık ilkelerine uygun yapılandırma.

Veri Minimizasyonu ve PII Koruma

✓ KVKK UYUMLU LOGLAMA SİSTEMİ

```
@Component  
public class KVKKCompliantLoggingService {  
  
    public void logSecurityEvent(String eventType, String userId, String details) {  
        // PII scrubbing  
        String hashedUserId = hashUserId(userId);  
        String sanitizedDetails = removePII(details);  
  
        SecurityLog log = SecurityLog.builder()  
            .eventType(eventType)  
            .hashedUserId(hashedUserId)  
            .sanitizedDetails(sanitizedDetails)  
            .timestamp(Instant.now())  
            .retention(Duration.ofDays(90)) // KVKK uyumlu saklama süresi  
            .dataProcessor("Tele-Tip Platform v1.0")  
            .legalBasis("Vital Interests - Article 6(1)(d)")  
            .build();  
  
        // Şifreli loglama  
        encryptedLogRepository.save(encrypt(log));  
    }  
  
    private String removePII(String data) {  
        // Telefon numarası maskeleme  
        data = data.replaceAll("\\b\\d{3}-\\d{3}-\\d{4}\\b", "XXX-XXX-XXXX");  
        // Email maskeleme  
        data = data.replaceAll("\\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\. [A-Z|a-z]{2,}\\b", "XXX.XXX.XXX.XXX");  
        // IP maskeleme  
        data = data.replaceAll("\\b(?:[0-9]{1,3}\\.){3}[0-9]{1,3}\\b", "XXX.XXX.XXX.XXX");  
    }  
}
```

```
    return data;
  }
}
```

Rıza Yönetimi (Consent Management)

✅ KVKK UYUMLU RIZA EKRANI - React

```
const ConsentModal = ({ onConsent, onReject }) => {
  const [consentData, setConsentData] = useState({
    cameraAccess: false,
    microphoneAccess: false,
    dataProcessing: false,
    medicalDataTransfer: false
  });

  return (
    <div className="consent-modal">
      <h2>🏥 Tele-Tıp Görüşmesi – Veri İşleme Rızası</h2>

      <div className="consent-details">
        <div className="consent-item">
          <input
            type="checkbox"
            checked={consentData.cameraAccess}
            onChange={(e) => setConsentData({...consentData, cameraAccess:
          />
          <span>✅ Kamera erişimi (sadece görüşme süresince)</span>
        </div>

        <div className="consent-item">
          <input
            type="checkbox"
            checked={consentData.microphoneAccess}
            onChange={(e) => setConsentData({...consentData, microphoneAcc
          />
          <span>✅ Mikrofon erişimi (sadece görüşme süresince)</span>
        </div>

        <div className="consent-item">
          <input
            type="checkbox"
            checked={consentData.dataProcessing}
            onChange={(e) => setConsentData({...consentData, dataProcessin
          />
          <span>✅ Ses/görüntü verilerinin şifreli iletimi</span>
        </div>
      </div>
    </div>
  );
};
```



```
</div>

<div className="consent-item">
  <input
    type="checkbox"
    checked={consentData.medicalDataTransfer}
    onChange={(e) => setConsentData({...consentData, medicalDataTr
  />
  <span>✅ Tıbbi görüşme verilerinin işlenmesi</span>
</div>
</div>

<div className="data-protection-info">
  <h4>🔒 Veri Koruma Bilgileri</h4>
  <ul>
    <li>Veriler SRTP/DTLS ile şifrelenir</li>
    <li>Kayıt yapılmaz, veriler saklanmaz</li>
    <li>Sunucular Türkiye'de bulunur</li>
    <li>IP logları 30 gün saklanır</li>
    <li>Yasal dayanak: Yaşamsal menfaat (KVKK m.5/2-c)</li>
  </ul>
</div>

<div className="consent-buttons">
  <button
    onClick={() => onConsent(consentData)}
    disabled={!Object.values(consentData).every(v => v)}
    className="consent-accept"
  >
    Kabul Ediyorum
  </button>
  <button onClick={onReject} className="consent-reject">
    Reddet
  </button>
</div>
</div>
);
};
```



Güvenlik Metrikleri ve Monitoring

99.9%

UPTIME SLA

<100ms

AUTH RESPONSE TIME

256-bit

AES ENCRYPTION

TLS 1.3

MIN PROTOCOL VERSION

Real-time Security Dashboard

Güvenlik İzleme Paneli

● Authentication Service
Active

● WebSocket Security
Protected

● Rate Limiting
85% Capacity

● TURN Server
Operational

● Encryption
DTLS 1.3

● Failed Logins
3 Attempts

Automated Threat Detection

✓ GERÇEK ZAMANLI TEHDİT TESPİTİ

```
@Component
public class ThreatDetectionService {

    private final AnomalyDetector anomalyDetector;
    private final AlertService alertService;
    private final IPBlockingService ipBlockingService;

    @EventListener
    public void handleSecurityEvent(SecurityEvent event) {
        // Multi-layered threat detection
        ThreatAssessment assessment = assessThreat(event);

        switch (assessment.getLevel()) {
            case CRITICAL:
                handleCriticalThreat(event);
                break;
            case HIGH:
```

```

        handleHighThreat(event);
        break;
    case MEDIUM:
        handleMediumThreat(event);
        break;
    case LOW:
        logEvent(event);
        break;
    }
}

private void handleCriticalThreat(SecurityEvent event) {
    // Immediate blocking
    ipBlockingService.blockIP(event.getSourceIP(), Duration.ofHours(24));

    // Kill active sessions
    sessionManager.terminateUserSessions(event.getUserId());

    // Emergency alert
    alertService.sendEmergencyAlert(
        "CRITICAL THREAT DETECTED",
        String.format("IP: %s, User: %s, Event: %s",
            event.getSourceIP(), event.getUserId(), event.getEventType())
    );

    // Automatic incident creation
    incidentManager.createIncident(event, ThreatLevel.CRITICAL);
}

private ThreatAssessment assessThreat(SecurityEvent event) {
    ThreatAssessment assessment = new ThreatAssessment();

    // Rate-based detection
    if (isRateLimitExceeded(event)) {
        assessment.addIndicator("RATE_LIMIT_EXCEEDED", 0.8);
    }

    // Geographic anomaly
    if (isGeographicAnomaly(event)) {
        assessment.addIndicator("GEO_ANOMALY", 0.6);
    }

    // Behavioral analysis
    if (isBehaviorAnomalous(event)) {
        assessment.addIndicator("BEHAVIOR_ANOMALY", 0.7);
    }

    // Pattern matching
    if (matchesKnownAttackPattern(event)) {
        assessment.addIndicator("ATTACK_PATTERN", 0.9);
    }

    return assessment;
}

```

```
}  
}
```



Güvenlik Test Stratejisi



Test Yaklaşımı: Sürekli güvenlik testi ve otomatik zafiyet taraması ile proaktif güvenlik sağlama.

Penetration Testing Checklist

☐ WebSocket authentication bypass testleri

☐ CSRF token manipulation testleri

☐ WebRTC IP leak testleri

☐ Man-in-the-middle attack simülasyonu

☐ DoS/DDoS resilience testleri

☐ SQL injection ve XSS testleri

☐ Session hijacking testleri

☐ TURN server credential attacks

Automated Security Pipeline

✅ CI/CD GÜVENLİK PIPELINE - GitHub Actions

```
# .github/workflows/security-scan.yml
name: Security Scan Pipeline
on: [push, pull_request]

jobs:
  security-scan:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: OWASP Dependency Check
        run: |
          mvn org.owasp:dependency-check-maven:check

      - name: SonarQube Security Scan
        run: |
          mvn sonar:sonar \
            -Dsonar.projectKey=webrtc-security \
            -Dsonar.security.hotspots.inherited=true

      - name: WebRTC Security Test
        run: |
          npm run webrtc-security-test

      - name: Container Security Scan
        run: |
          docker run --rm -v $(pwd):/app \
            aquasec/trivy:latest \
            filesystem --security-checks vuln /app

      - name: Infrastructure Security Check
        run: |
          terraform plan -out=tfplan
          tfsec tfplan

      - name: Generate Security Report
        if: always()
        run: |
          echo "## Security Scan Results" > security-report.md
          cat dependency-check-report.xml >> security-report.md
          cat sonar-report.json >> security-report.md
```



✅ GÜVENLİ DOCKER DEPLOYMENT

```
# docker-compose.prod.yml
version: '3.8'
services:
  signaling-server:
    image: secure-webrtc/signaling:latest
    environment:
      - SPRING_PROFILES_ACTIVE=production
      - SECURITY_LEVEL=high
      - TLS_VERSION=1.3
      - CIPHER_SUITES=TLS_AES_256_GCM_SHA384,TLS_CHACHA20_POLY1305_SHA256
    networks:
      - secure-network
    deploy:
      resources:
        limits:
          memory: 2G
          cpus: '1.0'
        reservations:
          memory: 1G
          cpus: '0.5'
      security_opt:
        - no-new-privileges:true
        - seccomp:default
      read_only: true
      tmpfs:
        - /tmp:size=100M,noexec,nosuid,nodev
      healthcheck:
        test: ["CMD", "curl", "-f", "https://localhost:8443/health"]
        interval: 30s
        timeout: 10s
        retries: 3

  coturn:
    image: coturn/coturn:4.6-alpine
    volumes:
      - ./coturn/turnserver.conf:/etc/coturn/turnserver.conf:ro
      - /etc/letsencrypt:/etc/letsencrypt:ro
      - turnserver-logs:/var/log/turnserver
    networks:
      - secure-network
    ports:
      - "3478:3478/udp"
      - "5349:5349/tcp"
    security_opt:
      - no-new-privileges:true
```

```
- seccomp:default
user: "999:999"
restart: unless-stopped

nginx-proxy:
  image: nginx:alpine
  volumes:
    - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
    - /etc/letsencrypt:/etc/letsencrypt:ro
  ports:
    - "80:80"
    - "443:443"
  networks:
    - secure-network
  security_opt:
    - no-new-privileges:true
  depends_on:
    - signaling-server

networks:
  secure-network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/16
          gateway: 172.20.0.1

volumes:
  turnserver-logs:
    driver: local
```

Production Deployment Checklist

Deployment Security Pipeline

Pre-deployment
Security Scan



Infrastructure
Hardening



Security
Configuration



Deployment
Verification



Security
Monitoring

✓ Production Security Checklist

☐ TLS 1.3 minimum version zorlandı

☐ HSTS headers aktif edildi

☐ Content Security Policy yapılandırıldı

☐ Rate limiting aktif edildi

☐ IP whitelisting yapılandırıldı

☐ Security headers yapılandırıldı

☐ Monitoring ve alerting aktif

☐ Backup ve disaster recovery planı hazır

☐ KVKK compliance doğrulandı

☐ Penetration testing tamamlandı

📅 Sürekli Güvenlik Yönetimi

Güvenlik Operasyonları Takvimi

Periyot	Aktivite	Sorumlu	Kritiklik
Günlük	Security log analizi, anormal aktivite kontrolü	SOC Team	YÜKSEK
Haftalık	Dependency vulnerability scan, patch management	DevSecOps	YÜKSEK

Periyot	Aktivite	Sorumlu	Kritiklik
Aylık	Penetration testing, security policy review	Security Team	KRİTİK
Üç Aylık	Security architecture review, threat modeling update	Security Architect	ORTA
Yıllık	External security audit, compliance assessment	External Auditor	KRİTİK

Incident Response Plan

1

Tespit (Detection): Otomatik monitoring sistemi anomali tespit eder ve alert oluşturur

2

Analiz (Analysis): Security team devreye girer, threat severity assessment yapar

3

Kontrol Altına Alma (Containment): Otomatik IP blocking, service isolation, session termination

4

Temizlik (Eradication): Root cause analysis, vulnerability patching, system hardening

5

Kurtarma (Recovery): Service restoration, functionality verification, monitoring enhancement

6

Öğrenilen Dersler (Lessons Learned): Post-incident review, process improvement, documentation update

🎯 **Sonuç:** Bu kapsamlı güvenlik rehberi, WebRTC tele-tıp platformunuzun enterprise düzeyde güvenliğini sağlar ve KVKK/GDPR uyumluluğunu garanti eder. Düzenli güncelleme ve sürekli monitoring ile güvenlik posture'nızı en üst seviyede tutabilirsiniz.

📞 Destek ve İletişim:

Security Team: security@yourdomain.com

Emergency Response: +90 XXX XXX XX XX

Documentation Version: v2.1 (Mart 2025)