Table of Contents

# 1 Telecom Call Center Infrastructure - Technical Overview

## 1.1 프로젝트 개요

**기간**: 2019.12 ~ 2021.08 (1년 9개월) **소속**: ㈜메타넷엠플랫폼 **역할**: 인프라·시스템 엔지니어 **프로젝트**: 대규모 통신사 콜센터 인프라 구축 및 운영 **규모**: 상담원 1,000명 이상, 신규 사이트 3개소 구축

## 1.2 시스템 아키텍처

### 1.2.1 대규모 재택근무 환경 구축

**배경**: COVID-19 팬데믹 대응 긴급 재택근무 인프라 구축

**규모**: - 동시 접속자: 1,000명 이상 - 신규 사이트: 3개소 (각 500명 규모) - VPN 동시 세션: 1,000+ concurrent connections

**구성요소**: 1. **SSL VPN Infrastructure** - HA (High Availability) 구성 - 로드 밸런싱을 통한 세션 분산 - 2FA (Two-Factor Authentication) 적용

1. **NAC (Network Access Control)**
   * 재택근무 단말 보안 정책 준수 검사
   * 백신, 패치 상태 자동 검증
   * 비인가 단말 접속 차단
2. **네트워크 인프라**
   * 다중 ISP 구성 (회선 이중화)
   * QoS (Quality of Service) 적용
   * 대역폭 최적화

## 1.3 주요 시스템

### 1.3.1 1. SSL VPN 통합 플랫폼

**아키텍처**: - Active-Active HA 구성 - 세션 지속성(Session Persistence) 보장 - 자동 Failover (30초 이내)

**보안**: - 2FA (OTP 기반 이중 인증) - 단말 인증서 기반 접속 제어 - 세션 암호화 (AES-256)

**성과**: - 동시 접속자 1,000명 안정적 지원 - VPN 가용률 99.8% 달성 - 평균 접속 시간 3초 이내

### 1.3.2 2. NAC 솔루션 운영

**기능**: - 단말 등록 및 인증 - 보안 정책 준수 검사 - 예외 정책 관리 (Ansible 자동화) - 실시간 단말 모니터링

**자동화**: - Ansible Playbook 기반 예외 정책 자동 배포 - 처리 시간 90% 단축 (건당 30분 → 3분) - 수작업 오류 100% 제거

**성과**: - 단말 보안 정책 준수율 95% 이상 - 비인가 단말 접속 차단 100% - 예외 정책 처리 시간 90% 단축

### 1.3.3 3. 네트워크 스위치 자동 점검 시스템

**배경**: 주간 수동 점검으로 인한 높은 운영 부담

**개발**: - Python 기반 네트워크 스위치 자동 점검 스크립트 - SSH 기반 장비 접속 및 명령 실행 - 로그 파싱 및 이상 징후 탐지 - 자동 리포트 생성 (HTML, PDF)

**점검 항목**: - 포트 상태 (Up/Down) - CPU/메모리 사용률 - 에러 카운터 - 링크 상태 및 속도 - VLAN 설정 검증

**성과**: - 주당 소요 시간 75% 단축 (8시간 → 2시간) - 점검 정확도 향상 (인적 오류 제거) - 실시간 알림 기능 추가

## 1.4 주요 성과

### 1.4.1 1. 자동화 구현

#### 1.4.1.1 Python 네트워크 자동 점검 시스템

**목표**: 수동 점검 작업 자동화

**구현**:

# 주요 기능  
- SSH 멀티 세션 관리  
- 병렬 처리 (ThreadPoolExecutor)  
- 이상 징후 자동 탐지  
- HTML/PDF 리포트 생성  
- Slack/이메일 알림

**성과**: - 주당 소요 시간 75% 단축 (8시간 → 2시간) - 점검 대상 스위치 수 200% 증가 (50대 → 150대) - 야간 자동 점검으로 24/7 모니터링

#### 1.4.1.2 Ansible NAC 정책 자동화

**목표**: NAC 예외 정책 배포 자동화

**구현**:

# Ansible Playbook 구조  
- 단말 정보 수집  
- 정책 템플릿 생성  
- NAC API 호출  
- 배포 검증  
- 롤백 대비

**성과**: - 처리 시간 90% 단축 (건당 30분 → 3분) - 수작업 오류 100% 제거 - 정책 일관성 보장

### 1.4.2 2. 안정성 개선

#### 1.4.2.1 백신-VPN 충돌 해결

**문제**: 특정 백신 제품과 VPN 클라이언트 충돌로 접속 불가

**분석**: - 패킷 캡처 및 분석 - 프로세스 모니터링 - 레지스트리 충돌 확인

**해결**: - 백신 정책 예외 추가 - VPN 클라이언트 버전 업그레이드 - 방화벽 룰 최적화

**성과**: - 장애 문의 40% 감소 (주 20건 → 12건) - VPN 접속 성공률 95% → 99.5% 향상 - 사용자 만족도 향상

### 1.4.3 3. 아키텍처 설계

#### 1.4.3.1 신규 사이트 3개소 네트워크 설계

**규모**: 각 사이트 동시 접속자 500명

**설계 요소**: - 네트워크 토폴로지 설계 - IP 주소 체계 수립 - VLAN 분리 (업무/관리망) - QoS 정책 수립 - 이중화 구성

**성과**: - 3개소 모두 일정 내 오픈 - 안정적인 서비스 제공 (가용률 99.9%) - 확장 가능한 아키텍처

## 1.5 기술 스택

### 1.5.1 Infrastructure

* **Network**: Cisco Catalyst, Nexus
* **VPN**: SSL VPN (F5 BIG-IP APM, FortiGate)
* **Load Balancer**: F5 BIG-IP LTM
* **NAC**: Network Access Control

### 1.5.2 Automation

* **Scripting**: Python 3.x
  + Paramiko (SSH)
  + Threading (병렬 처리)
  + Jinja2 (템플릿)
  + Pandas (데이터 처리)
* **Configuration Management**: Ansible
  + Playbooks
  + Roles
  + Inventory Management
  + API Integration

### 1.5.3 Monitoring

* **Network Monitoring**: Zabbix, MRTG
* **Log Analysis**: ELK Stack
* **APM**: Application Performance Monitoring

## 1.6 핵심 역량

1. **대규모 재택근무 인프라**: 1,000명 규모 VPN 환경 구축 및 운영
2. **자동화 개발**: Python 기반 네트워크 점검 자동화, Ansible 기반 정책 배포 자동화
3. **네트워크 설계**: 신규 사이트 3개소 네트워크 아키텍처 설계 및 구축
4. **장애 분석**: 백신-VPN 충돌 근본 원인 분석 및 해결
5. **성능 최적화**: QoS 적용 및 대역폭 최적화를 통한 안정적인 서비스 제공

## 1.7 교훈 및 인사이트

### 1.7.1 1. 긴급 재택근무 인프라의 중요성

COVID-19 팬데믹 초기, 2주 만에 1,000명 규모 재택근무 환경을 구축했습니다. 평소 구축해둔 VPN 인프라와 자동화 시스템이 신속한 대응을 가능하게 했습니다.

### 1.7.2 2. 자동화의 실질적 가치

Python 스크립트로 네트워크 점검을 자동화하여 주당 6시간을 절약했습니다. 절약된 시간은 더 중요한 아키텍처 설계와 장애 분석에 투입되었습니다.

### 1.7.3 3. 근본 원인 분석의 중요성

백신-VPN 충돌 문제를 단순 회피가 아닌 근본 원인 분석을 통해 해결하여 장애 문의를 40% 감소시켰습니다. 임시방편이 아닌 근본 해결이 중요합니다.

### 1.7.4 4. 확장 가능한 아키텍처

신규 사이트 3개소를 설계할 때 향후 확장성을 고려하여 모듈형 아키텍처를 적용했습니다. 이후 추가 사이트 구축 시 설계를 재사용할 수 있었습니다.

**문서 작성일**: 2025-10-20 **작성자**: 이재철 (인프라·보안 엔지니어)