<붙임 2> 시나리오 제출양식

시나리오 제목

ATHENA 2025 — TeamCity 인증우회 기반 CI/CD 공급망 오염 모의훈련

시나리오 단계

Phase 0: 환경 준비 — 패키지 자동 탐지·복사·진입
Phase 1: 패키지 루트 자동 찾기 → 해당 폴더로 이동
Phase 2: Docker/Compose 점검·설치
Phase 3: 스택 기동 — proxy / teamcity-mock / hny-collector / siem-lite 빌드·실행
Phase 4: 헬스체크 — 8080/8088/9090 엔드포인트 확인
Phase 5: 공격 A1-A5 — 세션 생성 → 셰도우 어드민 계정 생성 → 권한 부여 → 빌드 정의 변경 → 러너 3대 등록
Phase 6: 수집기 확립 — hny-collector 재빌드·재시작, 프록시 경로 확인
Phase 7: 탐지 유발 — 허니토큰 1회 트리거, SIEM 60초 디듀프 동작 확인
Phase 8: 산출물 패키징 — 로그·SVG·요약 묶어 tar.gz 생성, SHA256 출력

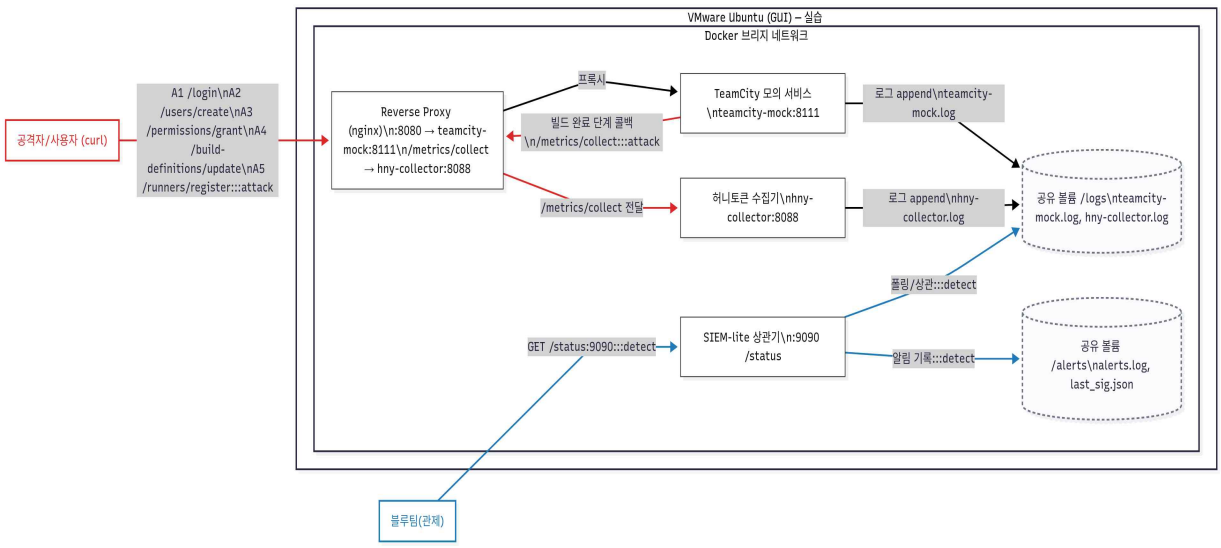
시나리오 목적

CI/CD 인증우회/권한오남용을 모의하고, 허니토큰/SIEM 상관분석으로 탐지/격리/보고 역량을 학습

시나리오 줄거리

공격자(레드팀)는 리버스 프록시 뒤 TeamCity 유사 서비스에 접근해 세션을 만들고, 셰도우 어드민을 생성해 프로젝트 권한을 획득한다. 이어 빌드 정의에 오염 스텝을 주입하고 러너를 등록해 공급망 오염 경로를 마련한다. 빌드 완료 단계에서 발생한 허니토큰 콜백은 수집기로 기록되고, SIEM은 [session → user_create → perm_grant → builddef_update + honeytoken] 연쇄를 상관 분석하여 TC_SUPPLYCHAIN_SIM 경보를 생성한다.
(방어자)블루팀은 /status로 상태를 점검하고, 로그·시각화를 포함한 산출물을 제출한다.

시나리오 네트워크 구성도



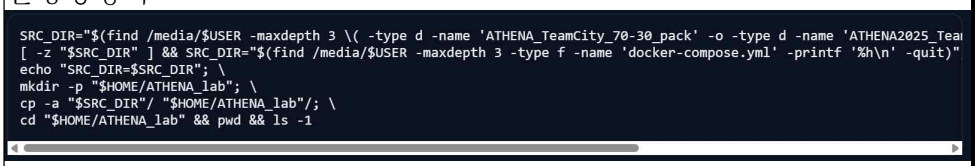
<table>
<tr><td colspan="2">단계 번호</td><td>Phase 0</td></tr>
<tr><td colspan="2">단계 명</td><td>환경 준비 — 실습 패키지 자동 탐지·복사·진입</td></tr>
<tr><td colspan="2">주요 목표</td><td>USB(또는 외부 미디어)에서 실습 패키지를 자동 탐지해 ~/ATHENA_lab로 복사하고, 패키지 루트로 진입하여 핵심 파일(README, docker-compose.yml) 존재를 검증한다.</td></tr>
<tr><td rowspan="2">유형</td><td>공격</td><td>해당 없음</td></tr>
<tr><td>방어</td><td>해당 없음</td></tr>
<tr><td colspan="2">주요 활동</td><td>I. 외부 미디어에서 패키지 위치 자동 탐지

II. 실습 작업 디렉터리(~/ATHENA_lab) 생성 및 전체 복사

III. 패키지 루트 이동 및 파일 목록 확인

IV. 성공/실패 기준 즉시 판정

완료 기준(판정 체크리스트)
I. 출력에 SRC_DIR=/media/... 와 경로가 정상 표시된다.
II. 현재 위치가 ~/ATHENA_lab로 변경되며 파일 목록에 아래 항목이 보인다:
-> docker-compose.yml
-> README.md (또는 동급 안내 문서)</td></tr>
<tr><td colspan="2">주요 사용 도구 및 기술</td><td>I. Bash, find, cp, ls
II. 경로/권한 점검(기본 리눅스 파일시스템 조작)

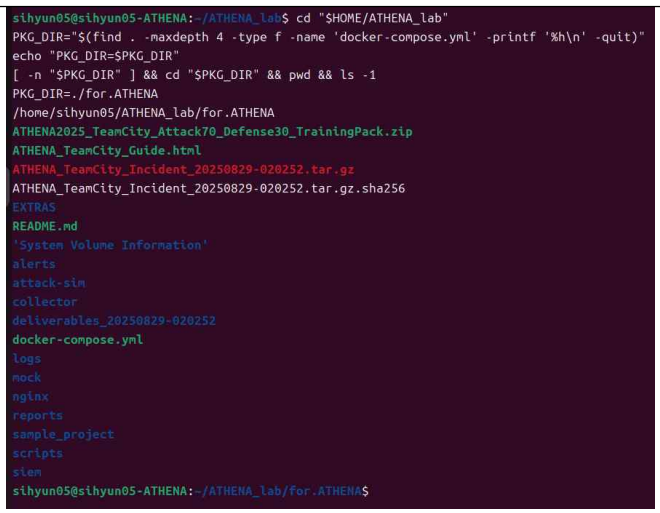
실행명령어:
  </td></tr>
<tr><td colspan="2">예상 결과물 및 상태</td><td> 
I. ~/ATHENA_lab/ 디렉터리 생성 및 실습 패키지 전체 복제됨
II. 핵심 파일 존재 확인(스크린샷 가능)
III. 다음 단계(도커/컴포즈 점검)로 진행 가능 상태</td></tr>
</table>

<table>
<tr><td colspan="2">단계 번호</td><td>Phase 1</td></tr>
<tr><td colspan="2">단계 명</td><td>패키지 루트 자동 찾기 → 해당 폴더로 진입</td></tr>
<tr><td colspan="2">주요 목표</td><td>~/ATHENA_lab 내에서 docker-compose.yml이 있는 패키지 루트를 자동 탐지하고, 그 디렉터리로 이동하여 핵심 파일 존재를 확인한다.</td></tr>
<tr><td rowspan="2">유형</td><td>공격</td><td>해당 없음</td></tr>
<tr><td>방어</td><td>해당 없음</td></tr>
<tr><td colspan="2">주요 활동</td><td>I. docker-compose.yml 위치 자동 검색

II. 패키지 루트로 디렉터리 변경

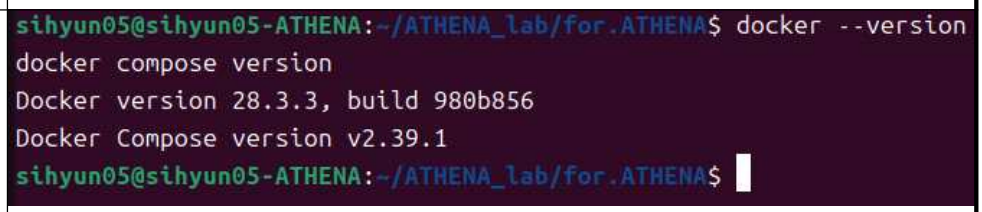
III. 파일 목록 확인 및 성공/실패 판정

완료 기준(판정 체크리스트)
I. 출력에 PKG_DIR=./... 경로가 표시된다.
II. 현재 경로가 패키지 루트로 바뀌고 파일 목록에 아래가 보인다:
docker-compose.yml
README.md(또는 동급 안내 문서)</td></tr>
<tr><td colspan="2">주요 사용 도구 및 기술</td><td>I. Bash, find, cd, ls
II. 경로 처리/표준 출력 확인

실행명령어:
<pre>cd "$HOME/ATHENA_lab"
PKG_DIR="$(find . -maxdepth 4 -type f -name 'docker-compose.yml' -printf '%h\n' -quit)"
echo "PKG_DIR=$PKG_DIR"
[-n "$PKG_DIR"] && cd "$PKG_DIR" && pwd && ls -1</pre></td></tr>
<tr><td colspan="2">예상 결과물 및 상태</td><td> 

I. 패키지 루트 경로 확인 및 진입 완료
II. 다음 단계(도커/컴포즈 점검 및 설치) 진행 가능 상태</td></tr>
</table>

<table>
<tr><td colspan="2">단계 번호</td><td>Phase 2</td></tr>
<tr><td colspan="2">단계 명</td><td>Docker & Docker Compose 점검/설치</td></tr>
<tr><td colspan="2">주요 목표</td><td>도커 엔진과 도커 컴포즈(플러그인)가 정상 동작하도록 설치/검증한다.</td></tr>
<tr><td rowspan="2">유형</td><td>공격</td><td>해당 없음</td></tr>
<tr><td>방어</td><td>해당 없음</td></tr>
<tr><td colspan="2">주요 활동</td><td>I. 버전 확인으로 설치 여부 판정
II. 미설치 시 공식 저장소 추가 후 설치
III. 서비스 활성화 및 권한(docker 그룹) 설정

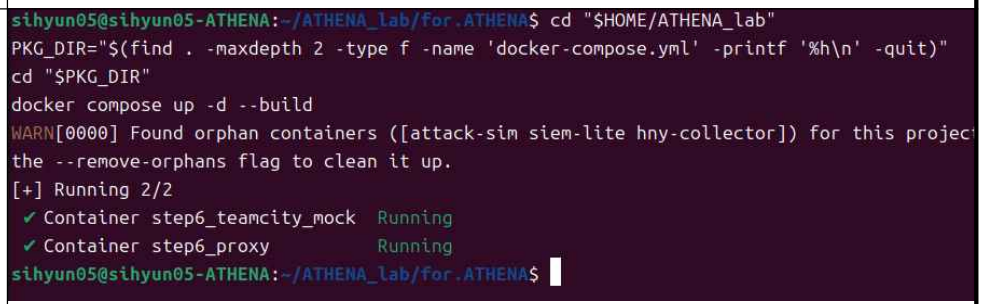
완료 기준(판정 체크리스트)
I. 버전 문자열이 출력되면 성공.</td></tr>
<tr><td colspan="2">주요 사용 도구 및 기술</td><td>I. Bash, APT, systemd
II. Docker CE, Docker Compose v2(plugin)</td></tr>
<tr><td colspan="2">예상 결과물 및 상태</td><td> 

I. Docker/Compose 최신 버전 설치 및 동작 확인
II. 현재 사용자로 sudo 없이 docker/docker compose 실행 가능
III. 다음 단계(스택 기동) 수행 준비 완료</td></tr>
</table>

<table>
<tr><td colspan="2">단계 번호</td><td>Phase 3</td></tr>
<tr><td colspan="2">단계 명</td><td>스택 기동 — 컨테이너 빌드/실행</td></tr>
<tr><td colspan="2">주요 목표</td><td>패키지의 Docker Compose 스택(reverse-proxy, teamcity-mock, hny-collector, siem-lite)을 빌드 후 백그라운드로 기동한다.</td></tr>
<tr><td rowspan="2">유형</td><td>공격</td><td>해당 없음</td></tr>
<tr><td>방어</td><td>해당 없음</td></tr>
<tr><td colspan="2">주요 활동</td><td>I. 패키지 루트에서 Compose 스택 빌드/실행
II. 컨테이너 상태 확인(Up / Running)
III. 권한·포트 이슈 즉시 조치

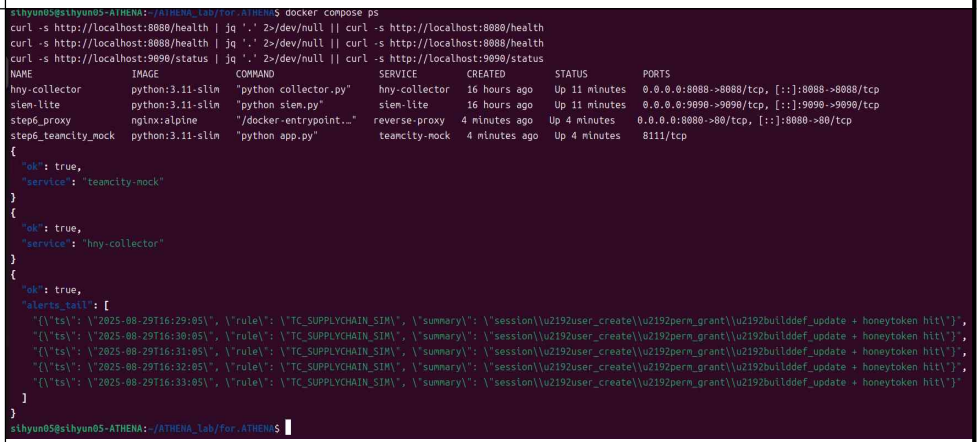
완료 기준(판정 체크리스트)
I. docker compose ps에 다음 4개 서비스가 Up/Running 으로 표시:
proxy / teamcity-mock / hny-collector / siem-lite

II. 네트워크가 생성되고 재시작 없이 안정적으로 유지</td></tr>
<tr><td colspan="2">주요 사용 도구 및 기술</td><td>I. Docker, Docker Compose v2
II. 리눅스 기본 네트워킹(포트 사용 여부 점검)

실행명령어:
  </td></tr>
<tr><td colspan="2">예상 결과물 및 상태</td><td> 
I. 4개 컨테이너가 정상 기동(Up)
II. 이후 Phase 4(헬스체크) 에서 HTTP 엔드포인트 응답 확인 가능 상태</td></tr>
</table>

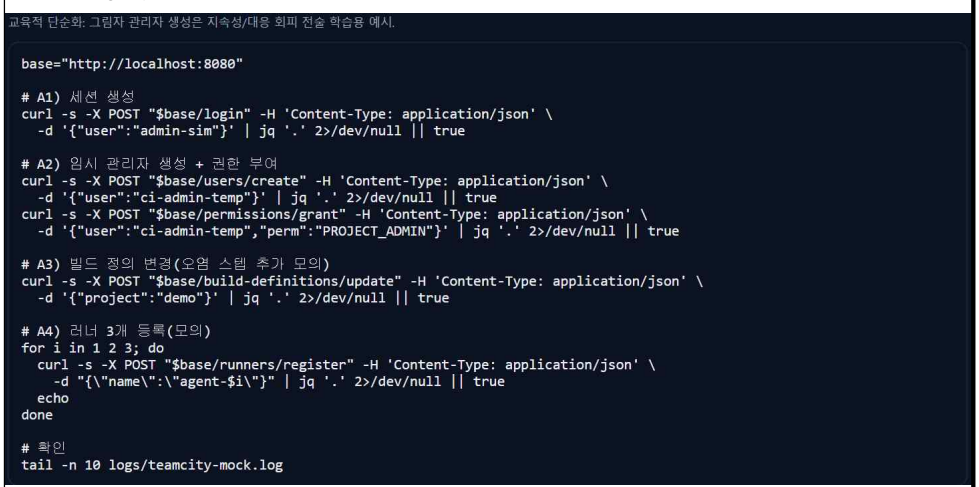
<table>
<tr><td colspan="2">단계 번호</td><td>Phase 4</td></tr>
<tr><td colspan="2">단계 명</td><td>헬스체크 — 핵심 서비스 상태 검증</td></tr>
<tr><td colspan="2">주요 목표</td><td>기동된 4개 서비스가 정상 응답하는지 HTTP 엔드포인트로 확인하고, 이후 단계(공격/탐지 시퀀스)로 넘어갈 수 있는지 판정한다.</td></tr>
<tr><td rowspan="2">유형</td><td>공격</td><td>해당 없음</td></tr>
<tr><td>방어</td><td>해당 없음(운영 점검)</td></tr>
<tr><td colspan="2">주요 활동</td><td>I. 컨테이너 상태(Up/Ports) 확인
II. 프록시(8080), 수집기(8088), SIEM(9090) 헬스 엔드포인트 호출
III. 성공/실패 기준으로 즉시 판정

완료 기준(판정 체크리스트)
I. 위 3개 엔드포인트가 모두 200 OK + JSON 응답을 반환 → 성공</td></tr>
<tr><td colspan="2">주요 사용 도구 및 기술</td><td>I. docker compose ps / logs
II. url 로컬 HTTP 점검

실행명령어:
<pre>docker compose ps
curl -s http://localhost:8080/health | jq '.' 2>/dev/null || curl -s http://localhost:8080/health
curl -s http://localhost:8088/health | jq '.' 2>/dev/null || curl -s http://localhost:8088/health
curl -s http://localhost:9090/status | jq '.' 2>/dev/null || curl -s http://localhost:9090/status</pre></td></tr>
<tr><td colspan="2">예상 결과물 및 상태</td><td> 
I. 프록시/수집기/SIEM의 헬스 엔드포인트가 정상 응답
II. 실습 스택이 공격·탐지 시퀀스 실행 준비 완료</td></tr>
</table>

<table>
<tr><td colspan="2">단계 번호</td><td>Phase 5</td></tr>
<tr><td colspan="2">단계 명</td><td>공격 시퀀스 실행(실행명령어 A1~A5) — 로그인 → 계정 생성 → 권한 부여 → 빌드 정의 변경 → 러너 등록</td></tr>
<tr><td colspan="2">주요 목표</td><td>프록시(8080)를 통해 TeamCity 모의 서비스에 일련의 관리 API를 호출하여 공격 체인을 재현하고, 로그에 단계별 이벤트가 올바른 순서로 기록되는지 확인한다.</td></tr>
<tr><td rowspan="2">유형</td><td>공격</td><td>정찰/취약점 악용(사전 인증 우회 가정), 권한 남용, 공급망 오염(빌드 정의 변경)</td></tr>
<tr><td>방어</td><td>해당 없음(탐지는 다음 Phase에서 집계)</td></tr>
<tr><td colspan="2">주요 활동</td><td>I. 세션 생성(/login)
II. 임시 관리자 생성(/users/create)
III. 프로젝트 관리자 권한 부여(/permissions/grant)
IV. 빌드 정의 오염 단계 추가 모의(/build-definitions/update)
V. 빌드 러너 3대 등록 모의(/runners/register)
VI. 결과 로그 확인(logs/teamcity-mock.log)

완료 기준(판정 체크리스트)
I. logs/teamcity-mock.log 마지막 부분에 아래 순서가 나타나면 성공:
session_created → user_create → perm_grant → builddef_update → runner_register×3</td></tr>
<tr><td colspan="2">주요 사용 도구 및 기술</td><td>I. curl(HTTP JSON POST)
II. Nginx 리버스 프록시 경유(localhost:8080 → teamcity-mock:8111
II. JSON 본문/HTTP 상태 확인

실행명령어:
교육적 단순화: 그림자 관리자 생성은 지속성/대응 회피 전술 학습용 예시.
 
<pre>
base="http://localhost:8080"

A1) 세션 생성
curl -s -X POST "$base/login" -H 'Content-Type: application/json' \
  -d '{"user":"admin-sim"}' | jq '.' 2>/dev/null || true

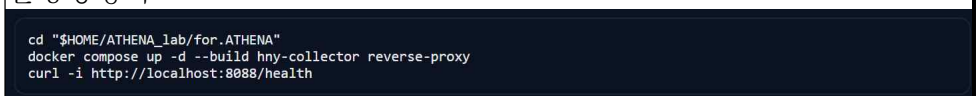
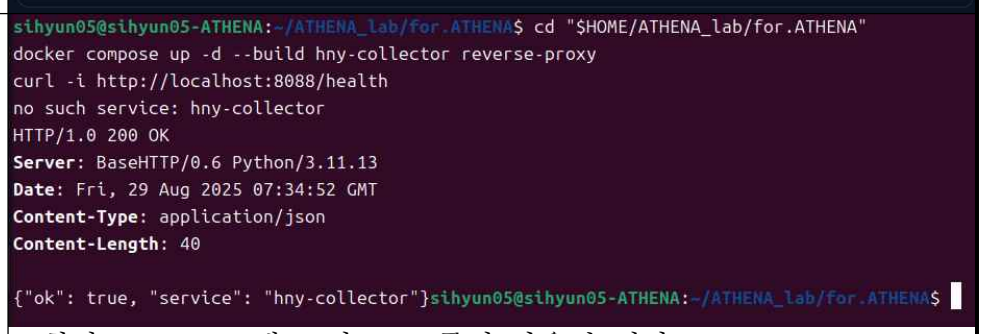
A2) 임시 관리자 생성 + 권한 부여
curl -s -X POST "$base/users/create" -H 'Content-Type: application/json' \
  -d '{"user":"ci-admin-temp"}' | jq '.' 2>/dev/null || true
curl -s -X POST "$base/permissions/grant" -H 'Content-Type: application/json' \
  -d '{"user":"ci-admin-temp","perm":"PROJECT_ADMIN"}' | jq '.' 2>/dev/null || true

A3) 빌드 정의 변경(오염 스텝 추가 모의)
curl -s -X POST "$base/build-definitions/update" -H 'Content-Type: application/json' \
  -d '{"project":"demo"}' | jq '.' 2>/dev/null || true

# A4) 러너 3개 등록(모의)
for i in 1 2 3; do
  curl -s -X POST "$base/runners/register" -H 'Content-Type: application/json' \
    -d "{\"name\":\"agent-$i\"}" | jq '.' 2>/dev/null || true
  echo
done

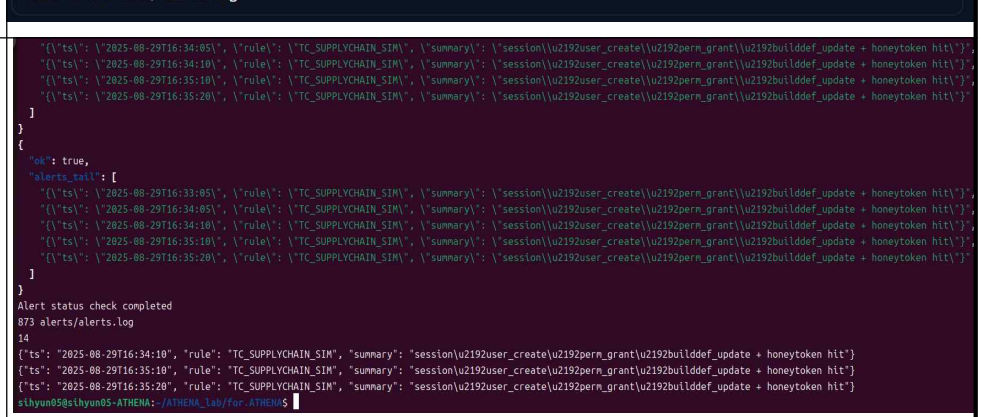
확인
tail -n 10 logs/teamcity-mock.log
</pre></td></tr>
</table>

예상 결과물 및 상태	 I. logs/teamcity-mock.log에 공격 체인의 이벤트가 순서대로 기록됨 II. 다음 단계(Phase 6)에서 허니토큰 수집기/프록시 상태 확인 및 트리거로 탐지 상관을 유도할 준비 완료

단계 번호		Phase 6
단계 명		허니토큰 수집기 교체·재기동 → 프록시/수집기 헬스체크
주요 목표		업데이트된 hny-collector 이미지를 재빌드·재시작하고, reverse-proxy(8080) 와 collector(8088) 헬스 엔드포인트가 정상인지 확인한다. (다음 단계에서 허니토큰 트리거/상관 확인 예정)
유형	**공격**	해당 없음
	방어	서비스 재배포·헬스 검증
주요 활동		I. 대상 서비스만 선택 빌드/재시작 (hny-collector, reverse-proxy) II. /health 응답 상태 확인(200 OK + JSON) III. 라우팅/업스트림 해석 오류 시 즉시 조치 완료 기준(판정 체크리스트) I. 두 엔드포인트가 모두 200 OK + JSON 응답 → 성공, Phase 7로 진행
주요 사용 도구 및 기술		I. Docker Compose 부분 배포(up -d --build <services...>) II. curl 헬스체크, Nginx 업스트림 확인 실행명령어: 
예상 결과물 및 상태		 I. 최신 collector 배포 완료, 프록시 라우팅 정상 II. 다음 단계(Phase 7)에서 허니토큰 트리거 → SIEM 스로틀/상관 확인 수행 준비 완료

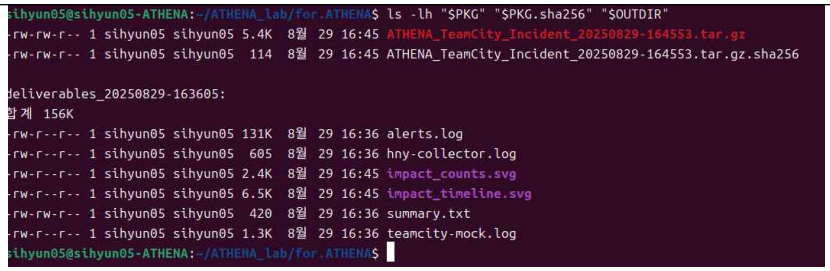
<table>
<tr><td colspan="2">단계 번호</td><td>Phase 7</td></tr>
<tr><td colspan="2">단계 명</td><td>허니토큰 1회 트리거 → SIEM 60초 디듀프(스로틀) 확인</td></tr>
<tr><td colspan="2">주요 목표</td><td>빌드 완료 콜백을 교육용 수동 트리거로 재현하여 허니토큰 수집기에 이벤트를 기록하고, SIEM가 동일 시그니처를 60초 동안 1회만 경보로 집계(디듀프)하는지 확인한다.</td></tr>
<tr><td rowspan="2">유형</td><td>공격</td><td>공급망 오염 후 메트릭 전송 모의</td></tr>
<tr><td>방어</td><td>허니토큰 수집 → SIEM 상관/디듀프 확인</td></tr>
<tr><td colspan="2">주요 활동</td><td>I. 허니토큰 수집기로 1회 콜백 전송
II. SIEM 상태 3회 조회(즉시/10초/20초)
III. 경보 로그 증가량 및 최근 라인 확인으로 스로틀 동작 검증

완료 기준(판정 체크리스트)
I. alerts/alerts.log 줄 수가 정확히 1줄 증가한다.
II. 이후 60초 이내 반복 조회에서도 추가 증가는 없다(동일 시그니처 디듀프 적용).
III. /status 응답의 alerts_tail 내용이 안정적으로 유지되거나, 한 번만 갱신된다.</td></tr>
<tr><td colspan="2">주요 사용 도구 및 기술</td><td>I. curl (HTTP POST/GET)
II. SIEM-lite /status, 경보 로그(alerts/alerts.log) 확인

사용명령어:
  </td></tr>
<tr><td colspan="2">예상 결과물 및 상태</td><td> 
I. logs/hny-collector.log에 허니토큰 수집 기록 1건 추가
II. alerts/alerts.log에 1건의 신규 경보만 추가(60초 내 중복 억제)
III. 다음 단계(Phase 8)에서 산출물 패키징 및 시각화(SVG) 포함 준비 완료</td></tr>
</table>

<table>
<tr><td colspan="2">단계 번호</td><td>Phase 8</td></tr>
<tr><td colspan="2">단계 명</td><td>산출물 패키징(로그 + 시각화 SVG + 요약) 생성</td></tr>
<tr><td colspan="2">주요 목표</td><td>Phase 6~7에서 발생한 로그/경보와 시각화 결과(타임라인·카운트 SVG)를 한 폴더로 모아 제출용 .tar.gz + SHA256 를 만든다.</td></tr>
<tr><td rowspan="2">유형</td><td>공격</td><td>해당 없음</td></tr>
<tr><td>방어</td><td>증적 수집, 보고 아티팩트 생성</td></tr>
<tr><td colspan="2">주요 활동</td><td>I. 컨테이너 내부 데이터로 SVG 2종 생성(표준 라이브러리 기반)
II. logs/*, alerts/*, summary.txt 와 함께 패키징
III. 무결성 확인을 위한 *.sha256 생성

완료 기준(판정 체크리스트)
I. OUTDIR=deliverables_YYYYMMDD-HHMMSS/ 가 생성되고 내부에 logs 3종 + summary.txt + impact_*.svg(2종) 존재
II. 루트에 ATHENA_TeamCity_Incident_*.tar.gz 와 같은 이름의 tar.gz + 동명 .sha256 가 생성
III. ls -lh 출력으로 파일 크기 및 타임스탬프가 확인됨</td></tr>
<tr><td colspan="2">주요 사용 도구 및 기술</td><td>I. docker compose exec … python(컨테이너 내 SVG 생성)
II. tar, sha256sum

사용명령어:
  </td></tr>
<tr><td colspan="2">예상 결과물 및 상태</td><td> 
I. 제출용 패키지(.tar.gz)와 무결성 파일(.sha256) 생성 완료
II. deliverables_* 폴더에 로그/경보/SVG/요약 모음 완료</td></tr>
</table>

```
cd "$HOME/ATHENA_lab/for.ATHENA"

OUTDIR="deliverables_$(date +%Y%m%d-%H%M%S)"
mkdir -p "$OUTDIR"

# 로그/경보 수집
for f in logs/teamcity-mock.log logs/hny-collector.log alerts/alerts.log; do
  [ -f "$f" ] && cp -f "$f" "$OUTDIR"/
done

# (이 가이드 포함)
[ -f ATHENA_TeamCity_Lab_Guide_ko_dark_v6.html ] && cp -f ATHENA_TeamCity_Lab_Guide_ko_dark_v6.html "$OUTDIR"/

# 요약
ALERTS_COUNT=$( [ -f alerts/alerts.log ] && wc -l < alerts/alerts.log || echo 0 )
HNY_COUNT=$( [ -f logs/hny-collector.log ] && wc -l < logs/hny-collector.log || echo 0 )
cat > "$OUTDIR/summary.txt" <<'EOF'
ATHENA 2025 — TeamCity Auth-Bypass → Secret Theft → Supply-Chain Sim
Environment & Timeline (UTC)
- Sequence: session_created → user_create → perm_grant → builddef_update → (A5) honeytoken
- SIEM alert dedup: 60s (alerts only)
Artifacts:
- teamcity-mock.log
- hny-collector.log
- alerts.log
- ATHENA_TeamCity_Lab_Guide_ko_dark_v6.html
EOF
date -u +'Generated at: %Y-%m-%dT%H:%M:%SZ' >> "$OUTDIR/summary.txt"
echo "Alert lines: $ALERTS_COUNT" >> "$OUTDIR/summary.txt"
echo "Collector lines: $HNY_COUNT" >> "$OUTDIR/summary.txt"

PKG="ATHENA_TeamCity_Incident_$(date +%Y%m%d-%H%M%S).tar.gz"
tar -czf "$PKG" "$OUTDIR"
sha256sum "$PKG" | tee "$PKG.sha256"
ls -lh "$PKG" "$PKG.sha256" "$OUTDIR"
```

```
sihyun05@sihyun05-ATHENA:~/ATHENA_lab/for.ATHENA$ ls -lh "$PKG" "$PKG.sha256" "$OUTDIR"
-rw-rw-r-- 1 sihyun05 sihyun05 5.4K  8월 29 16:45 ATHENA_TeamCity_Incident_20250829-164553.tar.gz
-rw-rw-r-- 1 sihyun05 sihyun05  114  8월 29 16:45 ATHENA_TeamCity_Incident_20250829-164553.tar.gz.sha256

deliverables_20250829-163605:
합계 156K
-rw-r--r-- 1 sihyun05 sihyun05 131K  8월 29 16:36 alerts.log
-rw-r--r-- 1 sihyun05 sihyun05  605  8월 29 16:36 hny-collector.log
-rw-r--r-- 1 sihyun05 sihyun05 2.4K  8월 29 16:45 impact_counts.svg
-rw-r--r-- 1 sihyun05 sihyun05 6.5K  8월 29 16:45 impact_timeline.svg
-rw-rw-r-- 1 sihyun05 sihyun05  420  8월 29 16:36 summary.txt
-rw-r--r-- 1 sihyun05 sihyun05 1.3K  8월 29 16:36 teamcity-mock.log
sihyun05@sihyun05-ATHENA:~/ATHENA_lab/for.ATHENA$
```