**6. 관제 및 대응**

**6.1 운용 정책**

|  |  |  |
| --- | --- | --- |
| **구분** | **기준** | **수행방안** |
| **기본 운용** | 2026년 주요정보통신  기반시설 수행가이드 | - 자동화 코드로 점검  - 부서별 취합 검검 |
| **시나리오 기반 운용** | 패킷분석(정상/비정상패킷)  로그분석(로그발생 이벤트) | - 이벤트 발생시 정책 보완 적용 |

**6.2 운용 흐름**

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **단계** | **구분** | **주요 활동 내용** | **목표** |
| 1단계 | 정책 수립 및 자동화 설계 | • 2026년 주정통 가이드 기반 표준 보안 항목 선정  • 기술 진단 항목의 분배 및 자동화 점검 | 점검 표준 가이드라인,  자동 진단 코드 (Code) |
| 2단계 | 위협 분석 및 대응 전략 | • 레드팀 침투 시뮬레이션을 통한 잠재 공격 경로 식별  • 시나리오별 맞춤형 탐지/차단 시그니처 도출  • 침투 테스트 결과 공유 및 부서별 보안 정책 문서화 | 침투 테스트 결과 보고서,  부서별 대응 전략서 |
| 3단계 | 계층별 보안 정책 이행 | [서버] 호스트 기반 차단(iptables) 및 로그 발생 경로 최적화  [네트워크] 토폴로지 분석을 통한 최적 위치 ACL 적용  [시나리오 기반] 정상/비정상 패킷 식별 기준 실무 적용 | 인프라 보안 설정 강화,  패킷/로그 분석 증적 |
| 4단계 | 방어 유효성 검증 | • 적용된 보안 정책 기반의 공격 재연을 통한 탐지 성능 확인  • 서버·네트워크 접점에서의 실질 차단 여부 로그 검증  • 방어 유효성 평가를 통한 최종 보안 표준 수립 | 방어 유효성 검증 결과,  최종 방어 기준점 정의 |
| 5단계 | 지속적 방어 체계 자산화 | • 검증된 탐지 규칙을 상시 관제 정책으로 전환  • 로그 경로, 패킷 식별 정보 등 대응 지식베이스 구축  • 정기 점검 및 사고 대응을 위한 보안 데이터 내재화 | 상시 관제 탐지 룰 (Rule),  보안 위협 지식베이스 DB |

**6.3 보안 기술리스트**

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **기능** | **주요 기술 요소** | **상세 설명** |
| **관제** | 통합 보안 관제 | Security Onion | IDS, 로그 관리, 트래픽 분석 기능이 통합된 보안 관제 플랫폼 |
| 침입 탐지 | Snort | 네트워크 패킷 분석 및 공격 패턴 매칭을 통한 실시간 침입 탐지 |
| 이벤트 분석 | Sguil | 탐지된 보안 이벤트의 가시성 확보 및 정밀 분석용 인터페이스 |
| 패킷 분석 | Wireshark | 네트워크 트래픽 상세 분석을 통한 공격 증거 및 통신 경로 검증 |
| **자동화** | 대응 자동화 | Rule 정책 | 공격 시나리오별 탐지를 위한 룰 규칙 자동화 코드 구현 |

**6.4 기술 활용 계획**

|  |  |  |
| --- | --- | --- |
| no. | 사용도구 | 내용 |
| 1 | Security Onion | - Security Onion을 활용한 실시간 통합 관제 체계 수립 및 시각화  - 트래픽 미러링 프로브 및 중앙 로그 수집 서버의 고가용성(HA) 구성 방안 |
| 2 | Snort | Snort 기반의 지능형 침입 탐지 룰셋(Rule-set) 설계 및 튜닝 |
| Sguil | Sguil 연동을 통한 실시간 위협 가시화 및 심층 분석 프로세스 수립 |
| 3 | Wireshark | - Deep Packet Inspection(DPI) 기반의 위협 트래픽 심층 분석 방안  -공격 단계별 네트워크 증적 추출 및 시각화 |
| 4 | python | - 공격 시나리오별 맞춤형 탐지 규칙(Rule-set) 자동화 구현  - 탐지-대응 시퀀스 최적화를 통한 오탐(False Positive) 최소화 |

**6.5 운용상세**

**6.5.1 로그분석 설계**

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 분석 항목 | 주요 설계 및  프로세스 내용 | 활용 방안 |
| **공격 자산 및**  **도구 식별** | **행위 로그 추출** | • Kali, Hydra, SQLmap 등 공격 도구 실행 시 발생하는 자체 로그 분석  • 도구별 고유한 공격 패턴 식별 (User-Agent, 리퀘스트 간격 등) | 탐지 시그니처 및 블랙리스트 기반 마련 |
| **추적 데이터 정의** | • 공격자 IP/Port 정보 및 세션 유지 시간 기록 프로세스 수립  • HTTP Request 파라미터(SQLi, LFI 등) 내 특수문자 및 구문 기록 | 공격 근거 확보 및 공격자 프로파일링 |
| **타겟 시스템 별 핵심 로그** | **인프라 로그** | • [Linux/Win] 시나리오 1(SSH Brute Force) 대응을 위한 auth.log,  Security Event Log 집중 수집 | 계정 탈취 및 비인가 접근 탐지 |
| **서비스 로그** | • [Web/DB] 시나리오 2, 3(XSS, CSRF, LFI) 분석을 위한  Access/ Error log 및 DB Audit log 확보 | 웹 취약점 공격 및 데이터 변조 감시 |

**6.5.2 패킷분석**

- Snort & Sguil: 발생된 경보(Alert)와 실제 패킷 데이터 간의 상관관계 분석

- Wireshark: 시나리오 1(ARP Spoofing) 및 시나리오 3(Reverse Shell)의 L7 페이로드 내 공격 구문 심층 식별

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **공격** **단계** | **공격 기법** | **필터 (Wireshark)** | **정상 패킷 패턴** | **비정상(공격)**  **패킷 패턴** | **정상/비정상**  **식별 기준** |
| **정보 수집** | dnsenum | dns.flags.opcode == 5 | 특정 도메인에 대한 단일 질의 | Standard query AXFR 요청 패킷 발생 | 응답 패킷 내 모든 존(Zone) 레코드 포함 여부 |
| **정보 수집** | dig | dns.flags.response == 0 | 일반적인 도메인 이름 해석 요청 | 특정 도메인 반복 쿼리 및 버전 정보 요청 | 동일 도메인 반복 질의 빈도 및 DNS 버전 정보 수집 패턴 |
| **정보 수집** | Nmap | tcp.flags.syn == 1 | 서비스 제공 포트로의 단일 접속 | 수천 개의 포트로 SYN 패킷 대량 전송 | 단시간 내 다수 포트(Multi-port) 접속 시도 |
| **정보수집** | Nmap 스캔 | tcp.flags.syn == 1 | 특정 포트 단일 접속 | 수천 개의 포트로 SYN 전송 | 단시간 내 다수 포트 접속 시도 |
| **망 확장** | DNS Zone Transfer | dns.flags.opcode == 5 | 특정 도메인 단일 질의 | Standard query AXFR 요청 | 응답 내 모든 레코드 포함 여부 |
| **권한 탈취** | SSH Brute Force | tcp.port == 22 | 간헐적 원격 접속 | 초당 수십 회 TCP 연결/해제 | 동일 IP의 인증 시도 횟수 폭증 |
| **초기 침투** | ARP 스푸핑 | arp | IP-MAC 주소 1:1 매칭 | ARP Reply 대량 발생 | 동일 IP에 MAC 주소 지속 변조 |
| **초기 침투** | FTP 패킷 스니핑 | ftp | 암호화된 전송 (식별불가) | ID/PW 평문 노출 | Payload 내 계정 정보 노출 여부 |
| **초기 침투** | WhatWeb | http.user\_agent | 브라우저 기반 요청 | UA 내 'WhatWeb' 포함 | 헤더 내 스캐너 명시 |
| **관리자 획득** | SQL Injection | http.request.uri | 일반적인 파라미터 요청 | SELECT, cat, ; 등 포함 | 요청 데이터 내 DB 예약어 포함 |
| **시스템 파괴** | Cmd Injection | http.request.uri | 정상 URI 경로 호출 | ; , cat , passwd 포함 | 메타문자 기반 OS 명령어 시도 |
| **시스템 파괴** | Reverse Shell | tcp.flags.syn == 1 | 인바운드 서비스 접속 | 비정상 아웃바운드 접속 | 외부 연결 및 쉘 명령어 |
| **최종 파괴** | DoS 공격 | icmp / tcp.flags.syn | 안정적 트래픽 유지 | SYN/ICMP 과다 유입 | PPS가 정상 임계치를 초과 |
| **세션 탈취** | Reflected XSS | http.request.uri | 일반 검색 파라미터 | <script>, cookie | URI 내 인코딩된 스크립트 포함 |
| **계정 침투** | Mail Brute Force | http.request.method=="POST" | 1회성 로그인 요청 | 로그인 POST 집중 | 대량의 401 에러 후 200 발생 |
| **정보 탈취** | Blind Injection | http.request.uri | 일반적인 웹 페이지 파라미터 요청 | union, select 등 예약어 포함 | 특정 IP의 요청 빈도 폭증 및 응답 크기 변화 |
| **미끼 투척** | 악성 쉘 실행 | tcp (외부 IP) | 목적지 직접 통신 | 해커 IP(Proxy) 경유 | 게이트웨이가 비인가 IP로 변경 |
| **권한 오용** | CSRF 공격 | http.referer | 내부 도메인발 호출 | 외부/비어있는 Referer | 외부 주소발 중요 기능 호출 |
| **서버 침투** | LFI 공격 | http.request.uri | 정상 파일 경로 호출 | ../ , /etc/passwd | 상위 디렉토리 접근 패턴 존재 |
|  | File Upload | http.request.method=="POST" | 이미지 데이터 전송 | 파일 내 <?php 포함 | 이미지 시그니처 내 스크립트 포함 |
| **서버 장악** | Reverse Shell | tcp.flags.syn==1 | 외부 접속(Inbound) | 서버발 아웃바운드 | 서버가 비인가 IP로 먼저 연결 |
| **금전 이득** | Salami Attack | mysql / tcp.port==3306 | 정상 트랜잭션 쿼리 | 소액 UPDATE 반복 | 미세 금액 조작 쿼리 임계치 초과 |
| **미끼 투척** | 악성 쉘 실행 | tcp  (외부 IP) | 목적지 직접 통신 | 해커 IP(Proxy) 경유 | 게이트웨이가 비인가 IP로 변경 |
| **권한 오용** | CSRF 공격 | http.referer | 내부 도메인발 호출 | 외부/비어있는 Referer | 외부 주소발 중요 기능 호출 |
| **서버 침투** | LFI 공격 | http.request.uri | 정상 파일 경로 호출 | ../ , /etc/passwd | 상위 디렉토리 접근 패턴 존재 |
| File Upload | http.request.method=="POST" | 이미지 데이터 전송 | 파일 내 <?php 포함 | 이미지 시그니처 내 스크립트 포함 |
| **서버 장악** | Reverse Shell | tcp.flags.syn==1 | 외부 접속(Inbound) | 서버발 아웃바운드 | 서버가 비인가 IP로 먼저 연결 |
| **금전 이득** | Salami Attack | mysql / tcp.port==3306 | 정상 트랜잭션 쿼리 | 소액 UPDATE 반복 | 미세 금액 조작 쿼리 임계치 초과 |

**6.5.3 Snort 탐지 정책 설계 및 자동화 전략**

1) 탐지 정책 식별 체계 (SID 명명 규칙) 수립

- 룰 번호만으로 공격 유형(정찰, DoS, 웹 등)을 즉시 식별하여 관제 요원의 직관적인 대응 지원

- 임의의 번호가 아닌, 내부 표준 아키텍처를 기반으로 룰을 계층화하여 관리 효율성 극대화

- SID 구조 : 2 + XX + YY + ZZ (고유 식별자 + 대분류 + 중분류 + 순번)

2) 공격 유형별 탐지 범위 및 분류 체계

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **대분류(XX)** | **공격 유형** | **중분류(YY)** | **탐지 목표 및 내용** | **공격 기술 (룰)** |
| **1** | 정보 수집 (염탐/정찰) | 01 스텔스/ 특수 스캔 | Nmap 등 도구의 특수 옵션을 사용하여 방화벽을 우회하거나 몰래 스캔하는 행위 식별 | Null Scan,  Xmas Scan |
| 02 일반/스크립트 스캔 | 핑(Ping)을 무시하거나 스크립트 엔진(NSE)을 사용하여 서버 정보를 수집하는 행위 탐지 | -Pn 스캔,  NSE 스크립트 스캔, DNS Zone Transfer |
| **2** | 서비스 마비  (DoS 공격) | 01 ICMP 과부하 | 패킷 크기를 비정상적으로 키우거나(Large Packet) 대량의 핑을 쏘아 가용성을 저해하는 행위 식별 | ICMP Flood,  Large Packet(-s) |
| 02 트래픽 폭탄 | TCP/UDP 연결을 무한대로 요청하여 시스템 자원을 고갈시키는 공격 탐지 | (TCP·UDP Flood/향후 확장 예정) |
| **3** | 웹 해킹 (사이트 공격) | 01 입력값 변조 | 로그인 창 등에 악성 쿼리나 스크립트를 주입하여 정보를 탈취하거나 인증을 우회하는 행위 식별 | SQL Injection,  XSS  (Cross Site Scripting) |
| 02 명령어 주입 | 웹 서비스를 통해 서버 컴퓨터의 시스템 명령어를 몰래 실행하려는 시도 탐지 | Command Injection,  Reverse Shell Connection |
| 03 경로 탐색 | 허용되지 않은 상위 폴더(../)로 이동하여 시스템 주요 파일을 열람하거나 탈취하려는 행위 식별 | Directory Traversal (../) |
| **4** | 패스워드 공격 (무차별 대입) | 01 SSH 접속 시도 | 관리자 포트(22번) 등 특정 서비스 포트에 패스워드를 반복적으로 대입하여 탈취하려는 행위 탐지 | Hydra  Brute Force Attack |

3) Snort 룰 자동화 프로세스

- 관리자로부터 공격 유형 및 침입 지표 (IP, Port, 패턴 등)를 입력받아 Snort의 local.rules

파일을 실시간으로 업데이트하는 자동화 코드를 설계함

**7. 자동화 코드 관리**

**7.1 코드 흐름도**

**7.1.1 네트워크 코드 흐름도**

|  |
| --- |
|  |

**7.1.4 관제 및 대응 코드 흐름도**

|  |
| --- |
|  |

**7.1.5 주요정보통신시설 기반 코드 흐름도**

**7.2.4 관제 및 대응부분 함수정의서**

- 입력받은 snort rule 옵션을 바탕으로 snort rule을 자동으로 작성하고, snort rule을 구분하기 위한 값인 sid값을 자동으로 배분한다. 룰을 작성한 이후 문법을 검증한다.

|  |  |
| --- | --- |
| **목적** | 간편한 snort rule 작성 및 모든 Security Onion 내 snort rule 동기화 |
| **특징** | 사용자 입력을 통한 snort rule 생성, IDS서버에 snort rule 추가, IDS 서버 내부 명령어를 통한 snort rule 문법 검증 |
| **사용 환경** | Security Onion |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NO** | **클래스** | **함수명** | **주요 기능 (기획서 연계)** | **입력 예시** | **출력/결과** |
| 1 | Server | init() | 인자값으로 클래스의 변수를 초기화한다. | 서버 ip, ssh id, ssh pw | N/A(해당없음) |
| 2 | ssh() | ssh 접속하여 인자로 받은 명령 실행 | 실행할 명령어 | 실행한 명령어의 결과 반환 |
| 3 | SnortManager | init() | 인자값으로 시작 SID 번호를 설정한다. | 시작 SID 번호 | N/A(해당없음) |
| 4 | create\_rule() | 사용자로부터 입력받아 snort rule 문자열을 생성한다. | 프로토콜, src ip, src port, dst ip, dst port, 탐지 메세지 등 | 생성한 snort rule 문자열 반환 |
| 5 | deploy\_to\_so() | create\_rule()을 통해 만든 문자열을 local.rules 파일 내에 추가하고 문법을 검증한다. | N/A(해당없음) | 문법 적용 결과(적용 성공 or 실패) 출력 |

**7.2.5 주요정보통신시설 기반 함수정의서**

- 서버 목록(Inventory) DB를 기반으로 주정통 항목을 자동 점검하고, 결과를 DB/웹으로 가시화할 수 있도록 점검·저장·확장 구조를 정의한다(현재는 CSV 저장까지 구현)

|  |  |
| --- | --- |
| **목적** | 주정통 취약점 진단 항목(U/W/WEB/S/N) 자동 점검 및 결과 리포팅을 통해 운영 효율과 보안 설정 준수 확인을 달성한다. |
| **특징** | 0/1/2 자동 판정, 증거(detail) 로그 수집, host+code 기준 최신 결과 유지(upsert), host\_memo(서버 역할) 기록으로 DB 적재/웹 가시화 연동을 용이하게 한다. |
| **사용 환경** | 진단 대상: Linux/Unix/Windows, WEB/WAS 및 보안/네트워크 장비(세션 01~05). 실행 환경: 관리자() 서버에서 Jupyter 또는 CLI로 실행. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NO** | **스크립트/함수 그룹** | **대상 서버/OS** | **주요 기능 (기획서 연계)** | **입력 예시 (IP/옵션)** | **출력/결과** |
| 1 | \_index/items.json (+ items.csv) / INDEX.md | 모든 대상 서버 | 주정통 전체 항목 인덱스(코드/영역/분류/중요도/페이지)를 기준 데이터로 제공(마스터/CSV 실행 시 참조) | items.json(코드 목록), items.csv(엑셀 공유용), INDEX.md(경로 인덱스) | 항목 메타데이터 로드 및 세션별 필터링 기반 제공 |
| 2 | Server.connect()/close() + \_\_enter\_\_/\_\_exit\_\_ | 모든 대상 서버 | SSH 세션 수립/종료를 표준화하고 with 문으로 안전하게 자원 정리(점검 연속 실행 시 연결 관리) | with Server(host, '', user, pw, port=22, timeout=10) as server: ... | 연결/해제 자동 처리(예외 발생 시에도 close 보장) |
| 3 | Server.ssh\_str() / Server.ssh() | 모든 대상 서버 | 원격 명령 실행 및 출력 수집. ssh()는 2025 스타일 호환(stdout 객체), ssh\_str()는 문자열 반환(utf-8 decode, strip). | server.ssh\_str('id') / server.ssh('cat /etc/os-release') | stdout 문자열/파일 객체 반환(증거(detail)로 저장 가능) |
| 4 | Server.detect\_os() | Linux/Unix 서버 | os-release 기반 OS 식별(ID=)로 점검 분기/메타 기록에 활용(미탐 시 unknown) | /etc/os-release | server.os='ubuntu'|'rocky'|'unknown' 등 |
| 5 | MASTER 템플릿: find\_root() + sys.path 보정 | Jupyter Notebook | 실행 위치가 달라도 server.py/\_index/items.json를 찾도록 루트 탐색 후 import 경로 자동 보정 | Jupyter 실행 위치(2026 주정통 코드 정리 내부/하위) | ROOT 자동 결정 + 모듈 import 안정화 |
| 6 | MASTER 템플릿: importlib.reload(\_tools.run\_checks\_csv) | Jupyter Notebook | 노트북 재실행 시 모듈 캐시로 인한 함수 시그니처 불일치(예: host\_memo) 방지 목적의 강제 reload | import importlib; import \_tools.run\_checks\_csv as m; importlib.reload(m) | 최신 코드 기준으로 실행(캐시로 인한 TypeError 예방) |
| 7 | MASTER\_01~05 + parse\_selection() | Jupyter Notebook | 세션별 항목 리스트 표시 및 다중 선택(all/1,2/범위/코드), host\_memo 입력 | 선택: all 또는 U-01,U-02 / memo: '웹서버' | 선택 항목 순차 점검 실행 + results.csv 갱신 |
| 8 | MASTER 입력: host\_memo(서버 용도/역할 태깅) | 모든 세션(01~05) 공통 | 웹서버/DNS/DB/보안장비/관제 등 서버 용도를 메모로 기록하여 결과(행 단위)에 함께 저장 | host memo: '웹서버', 'DNS', 'Security Onion' 등 | results.csv 컬럼 host\_memo에 저장(추후 DB/웹 가시화 필터링에 활용) |
| 9 | \_tools/run\_checks\_csv.py::CSV\_FIELDS + RESULT\_LABEL | CLI/Jupyter | 결과 스키마(컬럼)와 0/1/2 라벨 매핑을 표준화하여 CSV/DB 적재 시 스키마 일관성 유지 | CSV\_FIELDS=[host,host\_memo,code,result,...], RESULT\_LABEL={0:'양호',...} | 결과 테이블(또는 CSV) 컬럼/라벨 표준 확정 |
| 10 | \_tools/run\_checks\_csv.py::\_load\_index() | CLI/Jupyter | items.json을 code→meta 맵으로 로드하여 결과에 domain/section/category/title/importance를 함께 기록 | \_index/items.json | index[code] 메타데이터 제공(결과 row enrichment) |
| 11 | \_tools/run\_checks\_csv.py::\_dedupe\_keep\_order() | CLI/Jupyter | 사용자 선택 순서를 유지하면서 중복 코드를 제거(동일 항목 중복 점검 방지) | ['U-01','U-01','U-02'] | ['U-01','U-02'] |
| 12 | \_tools/run\_checks\_csv.py::run\_checks\_to\_csv() | CLI/Jupyter | 선택 코드 순차 점검, 0/1/2 판정, detail(증거) 수집, host+code 최신 덮어쓰기(upsert), updated\_at 기록 | --host 192.168.0.10 --host-memo '웹' --codes U-01,U-02 | \_results/results.csv 생성/갱신(추후 MariaDB 적재/웹 가시화 연동 용이) |
| 13 | \_tools/run\_checks\_csv.py::\_parse\_codes() | CLI/Jupyter | 콤마 목록(--codes) 또는 파일(--codes-file) 기반으로 점검 코드 리스트 구성(중복 제거, 대문자 통일) | --codes U-01,U-02 또는 --codes-file my\_codes.txt | codes=[...] 확정(선택한 항목만 연속 점검) |
| 14 | \_tools/run\_checks\_csv.py::\_host\_id() | CLI/Jupyter | upsert 키(호스트 식별자) 생성: 22면 host, 아니면 host:port | host=192.168.0.10, port=22 / port=2222 | host\_id='192.168.0.10' 또는 '192.168.0.10:2222' |
| 15 | \_tools/run\_checks\_csv.py::\_read\_existing\_csv() + \_write\_csv() | CLI/Jupyter | 기존 결과를 읽고(host+code) 최신 결과로 덮어써서 저장(최신만 유지). 스키마: host\_memo 포함 | \_results/results.csv | CSV upsert 저장(중복 누적 없이 최신 결과만 남김) |
| 16 | \_tools/run\_checks\_csv.py::detail 캡처(redirect\_stdout) + \_escape\_one\_line() | CLI/Jupyter | 각 항목 점검 중 출력(print)을 증거(detail)로 캡처하고, 줄바꿈을 \n으로 이스케이프하여 1행=1결과 유지 | checks 모듈의 print 출력 | CSV detail 컬럼에 증거 저장(MariaDB LOAD DATA 적재 친화) |
| 17 | \_tools/run\_checks\_csv.py::\_run\_one() (U/W/WEB/S/N 라우팅) | CLI/Jupyter | 코드 prefix별 checks 모듈로 분기하여 해당 run(code, server)를 호출 | U-01 / W-01 / WEB-01 / S-01 / N-01 | 각 도메인 점검 로직 실행 및 0/1/2 반환 |
| 18 | checks.<domain>.run() 공통 패턴 + CHECKS 레지스트리 | 세션 01~05 공통(U/W/WEB/S/N) | 코드→함수 맵(CHECKS)에서 함수 선택 후 실행, 미구현/예외는 2로 처리하여 배치 점검을 중단하지 않음 | run('U-01', server) / CHECKS['U-01'](server) | 0/1/2 반환 + verbose 시 원인/가이드 출력 |
| 19 | checks.unix 공통 유틸(\_cmd/\_read/\_exists/\_config\_value/권한 파싱) | Linux/Unix | 원격 명령 실행 래퍼, 파일 존재/읽기, 설정값 추출, stat/ls 기반 권한·소유자 판별, 서비스/포트/프로세스 확인 등 재사용 유틸 제공 | /etc/ssh/sshd\_config, stat/ls, systemctl, ss/netstat 등 | 점검 함수의 공통 판정 근거/증거 출력 표준화 |
| 20 | checks.unix.run() + CHECKS(U-xx) | Linux/Unix 서버(Ubuntu/Rocky 등) | 계정/파일권한/서비스/패치/로그 등 U-xx 자동 점검(증거 출력 + 0/1/2 반환) | /etc/ssh/sshd\_config, /etc/passwd 등(원격 명령 기반) | 0=양호 / 1=취약 / 2=수동확인·점검실패 |
| 21 | checks.windows 공통 유틸(\_ps\_json/\_secpol\_values/\_reg\_get/\_resolve\_sids) | Windows Server(OpenSSH) | PowerShell EncodedCommand 실행, JSON 추출/파싱, 로컬 보안정책(secedit export) 캐시, 레지스트리 조회, SID→계정명 변환 등 재사용 유틸 제공 | secedit /export, HKLM 레지스트리 경로, SID 목록 등 | 정책/계정/감사 설정값을 안정적으로 수집(점검 함수에서 재사용) |
| 22 | checks.windows.\_ps(EncodedCommand) + run() + CHECKS(W-xx) | Windows Server(OpenSSH) | PowerShell(EncodedCommand) 기반 정책/레지스트리/서비스 자동 점검(증거 출력 + 0/1/2 반환) | secedit export, registry query, service status 등 | 0/1/2 + 증거(detail) 출력 |
| 23 | checks.web 공통 유틸(\_nginx\_dirs/\_grep/\_curl\_head/\_header\_value/권한 파싱) | WEB/WAS 서버 | 웹서버 설정 디렉터리 탐색, grep 기반 설정 검색, curl -I 헤더 수집 및 보안 헤더 추출, 파일 권한 판별 등 재사용 유틸 제공 | /etc/nginx,/etc/apache2,/etc/httpd, curl -I http://127.0.0.1 등 | 웹 설정/헤더 기반 자동판정 근거 제공(다중 웹서버 공존 시 결과 합산) |
| 24 | checks.web.run() + CHECKS(WEB-xx) | WEB/WAS 서버(nginx/apache/tomcat 등) | 웹 서비스 설정/헤더/SSL/로그권한 등 자동 점검(가능한 항목은 자동판정, 불가 항목은 수동확인) | curl -I http://127.0.0.1, /etc/nginx/\*, /etc/httpd/\* 등 | 0/1/2 + 증거 출력 |
| 25 | checks.security.\_manual() (수동확인 템플릿) | 보안 장비 | 장비별 상이한 UI/CLI를 고려해 '확인 포인트'를 표준 포맷으로 출력(현재는 2 반환, 추후 자동화 확장 지점) | 정책/로그/펌웨어/계정 설정 화면 또는 CLI 출력 | 수동 점검 체크리스트 출력 + 2 반환 |
| 26 | checks.security.run() + CHECKS(S-xx) | 보안 장비(방화벽/VPN/IPS/WAF 등) | 장비 계정/접근통제/로그/정책 항목 점검(현재 수동 확인 템플릿 기반) | 장비 설정 메뉴/CLI/관제 로그 기준으로 확인 | 2(수동확인) + 체크리스트 출력(자동판정 확장 예정) |
| 27 | checks.network.\_manual() (수동확인 템플릿) | 네트워크 장비 | 벤더/OS별 명령 차이를 고려해 '확인 포인트/예시 명령'을 표준 포맷으로 출력(현재는 2 반환) | show running-config / line vty / snmp 설정 등 | 수동 점검 체크리스트 출력 + 2 반환 |
| 28 | checks.network.run() + CHECKS(N-xx) | 네트워크 장비(라우터/스위치 등) | VTY/ACL/SNMP/불필요 서비스 차단 등 설정 점검(현재 수동 확인 템플릿 기반) | show running-config / line vty / snmp 설정 등 | 2(수동확인) + 체크리스트 출력(자동판정 확장 예정) |

**8.4 관제 및 대응 구현테스트 및 침투테스트**

**8.4.1 관제 및 대응 구현테스트**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **분야** | **테스트 ID** | **테스트 케이스** | **확인 방법** | **수행 결과** |
| 웹 해킹 | PP- WEB-01 | SQL Injection을 통한 관리자 계정 탈취 시도(예: sqlmap, 대상: nginx reverse proxy + apache backend) | WAF/IPS 차단 로그 또는 웹 접근 로그(access.log)에서 공격 패턴 기록 및 경고/차단 여부, SIEM 알람 확인 |  |
| PP- WEB-02 | 파일 업로드 취약점을 이용한 웹셸(Web Shell) 업로드 시도(대상: 업로드 기능/디렉터리) | 웹 서버 error.log/접근 로그에 업로드 시도 기록, 보안관제(Security Onion/Snort) 이벤트 발생 여부 확인 |  |
| PP - WEB-03 | 디렉터리 트래버설(Traversal)로 민감 파일 접근 시도(예: /etc/passwd, 설정 파일) | 웹 접근 로그에서 비정상 URI 패턴 확인, 차단 정책 적용 여부 및 경고(관제/알람) 확인 |  |
| 시스템 해킹 | PP-PT-01 | SSH 서비스 식별 및 접근 시도(Nmap 스캔으로 포트/버전 수집) | auth.log/secure 및 방화벽 로그에서 스캔 패턴 확인, 관제(Security Onion)에서 스캔 탐지 이벤트 확인 |  |
| PP-PT-02 | 무작위 대입(Brute Force)으로 SSH 접속 시도(hydra 등) | 로그인 실패 패턴(auth.log/secure) 누적 확인, 임계치 초과 시 iptables/ufw 차단 적용 여부 확인, 차단 후 재시도 시도 실패 확인 |  |
| PP-PT-03 | root 원격 로그인 시도(정책: PermitRootLogin 제한 및 직접 로그인 차단) | sshd\_config 정책 적용 여부 및 접속 차단 확인, 로그인 실패 로그 및 경고 메시지 기록 확인 |  |
| 네트워크 해킹 | PP-NET-01 | ARP Spoofing으로 내부 구간 MITM 시도(동일 IP에 MAC 주소 지속 변조) | ARP 테이블 변조/게이트웨이 MAC 변경 징후 확인(호스트/스위치), IDS/Snort 경고 및 관제 이벤트 발생 여부 확인 |  |
| PP-NET-02 | FTP 패킷 스니핑으로 계정 정보 노출 여부 확인(평문 전송) | Wireshark 필터(tcp.port==21)로 Payload 내 계정 정보 노출 확인, 정책상 FTP 미사용/대체(SFTP/FTPS) 적용 여부 확인 |  |
| DNS 공격 | PP-DNS-01 | DNS 정보 탈취 시도(예: Zone Transfer(AXFR) 요청, dig) | DNS 서버 로그에서 AXFR/비정상 요청 기록 확인, 접근 통제/차단 여부 확인, 관제 이벤트 발생 여부 확인 |  |
| PP-DNS-01 | DNS 서버 설정 상태 점검 및 정보 노출 여부 확인(재귀/권한 없는 질의 허용 등) | named.conf 등 설정 검토 결과를 증거로 남기고, 보안 정책 기준 충족 여부(양호/취약) 판정 |  |
| DDoS | PP-DOS-01 | SYN Flood/대량 연결 시도(hping3 등)로 서비스 가용성 저하 유발 | 트래픽 모니터링(MRTG 등)에서 급증 탐지, 방화벽/iptables rate-limit 또는 차단 정책 적용 여부 확인, 서비스 영향도 기록 |  |
| PP-DOS-01 | HTTP Flood 또는 Slow 공격(slowloris 등)으로 웹 서비스 자원 고갈 시도 | nginx/apache 접속 로그 및 에러 로그에서 비정상 세션 패턴 확인, 차단/제한(연결수/타임아웃) 정책 적용 여부 확인 |  |
| APT 시나리오 | PP-APT-01 | 내부망 침투 및 시스템 파괴 시나리오(권한 획득 → 내부 이동 → 중요 파일 변경/삭제 시도) | 서버/시스템 로그에서 권한 상승·명령 실행 흔적 확인, 관제(Security Onion/SIEM)에서 이상행위 알람 발생 여부 확인 |  |
| PP-APT-01 | C2 통신/Beaconing 패턴 시뮬레이션(주기적 외부 통신/비정상 도메인 질의 등) | DNS/네트워크 로그에서 주기적 패턴 확인, IDS(Snort) 탐지 룰 매칭 또는 경고 발생 여부 확인 |  |