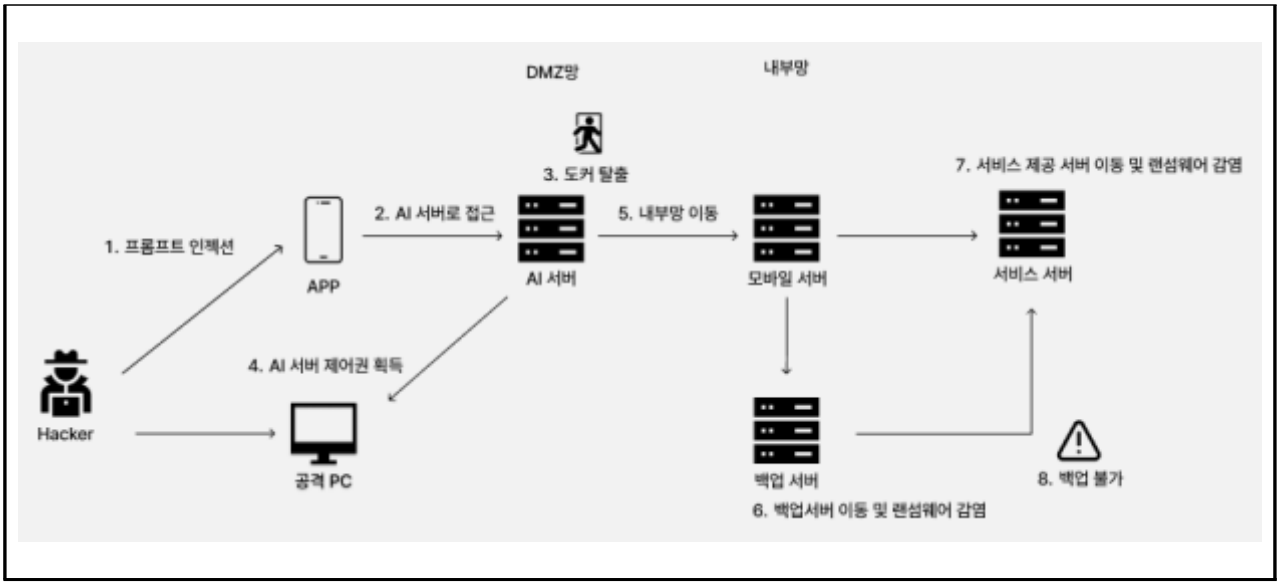
<붙임 2> 시나리오 제출양식

시나리오 제목
모바일 앱 AI 챗봇 기능 악용을 통한 내부망 침투 및 랜섬웨어 공격 시나리오
시나리오 단계
총 7단계 (모바일 앱 챗봇 리버스 쉘 시도 & 필터링 탐지 → 앱 디컴파일·보안 로직 분석 & Frida 탐지 우회 설계 → Frida 기반 필터 우회 적용 & 리버스 쉘 질의 전송 성공 → 컨테이너 정찰 후 호스트 서버 탈출 → 모바일 서버로의 피봇 및 내부망 확산 → 백업 서버 권한 장악 및 AWS 키 탈취 → 랜섬웨어 동시 배포로 서비스·백업 마비)
시나리오 목적
본 시나리오는 실제 조직 환경을 가정하여 모바일 앱의 단일 취약점이 조직 전체 보안 체계에 미치는 영향을 검증하기 위해 설계되었다. 공격자는 앱 리버스 엔지니어링, 프롬프트 인젝션, 서버 간 신뢰 관계, 키 관리 취약점 등 현실적인 공격 벡터를 체계적으로 조합하여 앱 → 서버 → 내부망 → 백업 인프라 → 랜섬웨어 공격으로 이어지는 공격 체인을 완성한다. 이를 통해 모바일 앱 보안 점검 시 반드시 고려해야 하는 서버 보안, 키 관리, 백업 시스템 보호 등 종합적인 엔드투엔드 보안 대책 필요성을 입증하고, 침투 테스트 결과를 바탕으로 위험 평가와 보안 정책 수립을 지원한다.
시나리오 줄거리
공격자는 APK를 디컴파일하여 앱의 보안 솔루션과 필터링 로직을 분석하고, 동적 후킹으로 보안 탐지를 우회한다. 이후 챗봇 입력 필터링 취약점을 악용해 Docker 컨테이너 내부 쉘을 확보하고, 알려진 runc 취약점(CVE-2024-21626)을 활용해 호스트 서버로 탈출한다. 확보한 서버 권한을 기반으로 모바일 서버와 백업 서버를 연계 침투하고, SSH 키 관리 허점을 이용해 다수 서버 접근 권한을 확보한다. 마지막으로 AES+RSA 암호화 기반 랜섬웨어를 배포해 웹 서버·모바일 서버·백업 서버를 동시에 마비시키며, 조직 전체 인프라가 단일 취약점으로 인한 체인 리스크에 취약함을 증명한다. 이 시나리오는 단순 앱 해킹을 넘어 인프라 레벨의 보안 위협을 입체적으로 검증할 수 있는 고난도 침투 테스트 사례다.
시나리오 네트워크 구성도


DMZ망
내부망
3. 도커 탈출
7. 서비스 제공 서버 이동 및 랜섬웨어 감염
1. 프롬프트 인젝션
2. AI 서버로 접근
5. 내부망 이동
APP
AI 서버
모바일 서버
서비스 서버
4. AI 서버 제어권 획득
Hacker
공격 PC
백업 서버
8. 백업 불가
6. 백업서버 이동 및 랜섬웨어 감염

단계 번호		1
단계 명		모바일 챗봇 리버스 쉘 시도 & 필터링 탐지
주요 목표		챗봇 입력값 기반 취약점 테스트, 코드 실행 가능성 확인, 클라이언트 필터링 존재 파악
유형	**공격**	챗봇 리버스 쉘 코드 삽입 테스트, 오류 메시지 분석
	방어	앱 단 입력값 검증, 서버 측 코드 실행 요청 차단
주요 활동		공격자는 먼저 챗봇 질의에 파이썬 기반 리버스 쉘 코드를 포함해 전송해 보며 동작 여부를 점검한다. 응답 과정에서 나타난 오류 메시지 및 시스템 힌트를 분석해 챗봇이 Docker 컨테이너 안에서 Python 코드를 실행해 응답을 생성한다는 점을 추론하지만, 클라이언트 측 입력값 필터링에 의해 해당 질의가 서버에 전달되기 전에 차단됨을 확인한다. 이로써 “우선 필터링/탐지 우회가 선행되어야 한다”는 공격 전술적 결론을 내린다. 방어자는 앱에서 클라이언트 입력값을 검증하고 코드 실행 관련 키워드를 차단하여 1차 방어선을 유지한다. 서버 측에서도 의심스러운 요청을 차단하고 오류 메시지를 외부에 노출하지 않도록 설계해야 한다.
주요 사용 도구 및 기술		Netcat, BurpSuite, Python Reverse Shell
예상 결과물 및 상태		코드 실행 가능성 추론, 앱 필터링 체계 확인, 우회 필요성 도출

단계 번호		2
단계 명		앱 디컴파일·보안 로직 분석 & Frida 탐지 우회 설계
주요 목표		앱 내부 보안 솔루션 구조 및 탐지 루틴 분석, 런타임 우회 전략 수립
유형	공격	APK 디컴파일, IDA Pro 정적 분석, Frida 탐지 경로 역공학
	방어	난독화, 무결성 검증, 런타임 탐지 강화
주요 활동		공격자는 APK를 디컴파일해 입력 필터링 로직의 구체적 차단 규칙(파이썬 키워드·함수 호출 패턴 등)과 탐지/무결성 검사 경로를 식별한다. 이어 런타임 후킹으로 필터를 무력화하려 하나, 앱이 Frida 동작을 감지하면 즉시 종료되는 것을 확인한다. 보안 솔루션이 자바 레벨이 아닌 네이티브(so) 라이브러리에서 Frida를 탐지한다는 구조적 사실을 파악하고, IDA Pro 분석으로 탐지 루틴의 분기·검사 지점을 역추적해 우회 포인트를 특정한다. 그 결과, 런타임에서 탐지 플래그를 무력화하고, 동시에 블랙리스트 기반 필터 결과를 '허용' 쪽으로 강제하는 후킹 전략을 수립한다. 방어자는 앱 난독화, 네이티브·자바 이중 탐지, 코드 무결성 체크를 강화해 리버스 엔지니어링 난이도를 높이고 런타임 후킹을 탐지·차단해야 한다.
주요 사용 도구 및 기술		Apktool, JADX, IDA Pro, Frida
예상 결과물 및 상태		보안 탐지 우회 전략 수립, 입력 필터 조작 가능 상태 확보

단계 번호		3
단계 명		Frida 기반 필터 우회 적용 & 리버스 쉘 질의 전송 성공
주요 목표		런타임 탐지 우회, 입력 필터 무력화, 챗봇 취약점 악용 성공
유형	**공격**	Frida 후킹, 필터 무력화, 리버스 쉘 연결
	방어	런타임 탐지 로깅, 보안 솔루션 이벤트 알림
주요 활동		공격자는 설계한 후킹을 적용해 Frida 탐지 우회와 입력 필터 무력화를 동시에 달성한다. 이후 동일한 맥락의 파이썬 리버스 쉘 코드를 다시 챗봇 질의에 포함해 전송하고, 이번에는 클라이언트 필터를 통과해 서버까지 도달함을 확인한다. 챗봇 서버가 컨테이너 내부에서 해당 코드를 실행하면서 공격자 측 리스너로 역방향 세션이 성립되고, 연결 직후 환경 확인(계정·커널·네트워크 특징)을 통해 Docker 컨테이너 맥락임을 확정한다. 즉, “파이썬 쉘 코드를 재분석→Frida로 필터 우회→AI 챗봇 취약점으로 리버스 쉘 연결”이라는 핵심 목표를 완수한다. 방어자는 보안 솔루션 탐지 시 앱 종료뿐 아니라 서버 측 경보와 로깅을 수행해 우회 시도를 추적하고, 서버 입력 검증 체계를 강화한다.
주요 사용 도구 및 기술		보안 솔루션 탐지 시 앱 종료뿐 아니라 서버 측 경보와 로깅을 수행해 우회 시도를 추적하고, 서버 입력 검증 체계를 강화한다.
예상 결과물 및 상태		Docker 컨테이너 쉘 확보, 서버 내부 매핑 가능 상태

단계 번호		4
단계 명		컨테이너 정찰 후 호스트 서버 탈출
주요 목표		Docker 컨테이너 환경 분석, runc 취약점 이용한 호스트 루트 권한 획득
유형	공격	컨테이너 정찰, CVE-2024-21626 runc Exploit, Cron 백도어 설치
	방어	runc 최신화, Seccomp/AppArmor 정책 강화
주요 활동		공격자는 확보한 컨테이너 쉘에서 프로세스/마운트/네트워크 메타데이터를 수집해 컨테이너 격리 수준과 런타임 구성을 파악한다. 호스트 인터페이스로의 우회 경로나 런타임 취약점 계열(runc 등) 존재 여부를 점검한 뒤, 컨테이너 경계에서 호스트 파일시스템/실행 경로에 대한 비정상 접근 징후를 체계적으로 검증한다. 유효한 탈출 경로가 확인되면, 지속성 확보(예: 예약 실행 경로 악용), 권한 승격, 로깅 회피까지 고려한 순서로 호스트 맥락 제어권을 획득하고, 챗봇 서버의 소스/설정/비밀정보를 열람해 후속 피봇 경로를 도출한다. 방어자는 컨테이너 런타임을 최신 버전으로 유지하고, 접근 제어 정책을 적용해 컨테이너-호스트 경계를 강화한다. 정기 취약점 점검과 경보 시스템을 운영해야 한다.
주요 사용 도구 및 기술		runc Exploit, Bash, Cron
예상 결과물 및 상태		호스트 서버 루트 쉘 확보, 서버 전면 제어

단계 번호		5
단계 명		모바일 서버로의 피봇 및 내부망 확산
주요 목표		챗봇-모바일 서버 간 신뢰 경로 악용, 모바일 서버 쉘 확보
유형	**공격**	서버 간 코드 전달 로직 악용, 역방향 세션 확보
	방어	Mutual TLS, API 접근 제어 강화
주요 활동		공격자는 챗봇-모바일 서버 간 내부 신뢰 통신 흐름과 오류 처리/코드 전달 로직을 분석해, 모바일 서버 측에 코드 실행성 있는 입력이 전달되도록 요청 구성을 교란한다. 서버 간 인증·검증이 허술한 구간을 통과시켜 모바일 서버에서도 역방향 세션을 성립시키고, 접속 후 서비스 구성·자격증명·아티팩트 저장 위치를 체계적으로 수색한다. 이 과정에서 SSH 키·액세스 토큰·백업 경로 정보 등 추가 권한 확장에 유리한 비밀 정보를 수집하고, 내부망 서비스/포트 지형을 스캔·매핑해 다음 단계(백업/핵심 서버) 공략 루트를 확정한다. 방어자는 서버 간 인증, API 요청 검증, 실시간 트래픽 모니터링을 통해 서버 간 코드 전달 악용을 탐지해야 한다.
주요 사용 도구 및 기술		Curl, Nmap, Python Reverse Shell
예상 결과물 및 상태		모바일 서버 쉘 확보, 내부망 경로 식별

단계 번호		6
단계 명		백업 서버 권한 장악 및 AWS 키 탈취
주요 목표		SSH 키 및 Jenkins 취약점 악용, AWS Access Key 확보, 클라우드 자산 접근
유형	**공격**	SSH 키 재사용, Jenkins CLI 취약점, 클라우드 키 확보
	방어	키 중앙화(KMS), Jenkins 보안 강화, IAM 접근 통제
주요 활동		공격자는 모바일 서버에서 확보한 키/자격증명을 활용해 백업 서버에 합법적 세션처럼 진입한다. 백업 서버가 다수 시스템의 인증/아카이브 허브 역할을 한다는 구조를 확인하고, 여기에 축적된 백업 데이터·스크립트·추가 키/경로를 분석해 조직 전반의 자산 지도를 완성한다. 이어 핵심 워크로드(웹/모바일/데이터 계층)로 접근권을 체계적으로 확장하면서, 향후 대규모 파급을 위한 자동화 스크립트/작업 스케줄을 준비한다. 이로써 방어 측 복구 동맥에 해당하는 백업 경로까지 공격하게 된다. 방어자는 SSH 키 재사용을 금지하고 KMS로 키를 중앙 관리한다. Jenkins 서버는 최신화 및 접근 제어 강화가 필요하며, 클라우드 자격 증명은 IAM 역할 기반으로 보호해야 한다.
주요 사용 도구 및 기술		SSH, Jenkins Exploit, AWS CLI
예상 결과물 및 상태		백업 서버 권한 확보, AWS 키 탈취, S3 자산 접근 가능

단계 번호		7
단계 명		랜섬웨어 동시 배포로 서비스·백업 마비
주요 목표		백업 서버부터 전체 서비스 암호화, 복구 경로 제거
유형	공격	AES+RSA 암호화, Base64 페이로드, SSH 자동화
	방어	오프라인 백업, EDR 탐지, DR 계획
주요 활동		공격자는 우선순위를 백업→핵심 서비스 순으로 설정해, 복구 수단을 선(先) 차단한 뒤 서비스 계층을 동시 암호화하는 전략을 집행한다. 전송 경로 제약을 고려해 컨텐츠 인코딩/분할 전송·현지화(현지 빌드/스크립트) 등을 활용해 페이로드를 배치하고, 대량 파일 접근·암호화 패턴이 탐지 임계값에 걸리기 전에 처리량과 속도를 조절하며 진행한다. 암호화 완료 후에는 사용자-facing 채널로 감염 사실을 공지(랜섬 노트)하고, 로그·스케줄·임시 아티팩트를 정리해 포렌식 가능성을 저하시키는 선에서 흔적 최소화를 수행한다. 결과적으로 서비스 가용성과 복구 경로가 동시에 붕괴해 조직 운영 전반이 마비된다. 방어자는 다계층 백업 체계와 실시간 EDR 탐지로 암호화 공격을 조기 차단하고, DR 복구 계획으로 피해 최소화 전략을 수립해야 한다.
주요 사용 도구 및 기술		OpenSSH, Base64, AES/RSA 랜섬웨어
예상 결과물 및 상태		전체 인프라 암호화, 서비스 및 복구 경로 동시 마비