

**팬택씨앤아이 OKPlaza(공급사)**

**모의해킹 결과 보고서**

2025.06

주의사항

|  |
| --- |
| 본 보고서는 보안에 관련된 내용이 기술되어 있으므로 본 문서를 열람할  권한이 없는 자에게 공개되어서는 안됩니다.  따라서 사전 승인 없이 본 내용이 전부 또는 일부에 대한 복사, 전제,  배포, 사용 등을 금합니다. |

|  |  |  |  |
| --- | --- | --- | --- |
| **문서정보** | | | |
| **사 업 명** | 팬택씨앤아이 서비스(1차) 모의해킹 | | |
| **작 성 자** | 박준태 | | |
| **개정이력** | | | |
| **버전** | **수정일자** | **작성자** | **개요** |
| 1.0 | 2025-06-20 | 박준태 | 원본 작성 |
|  |  |  |  |
|  |  |  |  |

목차

I. 개요 6

1. 개요 6

1.1. 수행목적 6

1.2. 진단방법 6

1.3. 진단항목 6

1.4. 어플리케이션 주요 진단 항목 6

2. 수행 방법 10

2.1. 일정 및 장소 10

2.2. 위험 등급 평가 기준 10

2.3. 대상 11

2.4. 수행 인원 11

II. 진단결과 요약 12

III. 세부 수행내역 13

1. OKPlaza(공급사) 13

1.1. [W-009]정보 누출 13

1.2. [W-011]크로스 사이트스크립팅(XSS) 16

1.3. [W-013]불충분한 인증 18

1.4. [W-018]불충분한 세션만료 21

1.5. [W-019]세션고정 23

1.6. [W-129]취약한 HTTPS 암호 알고리즘 이용 25

1.7. [W-233]크리덴셜 스터핑 26

IV. 주요 취약사항 및 대응방안 29

1. 웹 어플리케이션 29

1.1. [W-009]정보 누출 29

1.2. [W-011]크로스사이트 스크립팅(XSS) 46

1.3. [W-013]불충분한 인증 55

1.4. [W-018]불충분한 세션만료 58

1.5. [W-019]세션고정 62

1.6. [W-129]취약한 HTTPS 암호 알고리즘 이용 63

1.7. [W-233]크리덴셜 스터핑 71

1. 개요
2. 개요
   1. 수행목적

본 진단은 팬택씨앤아이 OKPlaza(공급사) 서비스의 존재하는 보안 약점을 도출하고, 발견된 보안 약점의 원인을 분석하여 대응방안을 도출하기 위함.

* 1. 진단방법

실제 해커가 사용하는 최신 해킹기법을 이용하여 응용 프로그램의 취약점을 수작업으로 진단하였으며, 운영 서비스의 장애 및 영향을 미치지 않는 범위 내에서 진단 수행.

* 1. 진단항목

아래 진단항목은 일반적인 모의해킹 진단 시 이용될 수 있는, 경험적으로 축적된 다양한 항목의 집합이며 대상 시스템 및 진단 환경에 따라 달라질 수 있음.

진단항목에 대한 상세 진단내용은 [IV. 주요 취약사항 및 대응방안]을 참고.

* 1. 어플리케이션 주요 진단 항목

※ W/M-0XX(주요정보통신), W/M-1XX(전자금융), W/M-2XX(OWASP)

| **취약점 코드** | **진단 항목** | **점검 목적** | **위험도** |
| --- | --- | --- | --- |
| W-001 | 버퍼 오버플로우 | 어플리케이션에서 파라미터 입력 값에 대한 적절성을 점검하여 비정상적 오류 발생을 차단 | 상 |
| W-002 | 포맷스트링 | 포맷스트링 버그로 인한 시스템 권한 탈취의 위험 방지 | 상 |
| W-003 | LDAP 인젝션 | 취약한 시스템에 신뢰할 수 없는 LDAP코드 삽입 공격을 통한 비인가자의 악의적인 행위를 차단 | 상 |
| W-004 | 운영체제 명령실행 | 적절한 검증절차를 거치지 않은 사용자 입력값에 의해 의도하지 않은 시스템 명령어 실행 방지 | 상 |
| W-005 | SQL 인젝션 | 대화형 웹 사이트에 비정상적인 사용자 입력값 허용을 차단하여 악의적인 데이터베이스 접근 및 조작 방지 | 상 |
| W-006 | SSI 인젝션 | 입력값 검증 절차를 마련하여 악의적인 파일을 include 못하도록 차단 | 상 |
| W-007 | XPath 인젝션 | XPath 쿼리에 대한 적절한 필터링을 적용하여 웹사이트의 로직 손상 및 특정 데이터 추출 차단 | 상 |
| W-008 | 디렉터리 인덱싱 | 디렉터리 내 불필요한 파일정보의 노출을 차단 | 하 |
| W-009 | 정보 누출 | 에러페이지 내 2차 공격에 활용될 수 있는 불필요한 정보 노출 차단(시스템정보, 절대경로정보, 컴파일 소스정보 등) | 하 |
| W-010 | 악성 콘텐츠 | 사이트 내 악의적인 콘텐츠 삽입 및 실행 방지 | 중 |
| W-011 | 크로스사이트 스크립팅(XSS) | 웹 페이지 내 크로스사이트 스크립팅 취약점을 제거하여 악성스크립트의 실행 차단 | 중 |
| W-012 | 약한 문자열 강도 | 유추 가능한 취약한 문자열 사용을 제한하여 계정 및 패스워드 추측 공격 방지 | 중 |
| W-013 | 불충분한 인증 | 중요 페이지에 추가 인증으로 접근을 강화하여 불필요한 정보의 노출 및 변조 차단 | 중 |
| W-014 | 취약한 패스워드 복구 | 비인가자를 통한 취약한 사용자 패스워드 획득 및 변경을 방지 | 하 |
| W-015 | 크로스사이트 리퀘스트변조(CSRF) | 사용자의 입력값에 대한 필터링 및 인증에 대한 유효성을 검증하여 인증 요청에 대한 변조 방지 | 중 |
| W-016 | 세션 예측 | 사용자의 세션ID를 추측 불가능하도록 난수로 생성하여 공격자의 불법적인 접근 차단 | 중 |
| W-017 | 불충분한 인가 | 접근 권한에 대한 검증 로직을 구현하여 다른 사용자가 민감한 정보나 인증이 필요한 페이지의 접근 차단 | 중 |
| W-018 | 불충분한 세션만료 | 세션 타임아웃 기능을 구현하여 공격자가 만료되지 않은 세션 활용 방지 | 하 |
| W-019 | 세션고정 | 로그인 시 예측 불가능한 신규 세션ID 발행하여 세션ID 고정 사용 방지 | 하 |
| W-020 | 프로세스 검증 누락 | 비인가자가 하위URL에 직접 접근, 스크립트 조작 등의 방법으로 중요페이지에 접근하는 것을 차단 | 중 |
| W-021 | 파일 업로드 | 서버사이드스크립트 파일 업로드 방지 및 실행 차단 | 상 |
| W-022 | 파일 다운로드 | 파일이 저장된 디렉터리 이외의 접근 및 임의의 위치에 있는 파일 열람, 다운받는 것 방지 | 상 |
| W-023 | 관리자페이지 노출 | 비인가자의 관리자 메뉴 접근 방지 | 하 |
| W-024 | 경로 추적 | 웹 서버/어플리케이션의 중요한 파일과 데이터의 접근 및 실행 방지 | 상 |
| W-025 | 위치 공개 | 폴더 및 파일의 위치를 예측하여 접근 및 정보 획득 방지 | 하 |
| W-026 | 데이터 평문 전송 | 서버와 클라이언트간 평문으로 전송되는 데이터의 유출 방지 | 중 |
| W-027 | 쿠키 변조 | 쿠키 인젝션과 같은 쿠키값 변조를 통한 위장 및 권한 변경을 방지 | 중 |
| W-128 | 취약한 HTTPS 프로토콜 이용 | 취약한 버전의 암호 프로토콜 사용으로 인한 암호화 통신 내용 유출 방지 | 하 |
| W-129 | 취약한 HTTPS 암호 알고리즘 이용 | 보안강도가 낮은 암호 알고리즘 사용으로 인한 암호화 통신 내용 유출 방지 | 하 |
| W-130 | 취약한 HTTPS 컴포넌트 사용 | 취약한 HTTPS 확장 모듈 사용으로 인한 암호화 통신 내용 유출 또는 변조 방지 | 하 |
| W-131 | 취약한 HTTPS 재협상 허용 | HTTPS 재협상 허용으로 인한 암호화 통신 내용 유출 방지 | 하 |
| W-132 | 사전에 정한 조회기간 이상으로 거래내역 조회가능 여부 | 과도한 내역 조회로 인한 불필요한 정보 유출 및 서버 부하 발생 방지 | 하 |
| W-233 | 크리덴셜 스터핑 | 유출된 크리덴셜을 사용하여 단기간 내 대량의 사용자 계정정보 탈취 방지 | 하 |
| W-134 | 불필요한 웹 메서드 허용 | GET, POST이외의 불필요한 Method를 통해 임의의 파일 업로드 또는 삭제 등의 위협 방지 | 하 |
| W-135 | 비밀번호 오류횟수 제한기능 제공 여부 | 비밀번호 오류 횟수 체크하여 무차별 대입 공격 방지 | 중 |
| W-136 | 서버 사이드 요청 위조(SSRF) | 공격자가 직접 접근할 수 없는 네트워크 대역(내부망 등)에 위치한 서버에서 변조된 요청 메시지에 대해 응답값이 반환되는지 여부 확인 | 중 |
| W-037 | 자동화 공격 | 무차별 대입 공격 및 자동화 공격으로 인한 웹 애플리케이션 자원 고갈 방지 | 하 |
| W-138 | XML 외부객체 공격(XXE) | XML 데이터 검증 미흡으로 외부 객체 삽입을 통한 악의적인 행위 차단 | 상 |
| M-101 | OS변조 탐지 | 권한이 상승된 OS변조 단말 이용으로 인한 보안 위협 방지 | 중 |
| M-102 | 프로그램 무결성 검증 | 변조된 프로그램이 정상 실행 될 경우, 악성코드가 포함되어 재배포 되는 등의 보안 위협 방지 | 권고 |
| M-103 | 소스코드 난독화 | 디컴파일된 소스코드의 프로그램의 구조 및 코드 흐름 파악을 통한 앱분석 방지 | 하 |
| M-104 | 디버깅 탐지 | 동적 디버깅을 통한 프로그램 흐름 파악을 통한 앱 분석 방지 | 하 |
| M-105 | 단말기 내 중요정보 저장 | 어플리케이션 사용 폴더 및 외부 저장소에 존재하는 파일 내 중요정보 저장을 통한 정보 유출 방지 | 중 |
| M-106 | 메모리 내 중요정보 노출 | 이용자 단말기 메모리 영역에서 이용자 중요정보의 평문노출 방지 | 권고 |
| M-107 | 화면 내 중요정보 평문 노출 | 이용자 단말기에서 화면 내에 이용자 중요정보의 평문노출 방지 | 상 |
| M-108 | 앱 소스코드 내 운영정보 노출 | 어플리케이션을 디컴파일 및 역분석하여 코드 상에 운영 정보의 노출 제거 | 하 |
| M-109 | 화면 강제실행에 의한 인증단계 우회 | 화면강제 실행, 인증관련 파일 조작 등을 통한 인증단계 우회 방지 | 중 |
| M-110 | 디버그 로그 내 중요정보 노출 | DDMS 등 디버그 로그를 확인 할 수 있는 도구를 이용하여 디버그 로그 내 중요정보 노출 방지 | 하 |
| M-111 | 백그라운드 화면 보호 | 실행중인 어플리케이션이 백그라운드 상태로 진입할 때 저장되는 스냅샷 파일 내 중요정보 노출 방지 | 하 |

1. 수행 방법
   1. 일정 및 장소

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **세부 업무** | **일정** | **장소** |
| 모의해킹 | OKPlaza(공급사) 웹 점검 | 2025-06-11 ~ 2025-06-20 | 신세계아이앤씨 본사 24층 |
| 결과보고서 | 결과보고서 작성 | 2025-06-20 | - |

* 1. 위험 등급 평가 기준

|  |  |  |  |
| --- | --- | --- | --- |
| **등급** | **주요 내용** | **조치기한** | **취약점 예시** |
| **상** | 사고 발생으로 인한 직접적인 피해가 예상되므로 즉시 조치가 필요한 취약점 | 1주일 이내 | - 개인정보의 유출  - 서버 측 명령어를 수행  - 내부 네트워크의 침입 가능  - 금융거래 정보를 조작 가능 |
| **중** | 사고 발생 가능성 및 피해규모가 높지 않으므로 중장기 조치가 가능한 취약점 | 1개월 이내 | - 서비스를 설계 목적과 벗어난 방식으로 오용 가능한 경우  - 게시판 변조  - URL 리다이렉션 경로 조작 등 |
| **하** | 사고 발생 가능성이 매우 낮으나, 예방 차원에서 조치가 필요한 취약점 | 3개월 이내 | - 에러 정보 노출  - 개인정보 마스킹 누락  - 비밀번호 오류 시 계정잠금 |
| **권고** | 회사에서 조치 여부를 선택할 수 있는 사안 | 선택 | - 정보성 항목  - 보안 강화 방안 등  - 전자금융기반 항목 |

* 1. 대상

|  |  |  |  |
| --- | --- | --- | --- |
| **번호** | **분류** | **서비스 이름** | **URL** |
| 1 | WEB | OKPlaza(공급사) | dev.okplaza.kr |

* 1. 수행 인원

|  |  |  |  |
| --- | --- | --- | --- |
| **담당 업무** | **소속** | **이름** | **비고** |
| 모의해킹 | 신세계아이앤씨 B2B영업팀 | 박준태 | - |

1. 진단결과 요약

다음은 팬택씨앤아이 OKPlaza(공급사) 모의해킹을 진행한 결과 발견된 취약점 현황임.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **대상** | **취약점 코드** | **취약점 항목** | **개수** | **위험도** | **비고** |
| OKPlaza(공급사) | W-009 | 정보 누출 | 3 | 하 | - |
| W-011 | 크로스사이트 스크립팅(XSS) | 2 | 중 | - |
| W-013 | 불충분한 인증 | 1 | 중 | - |
| W-018 | 불충분한 세션만료 | 1 | 하 | - |
| W-019 | 세션고정 | 1 | 하 | - |
| W-129 | 취약한 HTTPS 암호 알고리즘 이용 | 1 | 하 | - |
| W-233 | 크리덴셜 스터핑 | 1 | 하 | - |
| **합계** | | **10** | - | - |

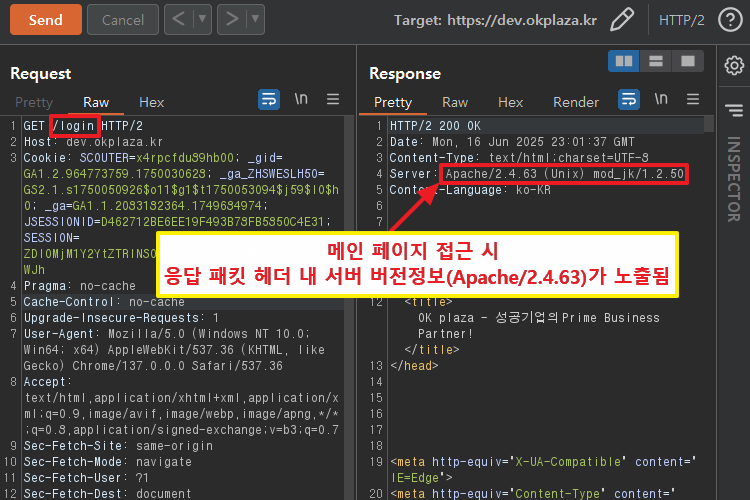
1. 세부 수행내역
2. OKPlaza(공급사)
   1. [W-009]정보 누출
      1. 응답 헤더 내 서버 버전정보 노출

메인 페이지 접근 시 응답 헤더 패킷 내 서버 버전정보(Apache/2.4.63)가 노출됨

▣ 취약점 발견 리스트

|  |  |  |
| --- | --- | --- |
| **NO** | **취약한 메뉴 및 URL** | **취약한 파라미터** |
| 1 | - | - |
| https://dev.okplaza.kr/ |

* + - 1. 메인 페이지 접근 시 응답 헤더 패킷 내 서버 버전정보(Apache/2.4.63)가 노출됨



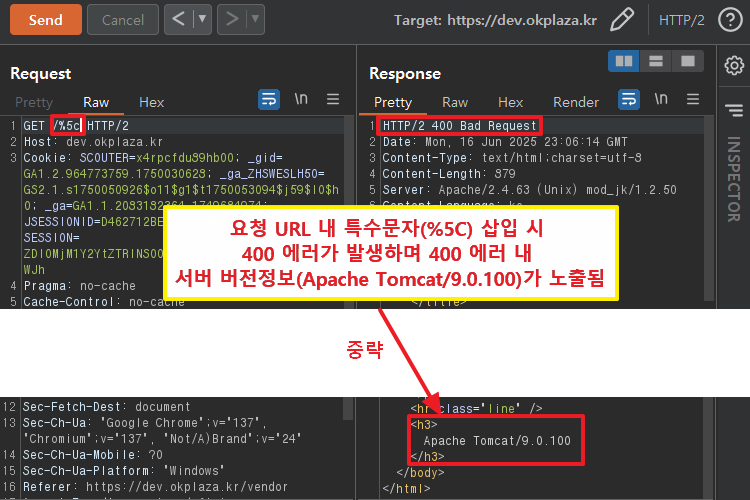
* + 1. 에러 페이지 내 서버 버전정보 노출

요청 URL 내 특수문자(%5C) 삽입 시 400 에러가 발생하며 400 에러 내 서버 버전정보(Apache Tomcat/9.0.100)가 노출됨

▣ 취약점 발견 리스트

|  |  |  |
| --- | --- | --- |
| **NO** | **취약한 메뉴 및 URL** | **취약한 파라미터** |
| 1 | - | - |
| https://dev.okplaza.kr/%5C |

* + - 1. 요청 URL 내 특수문자(%5C) 삽입 시 400 에러가 발생하며 400 에러 내 서버 버전정보(Apache Tomcat/9.0.100)가 노출됨



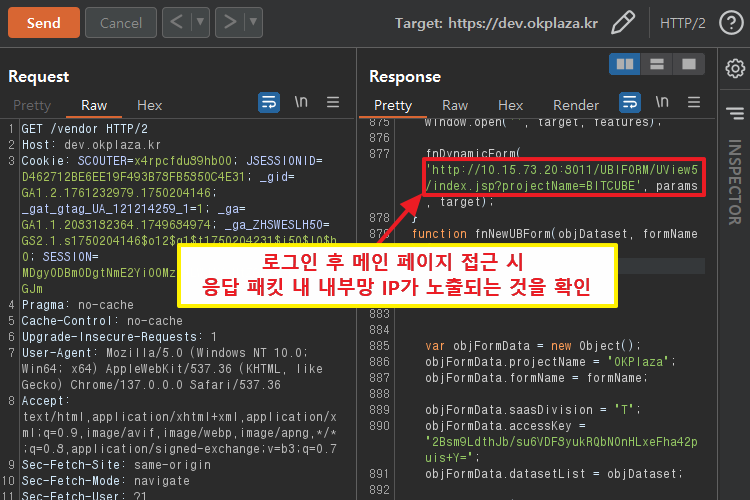
* + 1. 내부망 IP 노출

로그인 후 메인 페이지 접근 시 응답 패킷 내 내부망 IP가 노출됨

▣ 취약점 발견 리스트

|  |  |  |
| --- | --- | --- |
| **NO** | **취약한 메뉴 및 URL** | **취약한 파라미터** |
| 1 | 홈 | - |
| https://dev.okplaza.kr/vendor |

* + - 1. 로그인 후 메인 페이지 접근 시 응답 패킷 내 내부망 IP가 노출되는 것을 확인



* 1. [W-011]크로스 사이트스크립팅(XSS)

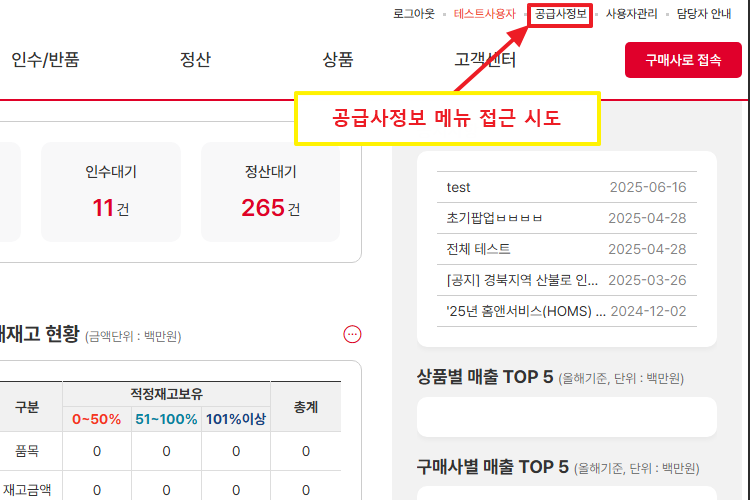
사용자 입력 값 검증이 미흡하여 임의의 스크립트 삽입 및 실행이 가능한 것을 확인함. 공격자가 입력한 스크립트가 실행될 경우 사용자가 의도하지 않은 브라우저 흐름제어가 가능하여 피싱 사이트 접근, 악성코드 감염 등의 위험성이 존재함

※ 이 외 동일한 로직의 기능에 대해서 전수 조치 필요

▣ 취약점 발견 리스트

|  |  |  |
| --- | --- | --- |
| **NO** | **취약한 메뉴 및 URL** | **취약한 파라미터** |
| 1 | 홈>공급사정보-저장 | vendornm, pressentnm, addresdesc, vendorbusiclas, vendorbusitype, homepage, referecedesc, closereason, workmanagenm, accountmanagenm |
| https://dev.okplaza.kr/vendor/organ/saveVendorUpdate |
| 2 | 홈>고객센터>질의응답-등록 | title |
| https://dev.okplaza.kr/vendor/venBoard/saveQna |

* + - 1. 공급사정보 메뉴 접근 시도

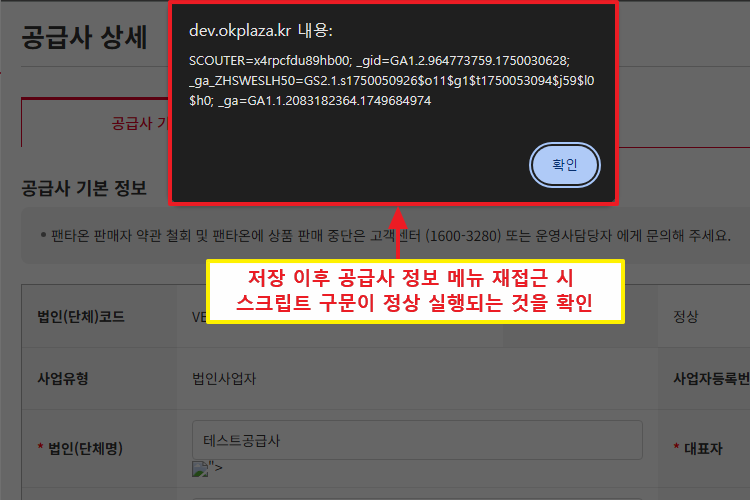


* + - 1. 법인(단체명)란에 스크립트 구문 삽입 후 저장 시도

공격 구문 : “/><img src=x onerror=alert(document.cookie);>



* + - 1. 저장 이후 공급사 정보 메뉴 재접근 시 스크립트 구문이 정상 실행되는 것을 확인



* 1. [W-013]불충분한 인증

사용자 상세 페이지 접근 시 비밀번호 검증이 존재하나 검증이 미흡하여 응답 패킷 내 응답 값 변조 시 비밀번호 없이 접근이 가능함

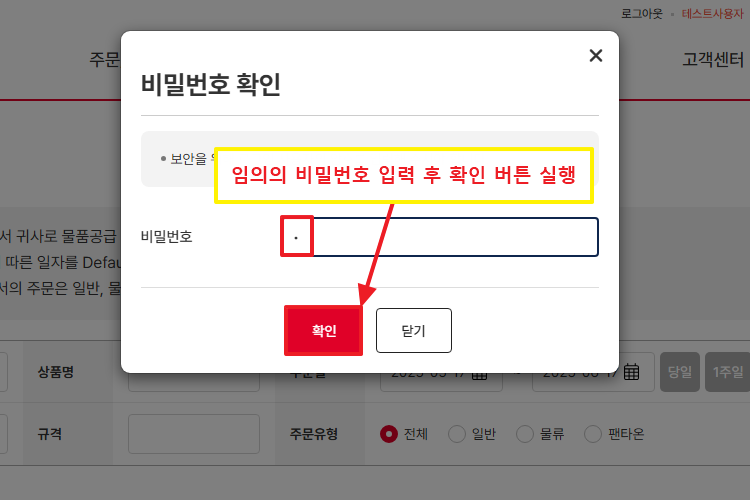
▣ 취약점 발견 리스트

|  |  |  |
| --- | --- | --- |
| **NO** | **취약한 메뉴 및 URL** | **취약한 파라미터** |
| 1 | 홈>사용자 상세 | code |
| https://dev.okplaza.kr/organ/userPasswordCheck |

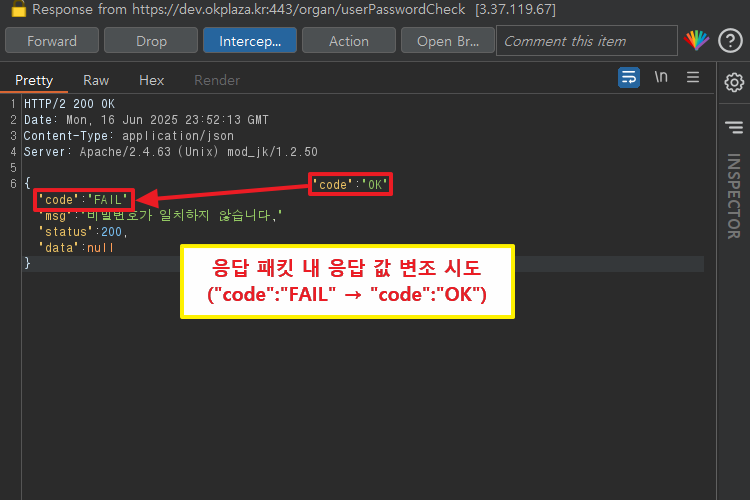
* + - 1. 사용자 이름을 눌러 개인정보 수정 페이지 접근 시도



* + - 1. 임의의 비밀번호 입력 후 확인 버튼 실행



* + - 1. 응답 패킷 내 응답 값 변조 시도



* + - 1. 비밀번호 인증이 우회되어 비밀번호 없이 사용자 상세 페이지 접근이 가능한 것을 확인



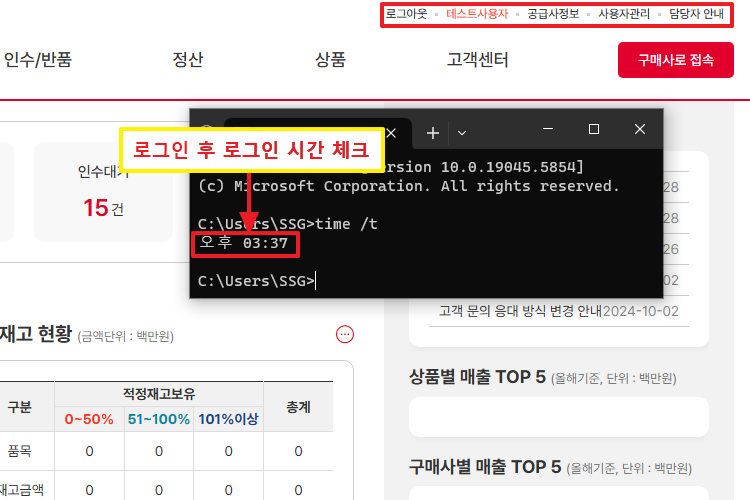
* 1. [W-018]불충분한 세션만료

세션 만료 시간이 설정되어 있지 않아 동작없이 장시간(1시간이상) 대기 시 세션이 만료되지 않는 것을 확인함

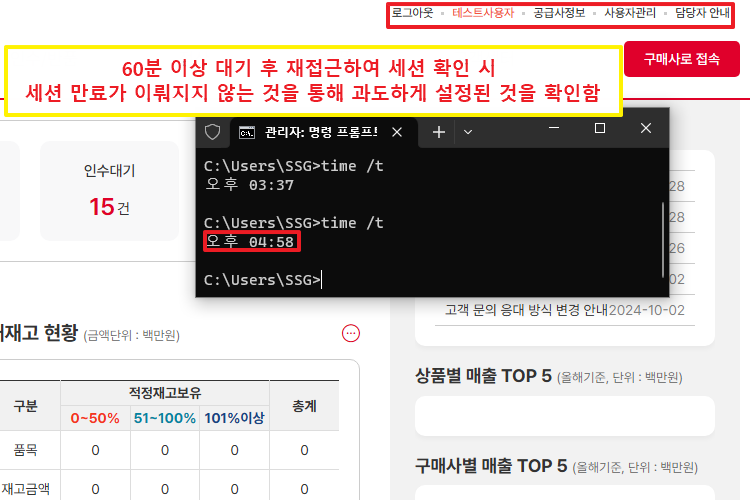
▣ 취약점 발견 리스트

|  |  |  |
| --- | --- | --- |
| **NO** | **취약한 메뉴 및 URL** | **취약한 파라미터** |
| 1 | 로그인 | - |
| https://dev.okplaza.kr/loginCheck |

* + - 1. 로그인 후 로그인 시간 체크



* + - 1. 60분 이상 대기 후 재접근하여 세션 확인 시 세션 만료가 이뤄지지 않는 것을 통해 과도하게 설정된 것을 확인함



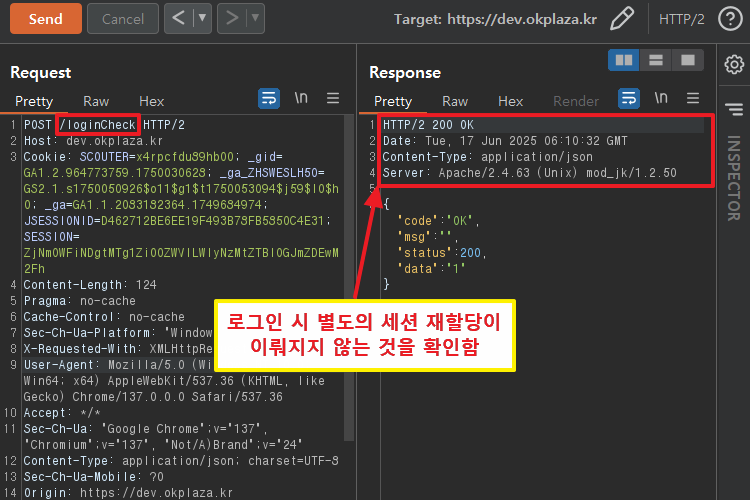
* 1. [W-019]세션고정

로그인/로그아웃 시 세션이 갱신되지 않는 것을 확인함

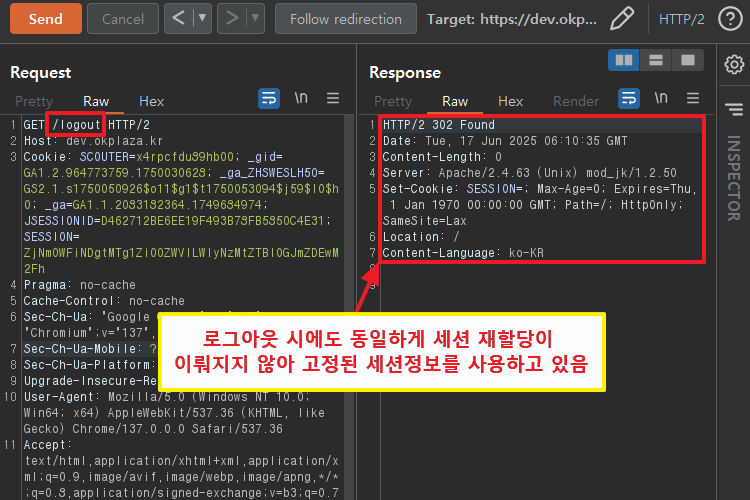
▣ 취약점 발견 리스트

|  |  |  |
| --- | --- | --- |
| **NO** | **취약한 메뉴 및 URL** | **취약한 파라미터** |
| 1 | 로그인 | - |
| https://dev.okplaza.kr/loginCheck |

* + - 1. 로그인 시 별도의 세션 재할당이 이뤄지지 않는 것을 확인함



* + - 1. 로그아웃 시에도 동일하게 세션 재할당이 이뤄지지 않아 고정된 세션정보를 사용하고 있음



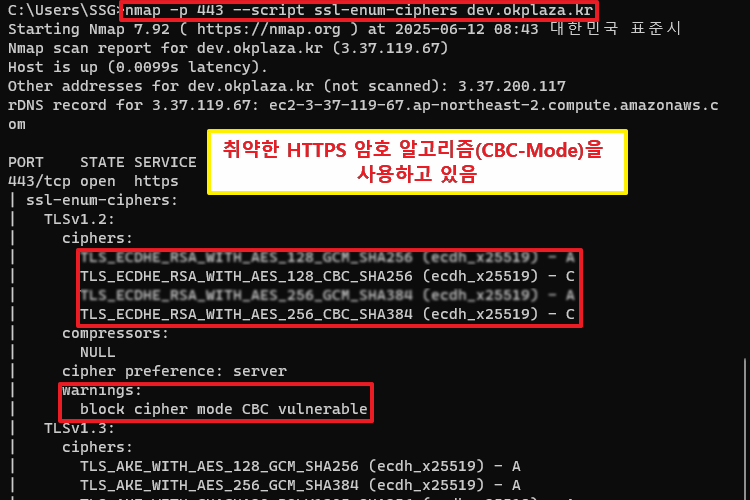
* 1. [W-129]취약한 HTTPS 암호 알고리즘 이용

취약한 HTTPS 암호 알고리즘(CBC-Mode)을 이용하고 있어 Lucky13 취약점을 이용한 암호 해독 공격이 가능할 것으로 판단됨

▣ 취약점 발견 리스트

|  |  |  |
| --- | --- | --- |
| **NO** | **취약한 메뉴 및 URL** | **취약한 파라미터** |
| 1 | 서버 보안 | - |
| https://dev.okplaza.kr/ |

* + - 1. 취약한 HTTPS 암호 알고리즘(CBC-Mode)을 사용하고 있음



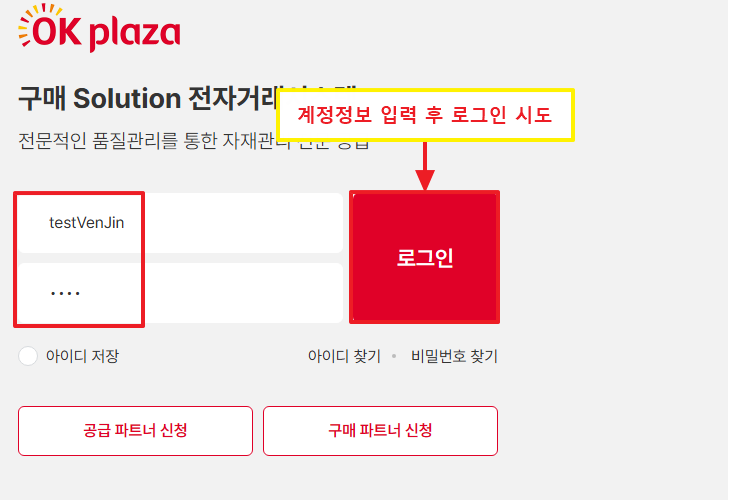
* 1. [W-233]크리덴셜 스터핑

동일 IP주소에서 단시간 내 반복적인 로그인 시도 시 별도의 검증, 차단 로직이 존재하지 않아 크리덴셜 스터핑 공격이 가능할 것으로 판단됨

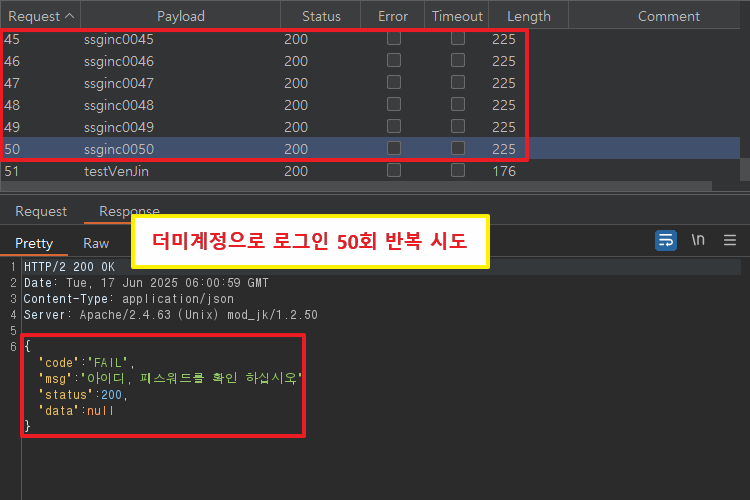
▣ 취약점 발견 리스트

|  |  |  |
| --- | --- | --- |
| **NO** | **취약한 메뉴 및 URL** | **취약한 파라미터** |
| 1 | 로그인 | - |
| https://dev.okplaza.kr/loginCheck |

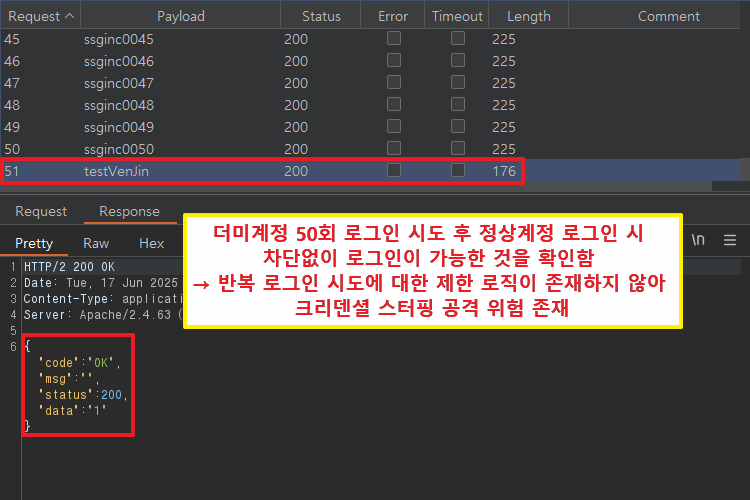
* + - 1. 계정정보 입력 후 로그인 시도



* + - 1. 더미계정으로 로그인 50회 반복 시도



* + - 1. 더미계정 50회 로그인 시도 후 정상계정 로그인 시 차단없이 로그인이 가능한 것을 확인함 → 반복 로그인 시도에 대한 제한 로직이 존재하지 않아 크리덴셜 스터핑 공격 위험 존재



1. 주요 취약사항 및 대응방안
2. 웹 어플리케이션
   1. [W-009]정보 누출

|  |  |
| --- | --- |
| **구분** | **내용** |
| **취약점 코드** | W-009 |
| **위험도** | 하 |
| **취약점 개요** | ㅇ 개발 과정의 코멘트나 오류 메시지 등에서 중요한 정보가 노출되어 공격자에게 2차 공격을 하기 위한 중요한 정보 제공 |
| **보안설정방법** | 1) 에러 처리 미흡  일반적으로 웹에서 발생하는 에러 메시지는 400, 500번대의 에러코드를 리턴하게 되는데, 이러한 에러 코드에 대해 별도의 에러 페이지로 Redirect 하거나 적절한 에러 처리 루틴을 설정하여 처리  특히 데이터베이스의 Query 정보가 노출될 경우 SQL Injection 시에 공격자가 유용하게 사용할 수 있는 정보를 계속 확인할 수 있으므로 에러 처리가 매우 중요함  ■ 에러 처리 방법(Apache)  ※ 참고: 특정 에러코드에 대해 에러 메시지 출력 및 특정 웹 페이지로 Redirect   |  | | --- | | //httpd.conf의 전역 설정에 추가  //또는 가상 호스트의 <VirtualHost> </VirtualHost> 사이에 추가  …..  ErrorDocument 500 “Error Message”  ErrorDocument 404 “/your web root/error.html”  ErrorDo  ument 402 http:/xxx.com/error.html  …. |   ■ 에러 처리 방법(Apache Tomcat)   |  | | --- | | //[웹 컨테이너 디렉터리]/WEB-INF/web.xml 에 아래와 같이 설정함  <web-app>  <error-page>  <error-code>404</error-code>  <location>/error.jsp</location>  </error-page>  ....  </web-app> |   → 전체적인 통합 에러 페이지를 작성한 후 모든 에러코드에 대해 통합 에러 페이지로 Redirect 되도록 설정하여 공격자가 서버 정보 및 에러코드를 수집할 수 없도록 함.  ※ 참고 2: Tomcat 일부 버전에서 URI에 특수문자 삽입 시 발생하는 에러 처리 방법   |  | | --- | | //주소 전체 인코딩(encodeURI) 또는 개별 파라미터 인코딩(encodeURIComponent)  var url = 'action.jsp?param='+encodeURI(str); |   ※ 참고 3: Tomcat 일부 버전에서 URI에 특수문자 삽입 시 발생하는 에러 처리 방법   |  | | --- | | //위의 인코딩하는 방법이 불가할 경우, 아래와 같이 설정함  //Tomcat server.xml의 Connector에 추가  <Connector connectionTimeout="20000"  port="8080"  protocol="HTTP/1.1"  redirectPort="8443"  relaxedQueryChars="[]()^|&quot;"/> //허용할 문자 리스트 |   ■ 에러 처리 방법(IIS 5.0, 6.0)    사용자 에러 페이지 생성 후, 등록    웹 서비스 에러에 대해 별도 페이지 확인    ■ 에러 처리 방법(IIS 7.0)  인터넷 정보 서비스(IIS) 관리자> 해당 웹 사이트> [오류 페이지]에서 400, 401, 403, 404, 500 등 웹 서비스 에러에 대해 별도 페이지 지정    2) 헤더 내 서버정보 노출  서버 및 OS의 종류, 버전, 설치된 모듈 등의 정보가 응답 헤더에 노출될 경우 공격자가 해당 버전의 CVE(공개된 취약점)을 이용하여 공격을 시도할 수 있음  ■ 헤더정보 삭제(PHP)    php.ini 파일에 expose\_php = Off 항목을 추가하고, 재기동 함    ■ 헤더정보 삭제(nginx)  nginx.conf 파일에 server tokens off; 항목을 추가하고, 재기동 함  ■ 헤더정보 삭제(Weblogic)    Weblogic 관리자 페이지에서 아래와 같이 설정함    ■ 헤더정보 삭제(Apache)    httpd.conf 파일에서 아래와 같이 설정해줌   |  | | --- | | ServerTokens Prod  ServerSignature Off |     ■ 헤더정보 삭제(Apache Tomcat)      [톰캣 설치 디렉토리]/tomcat/conf/server.xml 파일에서 아래와 같이 설정해줌   |  | | --- | | …  <Connector ...  server = " "  ... > |   ■ 헤더정보 삭제(IIS 6)    IIS 에서 서버 버전 정보 노출을 막기 위해 URLScan 2.5를 다운로드 함.   microsoft.com/downloads/en/confirmation.aspx?FamilyID=23d18937-dd7e-4613-9928-7f94ef1c902a  설치된 경로의 urlscan.ini 파일에서 아래와 같이 설정   %systemroot%\System32\Inetsrv\urlscan.ini   |  | | --- | | RemoveServerHeader=1 |   (설정 후 적용되기 위해서 시스템 재기동이 필요할 수 있음)  웹 사이트 속성의 HTTP 헤더 탭으로 이동 후 사용자 지정 HTTP 헤더를 선택하고 제거.      ■ 헤더정보 삭제(IIS 8.5)    iis\_stripheaders\_module\_1.0.4.msi 를 다운로드 하여 설치함   https://github.com/Dionach/StripHeaders/releases/tag/v1.0.4    ■ 헤더정보 삭제(IIS 7 버전 이상 HTTPAPI/2.0 제거)     |  | | --- | | HKLM\SYSTEM\CurrentControlSet\Services\HTTP\Parameters 위치에 DisableServerHeader Dword 값 생성 후, 값을 1로 설정 |        PHP 에러메시지 노출 비활성    php.ini 파일에 display\_errors = Off 항목을 추가하고, 재기동 함    2) 시스템 운영정보 노출  HTML 소스, javascript, 응답 패킷 등 일반 사용자가 확인할 수 있는 부분에 시스템 운영정보가 노출될 경우 2차 공격(CVE 취약점 PoC, 내부 정보 스캔 등)에 활용될 위험이 존재하므로 시스템 운영정보는 노출되지 않도록 조치해야함  - 시스템 운영정보 종류 : 개발 서버 경로(dev 등), 품질 서버 경로(stg 등), 관리 페이지 등 내부 시스템 경로, 내부 IP, 관리자 계정 정보 등 |

* 1. [W-011]크로스사이트 스크립팅(XSS)

|  |  |
| --- | --- |
| **구분** | **내용** |
| **취약점 코드** | W-011 |
| **위험도** | 중 |
| **취약점 개요** | ㅇ 크로스사이트 스크립팅(Cross Site Scripting)은 공격자가 전달한 JavaScript, VBScript등과 같이 사용자 브라우저에서 실행되는 코드가 HTTP 응답으로 사용자에게 전달될 경우 코드가 실행되어 사용자 쿠키∙세션정보 탈취, 브라우저 흐름제어 등과 같은 악성행위를 유도할 수 있는 취약점  시스템을 공격하는 것이 아니라 클라이언트 측 사용자를 공격하는 기법이며, 아래 두 가지 방식으로 구분  Case 1) 저장(Stored) 방식  메시지 입력기능(방명록, 댓글 등)을 통해 악성 스크립트를 시스템 상에 저장 시키고 사용자가 해당 기능에 접근하면 스크립트가 실행되는 구조  Case 2) 반사(Reflective) 방식  URL의 파라미터 위치에 스크립트를 삽입하여 링크를 생성 후 이메일 등을 통해 해당 링크를 전송하여 사용자가 해당 링크를 클릭했을 경우 스크립트가 실행되는 구조 |
| **보안설정방법** | ㅇ 사용자로부터 입력 가능한 문자를 제한하고 이외의 문자들은 필터링 필요  - 필터링 대상은 GET 질의 문자열, POST 데이터, 쿠키, URL 그리고 브라우저와 웹 서버가 주고 받는 모든 데이터를 포함  ㅇ 다음의 두 가지 유형에 대해 모두 필터링을 적용해야 함  1) 사용자 입력 값이 데이터베이스에 저장되어 해당 값이 웹 브라우저에 노출되는 경우 (데이터베이스에 저장 전 필터링)  2) 사용자 입력 값이 데이터베이스에 저장되지 않고 직접적으로 웹 브라우저에 노출되는 경우  (예) 사용자 입력한 검색어로 게시물 검색 시 해당 검색어가 웹 브라우저에 노출  문자열을 필터링 할 때는 우회 방지를 위해 소문자로 치환하여 필터링 하도록 해야 하며, Client Side Script (JavaScript, VBScript)가 아닌 Server Side Script (JSP, PHP, ASP 등)로 구현해야 함  ※ 참고: 크로스사이트 스크립팅에 대한 대응책  1) 게시판에서 HTML 포맷을 사용할 수 없도록 설정  2) 일반적으로 사용되는 특수문자를 소스레벨에서 치환처리   |  |  | | --- | --- | | **기호** | **URL 인코딩** | | < | &lt; | | > | &gt; | | “ | &quot; | | ‘ | &#39; | | Space(공백) | &nbsp; | | ( | &#40; | | ) | &#41; | | # | &#35; | | & | &#38; |   3) 게시판을 비롯 단문이상의 모든 입력에서 HTML 포맷의 사용이 필요한 경우 허용된 태그만 사용이 가능하도록 설정해야 함  ※ 참고: 유해 문자   |  |  |  |  |  | | --- | --- | --- | --- | --- | | javascript | eval | onmousewheel | onactive | innerHTML | | expresson | charset | ondatavailable | oncut | onfocusout | | applet | document | onafterinpudate | onclick | onkeyup | | meta | string | onmousedown | onchange | onkeypess | | xml | create | onbeforeacivate | onbeforecut | onload | | blink | append | onbeforecopy | ondbclick | onbounce | | link | binding | onbeforedeacivate | ondeactivate | onmouseenter | | style | alert | ondatasetchaged | ondrag | onmouseout | | script | msgbox | cnbeforeprint | ondragend | onmouseover | | embed | refresh | cnbeforepaste | ondagenter | onsubmit | | object | vold | onbeforeeditfocus | ondagleave | onmouseend | | iframe | cookie | onbeforeuload | ondragover | onresizestart | | frame | href | onbeforeupdate | ondragstart | onuload | | frameset | onpaste | onpropertychange | ondrop | onselectstart | | ilayer | onresize | ondatasetcomplete | onerror | onreset | | layer | onselect | oncellchange | onfinish | onmove | | bgsound | base | onlayoutcomplete | onfocus | onstop | | title | onblur | onselectionchange | vbscript | onrowexit | | onbefore | onstart | onrowsinserted | onkeydown | onerrorupdate | | onmouseup | onfousin | oncontrolseleced | onrowsdelete | onfilterchage | | onrowenter | onhelp | onreadystatechange | onmouseleave | onlosecapture | | oncontextmenu | onmousemove | atob | Function | location | | replace | attr | assign | navigate | - |   4) 웹 어플리케이션 언어 별 특정 함수를 이용하여 XSS에 악용될 소지가 있는 태그를 제거하거나 특정문자로 치환  ■ JSP   |  | | --- | | <%  String search= request.getParameter("search");  search = search.replaceAll("<","&lt;"); // 유해문자 치환  search = search.replaceAll(">","&gt;");  search = search. replaceAll("\"","&quot;");  search = search. replaceAll("\’","&#39;");  search = search. replaceAll(" ","&nbsp;");  // 이외 유해문자를 모두 치환해야 함  %>  <input type="text" name="search" value="<%=search%>"> |   ※ 참고: JSTL core 사용시 출력 값 c:out 처리   |  | | --- | | <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>  ..  <input type="hidden" name="searchWord" value="<c:out value='${value}'/>"/>  … |   ■ PHP  ※ 참고: HTML 태그불허: htmlspecialchars 함수 이용   |  | | --- | | //xss mitigation functions  function xssafe($data,$encoding='UTF-8'){  return htmlspecialchars($data,ENT\_QUOTES | ENT\_HTML401,$encoding);  }  function xecho($data){  echo xssafe($data);  }  //usage example  <input type='text' name='test' value='<?php xecho ("' onclick='alert(1)");  ?>' /> |   ※ 참고: HTML 태그 일부 허용   |  | | --- | | // 압축해제  require\_once('/경로/htmlpurifier/library/HTMLPurifier.auto.php');  // 기본 설정을 불러온 후 커스터마이징  $config = HTMLPurifier\_Config::createDefault();  $config->set('Attr.EnableID', false);  $config->set('Attr.DefaultImageAlt', '');  // 인터넷 주소를 자동으로 링크로 바꿔주는 기능  $config->set('AutoFormat.Linkify', true);  // 이미지 크기 제한 해제  $config->set('HTML.MaxImgLength', null);  $config->set('CSS.MaxImgLength', null);  // EUC-KR인 경우 iconv로 UTF-8 변환  $config->set('Core.Encoding', 'UTF-8');  // 필요에 따라 DOCTYPE 바꾸어 사용  $config->set('HTML.Doctype', 'XHTML 1.0 Transitional');  // 플래시 삽입 허용  $config->set('HTML.FlashAllowFullScreen', true);  $config->set('HTML.SafeEmbed', true);  $config->set('HTML.SafeIframe', true);  $config->set('HTML.SafeObject', true);  $config->set('Output.FlashCompat', true);  // iframe 동영상 삽입 허용  $config->set('URI.SafeIframeRegexp', '#^(?:https?:)?//(?:'.implode('|', array(  'www\\.youtube(?:-nocookie)?\\.com/',  'maps\\.google\\.com/',  'player\\.vimeo\\.com/video/',  'www\\.microsoft\\.com/showcase/video\\.aspx',  '(?:serviceapi\\.nmv|player\\.music)\\.naver\\.com/',  '(?:api\\.v|flvs|tvpot|videofarm)\\.daum\\.net/',  'v\\.nate\\.com/',  'play\\.mgoon\\.com/',  'channel\\.pandora\\.tv/',  'www\\.tagstory\\.com/',  'play\\.pullbbang\\.com/',  'tv\\.seoul\\.go\\.kr/',  'ucc\\.tlatlago\\.com/',  'vodmall\\.imbc\\.com/',  'www\\.musicshake\\.com/',  'www\\.afreeca\\.com/player/Player\\.swf',  'static\\.plaync\\.co\\.kr/',  'video\\.interest\\.me/',  'player\\.mnet\\.com/',  'sbsplayer\\.sbs\\.co\\.kr/',  'img\\.lifestyler\\.co\\.kr/',  'c\\.brightcove\\.com/',  'www\\.slideshare\\.net/',  )).')#');  // 설정을 저장하고 필터링 라이브러리 초기화  $purifier = new HTMLPurifier($config);  // HTML 필터링 실행  $html = $purifier->purify($html); |   ■ ASP.NET  ※ 참고: 입력값 제한  1) ASP.NET 유효성 검사기 컨트롤 RegularExpressionValidator 및 RangeValidator 를 사용함.  2) 다른 소스의 입력을 통해 제공되는 입력값을 제한하려면 서버 쪽 코드에서 System.Text.RegularExpressions.Regex 클래스를 사용하여 정규식으로 예상되는 내용을 확인.  3) integers, doubles, dates, 통화 같은 형식을 확인하려면 입력 데이터를 동등한 .NET Framework 데이터 형식으로 변환하고 결과 변환 오류를 처리.   |  | | --- | | //RegularExpressionValidator: 텍스트 입력을 제한할 때 사용  //아래 예제는 입력 이름 필드를 영문자, 공백문자, ', . 로 제한함  <form id="WebForm" method="post" runat="server">  <asp:TextBox id="txtName" runat="server"></asp:TextBox>  <asp:RegularExpressionValidator id="nameRegex" runat="server"  ControlToValidate="txtName"  ValidationExpression="^[a-zA-Z'.\s]{1,40}$"  ErrorMessage="Invalid name">  </asp:regularexpressionvalidator>  </form>  //유효성 검사 컨트롤을 사용할 수 없거나, 입력 필드 이외의 입력 값에서 유효성을 검사해야 하는 경우 Regex 클래스를 사용.  // Instance method:  Regex reg = new Regex(@"^[a-zA-Z'.\s]{1,40}$");  Response.Write(reg.IsMatch(txtName.Text));  // Static method:  if (!Regex.IsMatch(txtName.Text,@"^[a-zA-Z'.\s]{1,40}$")) {  // Name does not match expression  } |   ※ 참고: 출력 인코딩   |  | | --- | | //  <%@ Page Language="C#" AutoEventWireup="true"%>  <html>  <form id="form1" runat="server">  <div>  Color:&nbsp;<asp:TextBoxID="TextBox1" runat="server">  </asp:TextBox><br />  <asp:Button ID="Button1" runat="server" Text="Show color"  OnClick="Button1\_Click" /><br />  <asp:Literal ID="Literal1" runat="server"></asp:Literal>  </div>  </form>  </html>  <script runat="server">  private void Page\_Load(Object Src, EventArgs e) {  protected void Button1\_Click(object sender, EventArgs e) {  Literal1.Text = @"<span style=""color:" + Server.HtmlEncode(TextBox1.Text)  + @""">Color example</span>";  } }  </Script> |   ■ ASP   |  | | --- | | <%  … 중략 …  If use\_HTML Then  content = Server.HTMLEncode(content)  … 중략 …  ub ReplaceStr(content, byref str)  content = replace(content, "'", " \'")  content = replace(content, "&", "&amp;")  content = replace(content, " \'", "&quot")  content = replace(content, "<", "&lt")  content = replace(content, ">", "&gt")  str = content  End Sub  … 중략 …  %> |   ㅇ 스프링프레임워크 기반 서비스에서 XSS 필터(lucy-xss-servlet-filter) 적용 시 Content-Type: multipart/form-data에 대한 필터 적용을 따로 설정해야함  ■ [web.xml] - spring framework에서 제공하는 CharacerEncodingFilter와 기존에 사용한 Filter 사이에 spring framework에서 제공하는 MultipartFilter를 추가   |  | | --- | | <filter>  <filter-name>MultipartFilter</filter-name>  <filter-class>org.springframework.web.multipart.support.MultipartFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>MultipartFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |   ■ [servlet-context.xml] - bean 추가   |  | | --- | | <beans:bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">  <beans:property name="defaultEncoding" value="UTF-8"/>  <beans:property name="maxUploadSize" value="40960000"/>  </beans:bean>  // servlet 버전이 3.0라면 id를 multipartResolver  // servlet 버전이 2.5라면 id를 filterMultipartResolver |   ■ tomcat 서버의 [context.xml] - Context 부분 수정   |  | | --- | | <Context allowCasualMultipartParsing="true" path="/">  <WatchedResource>WEB-INF/web.xml</WatchedResource> // 없어도 됨  <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource> // 없어도 됨  </Context> |   ※ 위의 코드는 점검자가 제시하는 샘플 코드이며 실제 업무 환경에서 적용하실 때는 용도와 현재 환경에 따라 개발자가 수정할 필요가 있으며, 충분한 테스트를 거친 후 사용해야 함 |

* 1. [W-013]불충분한 인증

|  |  |
| --- | --- |
| **구분** | **내용** |
| **취약점 코드** | W-013 |
| **위험도** | 중 |
| **취약점 개요** | ㅇ 민감한 데이터에 접근 가능한 경로에 대한 인증 절차가 불충분할 경우 발생하는 취약점으로 사용자가 권한 외의 페이지에 접근하여 정보를 유출하거나 변조할 수 있으므로 추가적인 인증 절차 필요 |
| **보안설정방법** | ㅇ 접근제어가 필요한 모든 페이지에 인증 절차 필요  ㅇ 하나의 프로세스가 여러 개의 페이지 또는 모듈로 이루어져 있을 때 권한 체크가 누락되는 경우를 방지하기 위해서 공통 모듈을 사용  ㅇ 실명인증 우회 대응방안  - 사용자가 회원가입 방법으로 인증수단(i-PIN) 선택  - 본인확인기관에서 실명인증을 받고 기관에서 제공하는 이용자 관련 정보(실명, 생년월일, 식별번호 등)를 웹 서버에 전달  - 전달받은 이용자 관련정보(실명, 생년월일, 식별번호 등)를 세션값에 저장  - 사용자는 회원가입에 필요한 정보를 입력  - 최종 회원가입 요청 시 세션 값에 저장된 이용자 관련 정보(실명)와 회원가입 페이지에서 작성한 실명을 비교하는 로직 구현  ㅇ Client Side Script (JavaScript, VBScript 등)를 사용하면 사용자가 임의로 수정할 수 있으므로 Server Side Script (JSP, PHP, ASP 등)를 통하여 인증 및 필터링 과정을 수행해야 함  ■ JSP  ※ 참고 1: 개인정보가 수정될 계정과 수정을 시도하는 계정 비교   |  | | --- | | ....  Public Boolean User\_Auth( HttpServletRequest request ) {  HttpSession session = request.getSession();  String USER\_ID = (String)session.getAttribute(“USER\_ID”);  // 개인정보 수정시도 계정의 세션 값(ID)을 가져옴  String USER\_PWD = request.getParameter(“USER\_PWD”);  // 재인증을 위한 화면에서 사용자에게 입력 받은 패스워드를 가져옴  String query = "select \* from TB\_01 where DB\_ID = ? and DB\_PWD = ?";  try {  con = ds.getConnection();  pstmt = con.prepareStatement(query);  pstmt.setString(1, USER\_ID);  pstmt.setString(2, USER\_PWD);  rs = pstmt.executeQuery();  rs.next();  String DB\_ID = rs.getString(“DB\_ID”);  // DB에서 개인정보가 수정될 계정을 가져옴  if( !(DB\_ID.equals(USER\_ID) ) {  // 개인정보가 수정될 계정과 수정을 시도하는 계정 비교  return false; // 불일치 시 false 반환  }  }  .... |   ※ 참고 2: 인증 후에 이용 가능한 페이지에서 인증 정보를 확인하는 예제   |  | | --- | | --- 인증 승인 ---  <%@ page contentType = “text/html;charset=euc-kr” %>  <%  //HttpSession session = request.getSession(true);  //form에서 사용자 id와 사용자 password를 아래 변수로 전달  if(!myfunc\_userauth(userid, userpw)) //DB에서 사용자 인증을 처리  out.println “인증 실패”;  LogSave(userid, 0); //접속에 실패한 ID 기록  else  session.putValue(“logged\_in”, “logok”);  session.putValue(“userid”, userid);  LogSave(userid); //접속한 사용자 ID 기록  --- 인증 승인 여부 확인(페이지마다 포함) ---  <%@page contentType=”text/html;charset=euc-kr” %>  //HttpSession session = request.getSession(true);  if(session.putValue(“logged\_in”).equals(“logok”)) //승인받은 사용자인지 검증  out.println “페이지에 접근할 수 없습니다”;  LogSave(userid, 0) //접속에 실패한 ID 기록  else  LogSave(userid); //접속한 사용자 ID 기록, 인증에 성공한 경우 처리해야 하는 부분 |   ※ 위의 코드는 점검자가 제시하는 샘플 코드이며 실제 업무 환경에서 적용하실 때는 용도와 현재 환경에 따라 개발자가 수정할 필요가 있으며, 충분한 테스트를 거친 후 사용해야 함 |

* 1. [W-018]불충분한 세션만료

|  |  |
| --- | --- |
| **구분** | **내용** |
| **취약점 코드** | W-018 |
| **위험도** | 하 |
| **취약점 개요** | ㅇ 세션 만료는 사용자 인증 후 웹 사이트 내에서 이벤트 없이 일정 시간이 경과하면, 임의로 세션을 종료시켜 서비스 접근을 제한하는 기능으로, 공개된 장소에서 사용자가 오랜 시간 자리를 비우는 경우 타 사용자에 의한 침해 발생 |
| **보안설정방법** | ㅇ 세션 타임아웃 구현 시 타임아웃 시간은 10분으로 설정할 것을 권고함  (운영 시스템에 따라 타임아웃 시간 변경하여 적용)   ASP  접속자 별로 세션을 생성하여 사용자 정보를 각각 저장할 수 있는 \*세션 오브젝트를 사용하여 타임아웃 기능을 구현함  - 세션 오브젝트: 페이지 접근을 허가하거나 금지할 때 또는, 사용자 별로 정보를 저장할 때 많이 사용하며 접속자의 브라우저에서 쿠키 기능을 지원해야 세션 오브젝트 사용이 가능  다음과 같은 설정이 적용될 경우 사용자가 로그아웃 할 경우 세션은 바로 삭제되며 로그아웃 하지 않고 10분 동안 웹 서버로의 요청이 없을 경우 세션은 없어지게 됨   |  | | --- | | … 중략 …  // Session의 유지 시간 Setting  Session.timeout = 10  … 중략 … |  |  |  |  | | --- | --- | --- | | **구분** | **명칭** | **설명** | | Property | SessionID | 사용자마다 갖게 되는 고유한 세션값 | | Timeout | 세션이 유지되는 기간 | | Method | Abandon | 강제로 세션을 소멸시키는 함수 | | Event | Onstart | 각각의 사용자가 처음 방문할 때 발생 | | Onend | 사용자의 세션이 끝나는 시점에 발생 |    JSP  세션 타임아웃 기능을 구현하는 방법은 session.getLastAccessedTime()를 이용하여 세션의 마지막 접근 시간으로부터 일정 시간 이내에 다시 세션에 접근하지 않은 경우 자동으로 세션을 종료하도록 함.  세션의 타임아웃 설정방법  1) web.xml 파일에서 <session-config> 태그를 사용하여 타임아웃을 지정하는 방법  - web.xml, Weblogic.xml 중 한 곳에만 설정 (만약, 두 곳 모두 설정 시 우선 순위에 의해 web.xml의 설정 적용)  ※ 참고: Web.xml : “분” 단위   |  | | --- | | <session-config>  <session-timeout>10</session-timeout>  </session-config> |   ※ 참고: Weblogic.xml: “초” 단위   |  | | --- | | <session-descriptor>  <timeout-secs>600</timeout-secs>  </session-descriptor>  또는,  <session-param>  <param-name>TimeoutSecs</param-name>  <param-value>600</param-value>  </session-param> |   2) 세션 기본 객체가 제공하는 setMaxInactiveInterval() 메소드 사용  - 주의할 점: web.xml에서는 타임아웃 시간단위가 분이지만 메소드에서는 초단위임   |  | | --- | | … 중략 …  // Session의 유지 시간을 Setting  String strTime = Param.getPropertyFromXML("SessionPersistenceTime");  if (strTime == null) {  session.setMaxInactiveInterval(600);  } else {  session.setMaxInactiveInterval((new Integer(strTime)).intValue());  }  … 중략 … |   ※ 참고: Spring boot : “초” 단위   |  | | --- | | application.properties 또는 application.yml 파일 내 코드 추가  # 스프링 부트 1.x  server:  session:  timeout: 600  # 스프링 부트 2.x  server:  servlet:  session:  timeout: 600 |   ※ 참고: 참고: Node.js : “ms” 기준   |  | | --- | | 프로젝트 엔트리 포인트 파일 ex) index.js 또는 app.js 파일 내 코드 추가  var session = require('express-session');  app.use(session({  secret: '비밀키 입력',  resave: false,  saveUninitialized: true  cookies: {  maxAge: 600000 // 1000 \* 60 \* 10 (10분)  }));  // 세션 초기 설정  app.get('/', function(req, res) {  sess = req.session;  });  // 세션 변수 설정  app.get('/login', function(req, res) {  sess = req.session;  sess.username = "사용자 ID"  });  // 세션 변수 사용  app.get('/', function(req, res) {  sess = req.session;  console.log(sess, username);  }); |   ※ 위의 코드는 점검자가 제시하는 샘플 코드이며 실제 업무 환경에서 적용하실 때는 용도와 현재 환경에 따라 개발자가 수정할 필요가 있으며, 충분한 테스트를 거친 후 사용해야 함 |

* 1. [W-019]세션고정

|  |  |
| --- | --- |
| **구분** | **내용** |
| **취약점 코드** | W-019 |
| **위험도** | 하 |
| **취약점 개요** | ㅇ 로그인 시 새로운 세션 ID가 발행되지 않는다면 세션 ID를 도용한 비인가자의 접근 및 권한 우회 가능 |
| **보안설정방법** | ㅇ 조작하여 요청 보낼 수 없도록 URL rewrite 기능을 사용하지 않고, 로그인 성공 시 세션 ID 를 재할당함  ㅇ 로그아웃 시 기존에 존재하는 세션을 받드시 삭제해야함  ■ JSP  ※ 참고: 로그인 시 기존 세션 파기 후 새로운 세션 발행   |  | | --- | | ....  //로그인 성공 시  session.invalidate(); //기존 세션 삭제  session = request.getSession(true);  session.setAttribute("ID", DB\_id);  .... |   ※ 참고: 로그아웃 시 세션 값 삭제   |  | | --- | | ....  session.invalidate();  .... |   ※ 위의 코드는 점검자가 제시하는 샘플 코드이며 실제 업무 환경에서 적용하실 때는 용도와 현재 환경에 따라 개발자가 수정할 필요가 있으며, 충분한 테스트를 거친 후 사용해야 함. |

* 1. [W-129]취약한 HTTPS 암호 알고리즘 이용

|  |  |
| --- | --- |
| **구분** | **내용** |
| **취약점 코드** | W-129 |
| **위험도** | 하 |
| **취약점 개요** | ㅇ 표준화되지 않은 암호 알고리즘을 사용하는 것은 공격자가 알고리즘을 분석하여 무력화시킬 수 있는 가능성 존재  ㅇ 컴퓨터의 성능이 향상됨에 따라 취약으로 분류되어 짧은기간(며칠 혹은 몇 시간) 내에 해독되어 정보가 유출 가능  ㅇ 취약점 조치를 위해 설정 변경 시 반드시 정보보안팀 SSL암/복호화 담당자인 박병용 님에게 연락 후 변경 必  ※ 운영/개발 모두 사전 연락 후 진행 |
| **보안설정방법** | ※ 참고: CBC-Mode 조치 시 SSL-handshake 에러 발생할 경우 Java 버전 확인 필요  o jdk-1.7 이하 버전의 경우 GCM 모드 확장이 필요하며 local\_policy.jar 업데이트 필요  → jdk-1.7 버전의 경우 GCM 암호화 모드를 지원하지 않아 CBC모드를 차단할 시 SSL-handshake 에러 발생  \* policy download URL: https://www.oracle.com/java/technologies/javase-jce7-downloads.html   보안설정방법  ※ 해당 보안 설정 방법은 범용적으로 사용되는 Windows, Linux(Apache) 기반으로 작성되었으며 별도의 상용 솔루션으로 운영할 경우, 보안설정 방법이 상이 할 수 있어 정확한 보안설정방법은 운영중인 솔루션 제조사 혹은 유지보수사에 확인 후 진행  ※ 서비스에 따라 Cipher 차단할 경우 홈페이지에 접속 안되는 장애 사례가 있으므로 확인 후 서버 반영 필요  1. 프로토콜 알고리즘 정책  ※ 취약한 알고리즘   |  | | --- | | 1. 64-bit block cipher 3DES vulnerable to SWEET32 attack  - TLS\_DHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (dh 1024)  - TLS\_ECDHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (secp256r1)  - TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (rsa 2048)  2. 64-bit block cipher IDEA vulnerable to SWEET32 attack  - TLS\_RSA\_WITH\_IDEA\_CBC\_SHA (rsa 2048)  3. Broken cipher RC4 is deprecated by RFC 7465  - TLS\_ECDHE\_RSA\_WITH\_RC4\_128\_SHA (secp256r1)  - TLS\_RSA\_WITH\_RC4\_128\_MD5 (rsa 2048)  - TLS\_RSA\_WITH\_RC4\_128\_SHA (rsa 2048)  4. Ciphersuite uses MD5 for message integrity  - TLS\_RSA\_WITH\_RC4\_128\_MD5 (rsa 2048)  5. block cipher mode CBC vulnerable  - CBC 블록 암호화를 사용하는 모든 암호는 제거  ex) TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA256 (dh 2048)  6. CBC-mode cipher in SSLv3 (CVE-2014-3566) (취약한 HTTPS 프로토콜에 포함)  - SSLv3 프로토콜 사용 중지 |   ※ 사용을 권장하는 알고리즘  (서비스에 따라 허용 및 차단에 필요한 Cipher 확인하여 적용 필요)   |  | | --- | | - TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384  - TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 |   ※ DHE 알고리즘 사용 권고  Diffie-Hellman(DH) 알고리즘은 웹서버 환경인 Openssl 및 Java 버전에 따라 취약한 bit를 사용하게 되며, 1024bit는 취약한 알고리즘이므로 대체할 Ephemeral Diffie-Hellman(EDH) 알고리즘 등의 고강도 암호화 키 교환 알고리즘의 사용을 권장함   |  |  | | --- | --- | | OPENSSL | Java | | OpenSSL 1.0.2 : openssl 1.0.1n 이상 OpenSSL 1.0.1 : openssl 1.0.2b 이상 | JDK 1.8 이상 |   2. 권장 알고리즘 설정  아래의 예시를 참고하여 설정한 뒤 웹 서비스를 재시작하여 적용  [Apache]  1) $(HOME\_BASE)/conf/httpd-ssl.conf 파일을 열어 프로토콜을 아래와 같이 설정   |  | | --- | | <VirtualHost \*:443>  SSLProtocol –ALL +TLSv1.2  SSLHonorCipherOrder On  SSLInsecureRenegotiation off |   2) $(HOME\_BASE)/conf/httpd-ssl.conf 파일을 열어 알고리즘을 아래와 같이 설정   |  | | --- | | SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384  :!aNULL:!eNULL:!MEDIUM:!LOW:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS:!RC4:!SEED:!CBC; |   [Tomcat]  1) $(HOME\_BASE)/conf/server.xml 파일을 열어 프로토콜을 아래와 같이 설정   |  | | --- | | # TLSv1.2 이상 프로토콜은 Tomcat 버전과 관련없이 jdk 1.7.x 버전 이상 필요  <Connector port="443" protocol="org.apache.coyote.http11.Http11Protocol"  maxThreads="150" SSLEnabled="true" scheme="https" secure="true"  clientAuth="false" sslProtocol ="TLS" sslEnabledProtocols = "TLSv1.2"/> |   2) $(HOME\_BASE)/conf/server.xml 파일을 열어 알고리즘을 아래와 같이 설정   |  | | --- | | ciphers="TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384,TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256" |   [IIS7 이상]  1) 시작 > 실행 > ‘gpedit.msc’ 실행  2) 컴퓨터 구성 > 관리 템플릿 > 네트워크 > SSL 구성 설정으로 이동  텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다.  3) SSL 암호 그룹 순서(SSL Cipher Suite Order)를 두 번 클릭  4) SSL 암호 그룹 순서(SSL Cipher Suite Order)창에서 사용을 클릭  5) 옵션: SSL 암호 그룹에 아래의 Cipher Suite 값을 입력   |  | | --- | | - TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384  - TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 |   [WebtoB]  1) $(HOME\_BASE}/config/http.m 파일을 열어 프로토콜을 아래와 같이 설정   |  | | --- | | \*SSL  ucertssl CertificateFile = "/usr/local/webtob/config/ssl/test.co.kr.crt",  CertificateKeyFile = "/usr/local/webtob/config/ssl/test.co.kr.key",  CACertificateFile = "/usr/local/webtob/config/ssl/test.co.kr.root",  CertificateChainFile ="/usr/local/webtob/config/ssl/test.co.kr.ca",  Protocols = "-SSLv2, -SSLv3, -TLSv1, -TLSv1.1, TLSv1.2" |   2) $(HOME\_BASE}/config/http.m 파일을 열어 알고리즘을 아래와 같이 설정   |  | | --- | | RequiredCiphers =  "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384  :!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!SSLv2:!SRP:!ADH:!AECDH:!EXP:!IDEA:!3DES:!CBC" |   3) http.m 설정 후 http.m 컴파일 및 WetoB 재기동   |  | | --- | | $ wscfl -i http.m # http.m 컴파일  $ wsdown # WebtoB 종료  $ wsboot # WebtoB 기동 |   [Nginx]  1) $(HOME\_BASE)/nginx.conf 파일을 열어 프로토콜을 아래와 같이 설정   |  | | --- | | ssl\_protocols TLSv1.2 TLSv1.3; # TLSv1.3은 Nginx 버전이 1.17.0 이상인 경우에 사용 |   2) $(HOME\_BASE)/nginx.conf 파일을 열어 알고리즘을 아래와 같이 설정   |  | | --- | | ssl\_ciphers ECDHE-RSA-AES128-GCM-SHA256: ECDHE-RSA-AES256-GCM-SHA384:  !aNULL:!eNULL:!MEDIUM:!LOW:!3DES:!MD5:!EXP:!PSK:!SRP:!DSS:!RC4:!SEED:!CBC;  ssl\_prefer\_server\_ciphers on; |   [AWS]  1) Application Load Balancer (ALB) 사용할 경우   |  | | --- | | - ELBSecurityPolicy-FS-1-2-Res-2020-10 정책 사용을 권고함  → 해당 정책은 허용되는 알고리즘이 최소로 제한되어 있으며, 서비스 이용 측면에서 적용이 불가할 경우, AWS에서 \*권장하고 있는 정책 사용을 권고함  \* ELBSecurityPolicy-TLS13-1-2-2021-06 |   2) Classic Load Balancer (CLB) 사용할 경우  - Load balancer 선택 후 Actions>Edit listener 클릭  - Cipher Change 시도  텍스트, 폰트, 스크린샷이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다.  - Custom Security Policy 선택 후 사용 \*권장하는 알고리즘만 체크 후 저장  \* 권장 알고리즘  ECDHE-ECDSA-AES256-GCM-SHA384, ECDHE-ECDSA-AES128-GCM-SHA256,  ECDHE-RSA-AES256-GCM-SHA384, ECDHE-RSA-AES128-GCM-SHA256  텍스트, 소프트웨어, 웹 페이지, 웹사이트이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다.  [Spring Boot (embedded Tomcat)]  1) Spring Boot 표준 설정방법 (Spring Boot 2.3.0 기준)  - application.properties 또는 application.yml 파일을 열거나 생성  - 아래 속성을 추가하여 암호화 알고리즘 지정  (지원하는 암호화 알고리즘은 내장 웹 서버별로 상이할 수 있음)   |  | | --- | | # EMBEDDED SERVER CONFIGURATION (ServerProperties)  server.ssl.ciphers=TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256,TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384  # SSL protocol to use  server.ssl.protocol=TLS  # Enabled SSL protocols  server.ssl.enabled-protocols=TLSv1.2 |   - 파일 저장 후 서비스 재시작  2) WebServerFactoryCustomizer 인터페이스를 사용할 경우 (Spring Boot 2.1이후)   |  | | --- | | import org.apache.catalina.connector.Connector;  import org.apache.coyote.http11.AbstractHttp11Protocol;  import org.springframework.boot.web.embedded.tomcat.TomcatConnectorCustomizer;  import org.springframework.boot.web.embedded.tomcat.TomcatServletWebServerFactory;  import org.springframework.boot.web.server.WebServerFactoryCustomizer;  import org.springframework.context.annotation.Bean;  import org.springframework.context.annotation.Configuration;  @Configuration  public class HttpsConfiguration {  @Bean  public WebServerFactoryCustomizer<TomcatServletWebServerFactory>  servletContainerCustomizer() {  return new WebServerFactoryCustomizer<TomcatServletWebServerFactory>() {  @Override  public void customize(TomcatServletWebServerFactory factory) {  factory.addConnectorCustomizers(new TomcatConnectorCustomizer() {  @Override  public void customize(Connector connector) {  AbstractHttp11Protocol<?> httpHandler = ((AbstractHttp11Protocol<?>) connector.getProtocolHandler());  httpHandler.setUseServerCipherSuitesOrder(true);  httpHandler.setSSLProtocol("TLSv1.2");  httpHandler.setSSLHonorCipherOrder(true);  httpHandler.setCiphers("TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256,  TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384"); |   3. 적용 확인 방법  Nmap(네트워크 보안 스캐너)을 이용한 확인 방법  1) URL(https://nmap.org/download.html)에서 프로그램을 다운로드  2) Nmap을 실행한 후에 ‘Command' 창에 명령어를 입력   |  | | --- | | nmap -sV --script ssl-enum-ciphers -p <포트번호> <host> |   3) 아래와 같이 적용 결과 알고리즘과 함께 사용중인 프로토콜 확인 가능  (참고 : CBC-mode의 경우 보안 취약 사항이 확인됨에 따라 자체적으로 C rank로 설정된 부분이므로 스캔 결과가 상이할 수 있음)   |  | | --- | | PORT STATE SERVICE VERSION  443/tcp open ssl/ssl Apache httpd (SSL-only mode)  |\_http-server-header: Apache  | ssl-enum-ciphers:  | TLSv1.2:  | ciphers:  | TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (rsa 2048) - A  | TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384 (rsa 2048) - A  | TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (rsa 2048) - C  | compressors:  | NULL  | cipher preference: server  | warnings:  | block cipher mode CBC vulnerable  | 64-bit block cipher 3DES vulnerable to SWEET32 attack | |

* 1. [W-233]크리덴셜 스터핑

|  |  |
| --- | --- |
| **구분** | **내용** |
| **취약점 코드** | W-233 |
| **위험도** | 하 |
| **취약점 개요** | ㅇ 공격자가 다른 웹사이트나 서비스에서 유출된 유효한 인증정보를 수집하고 이를 기반으로 타겟 서비스에 불법적으로 접근하는 공격 기법  1) A 사이트의 계정정보(ID, PW) 유출  2) 다크웹 등에서 유출된 계정정보를 공격자가 수집  3) B 사이트에 2)에서 수집한 계정정보 무차별 대입  → A, B 사이트에서 동일한 계정정보를 사용하는 사용자의 계정 탈취  ㅇ 자동화 도구를 이용하여 특정 IP주소에서 짧은 시간 내에 대량의 인증정보를 대입(로그인)함으로써 사용자 계정을 탈취하는 취약점으로, 동일한 IP 주소에서 무차별적 로그인 시도에 대한 차단 정책의 부재로 발생 |
| **보안설정방법** | ㅇ 반복 요청을 차단하는 로직 적용  - 아래와 같이 일회성 토큰을 생성하여 검증하는 방법을 사용할 수 있으며, 로그인 요청이 들어오면 사용자가 제공한 토큰(providedToken)과 세션에 저장된 토큰(storedToken)을 비교하여 일치 여부를 확인하고 결과에 따라 처리를 수행함  ■ JSP  ※ 참고: 로그인 시 일회성 토큰 생성 예시   |  | | --- | | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>  <%@ page import="java.util.UUID" %>  <%  //일회성 토큰 생성  String generateOneTimeToken() {  String token = UUID.randomUUID().toString();  //생성된 토큰을 세션에 저장  session.setAttribute("oneTimeToken", token);  return token;  }  //로그인 요청 처리  if (request.getMethod().equals("POST")) {  String username = request.getParameter("username");  String password = request.getParameter("password");  String providedToken = request.getParameter("token");  String storedToken = (String) session.getAttribute("oneTimeToken");  if (providedToken != null && providedToken.equals(storedToken)) {  //토큰이 일치하는 경우, 사용자 인증 로직 수행  // ...  //인증 성공 시 세션에 사용자 정보 저장  session.setAttribute("username", username);  //로그인 성공 후에는 일회성 토큰을 제거  session.removeAttribute("oneTimeToken");  //로그인 성공 후 리다이렉트 또는 다음 동작 수행  response.sendRedirect("dashboard.jsp");  } else {  //토큰이 일치하지 않는 경우, 로그인 실패 처리  // ...  //로그인 실패 시 메시지를 표시하거나 다시 로그인 페이지로 리다이렉트  response.sendRedirect("login.jsp?error=invalid\_token");  }  }  %> |   - 로그인 시도가 있을 때마다 횟수를 증가시키고, 일정 횟수 이상이 되면 자동화 공격으로 인식하여 로그인을 할 수 없도록 차단하는 로직을 적용  ■ JSP  ※ 참고: 프로세스 호출 횟수 제한 예시   |  | | --- | | String callProcess = “test.do”; //호출된 프로세스 이름  ….  rs = pstmt.executeQuery();  if(rs.next()){  chknum = rs.getString(callProcess);  }else{  //쿼리 에러  }  if(chknum >= 5){  //제한  }else{  //프로세스 호출  } |   - 동일한 IP주소에서 다량의 로그인 등의 불법적인 로그인 시도가 발생할 경우, 서버와의 통신을 임의의 시간을 부여하여 지연하거나 차단  - 사용자에게 주기적으로 패스워드 변경을 요청하며, 타 사이트와 동일한 패스워드를 사용하지 않도록 고지 등 알림  ㅇ 대량의 로그인을 시도하는 IP를 차단하는 방화벽 정책을 구현  ㅇ 추가 인증수단(OTP인증, SMS인증, 이메일인증)을 이용한 다단계 인증 적용  ㅇ 보안문자(CAPTCHA) 적용을 통해 자동화된 로그인 시도를 사전에 예방  - Google reCAPTCHA v3 적용 가이드  https://developers.google.com/recaptcha/docs/v3?hl=ko  - Google reCAPTCHA 사이트(https://www.google.com/recaptcha/admin/create)에서 reCAPTCHA v3를 등록하고 사이트 키를 발급받도록 하며, 발급받은 사이트 키는 아래 코드의 YOUR\_SITE\_KEY 부분에 해당함  ※ (주의) Captcha 사용 시 Captcha 인증을 보호할 프로세스와 분리하지 말아야 함. 분리할 경우 Captcha 인증 뒤에 호출되는 프로세스를 직접 호출할 수 있으므로 보호 효과가 없음  ■ JSP  ※ 참고: reCAPTCHA v3 적용 예시   |  | | --- | | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page import="java.net.\*, java.io.\*"  %>  <%  //생성된 토큰 받음  String g\_recaptcha\_response = request.getParameter("g-recaptcha-response");  System.out.println(g\_recaptcha\_response);    //토큰과 보안키를 가지고 성공 여부를 확인  HttpURLConnection conn = (HttpURLConnection) new URL("https://www.google.com/recaptcha/api/siteverify").openConnection();  String params = "secret=보안키입력" + "&response=" + g\_recaptcha\_response;  conn.setRequestMethod("POST");  conn.setDoOutput(true);  DataOutputStream wr = new DataOutputStream(conn.getOutputStream());  wr.writeBytes(params);  wr.flush();  wr.close();    //결과코드 확인(200 : 성공)  int responseCode = conn.getResponseCode();  StringBuffer responseBody = new StringBuffer();  if (responseCode == 200) {    //데이터 추출  BufferedInputStream bis = new BufferedInputStream(conn.getInputStream());  BufferedReader reader = new BufferedReader(new InputStreamReader(bis));  String line;  while ((line = reader.readLine()) != null) {  responseBody.append(line);  }  bis.close();    //JSON으로 변환하여야 하지만 기본 모듈에서 처리하기 위하여 아래와 같이 진행  if(responseBody.toString().indexOf("\"success\": true") > -1)  out.println("인증 되었습니다.");  }    %>  <html>  <head>  <script src="https://www.google.com/recaptcha/api.js?onload=onloadCallback&render=explicit" async defer></script>  <title>google, reCAPTCHA - reCAPTCHA2 Type</title>  <script type="text/javascript">  var correctCaptcha = function(response) {  console.log();  };  var onloadCallback = function() {  grecaptcha.render('html\_element', {  'sitekey' : '사이트키입력',  'callback' : correctCaptcha  });  };  </script>  </head>  <body>  <form action="" method="get">  <div id="html\_element"></div>  <input type="submit" value="Submit">  </form>  <!-- JSON String 찍기 -->  <%=responseBody != null ? responseBody.toString() : ""%>  </body>  </html> |   ■ PHP  ※ 참고: reCAPTCHA 토큰 확인 예시   |  | | --- | | $url = "https://www.google.com/recaptcha/api/siteverify?secret=보안키입력&  response={$captcha}";  $ch = curl\_init();  curl\_setopt($ch, CURLOPT\_URL, $url);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);  curl\_setopt($ch, CURLOPT\_CONNECTTIMEOUT, $timeout);  $g = curl\_exec($ch);  curl\_close($ch);  $g\_cvt = json\_decode($g);  //메일 전송  if ($g\_cvt->success==true) {  //정상 처리 메시지  } else {  //잘못된 접근 메시지  } |   ※ 위의 코드는 점검자가 제시하는 샘플 코드이며 실제 업무 환경에서 적용하실 때는 용도와 현재 환경에 따라 개발자가 수정할 필요가 있으며, 충분한 테스트를 거친 후 사용해야 함 |

**Shinsegae I&C는 고객의 정보보호를 위해 최선을 다합니다.**