**모의 해킹 결과 보고서**

작성자: 정재호

작성일: 2024.11.05

0. 목차

[1. 모의해킹 수행 정보 3](#_Toc182421188)

[1.1. 개요 3](#_Toc182421189)

[1.2. 대상 3](#_Toc182421190)

[1.3. 수행 기간 3](#_Toc182421191)

[1.4. 진행 방식 4](#_Toc182421192)

[1.5. 수행 인력 4](#_Toc182421193)

[1.6. 점검 도구 4](#_Toc182421194)

[2. 모의해킹 결과 5](#_Toc182421195)

[2.1. 총평 5](#_Toc182421196)

[2.2. 결과 요약 5](#_Toc182421197)

[3. 취약점 상세 5](#_Toc182421198)

[3.1. 로그인 시스템 5](#_Toc182421199)

[3.2. 검색 시스템 5](#_Toc182421200)

[4. 보안 권고안 5](#_Toc182421201)

# 1. 모의해킹 수행 정보

## 1.1. 개요

도표, 라인, 원, 디자인이(가) 표시된 사진

자동 생성된 설명

본 프로젝트는 웹 어플리케이션에 대한 모의 해킹을 수행한 후 **발견된 취약점에 대한 대응 방안을 제시**, 차후 발생할 수 있는 **침해 사고를 예방하고 보안 수준을 향상할 수 있는 대책을 수립**하는데 기여하는 것을 목적으로 한다.

## 1.2. 대상

본 프로젝트의 점검 대상은 아래와 같다.

|  |  |  |
| --- | --- | --- |
| **No** | **Platform** | **Service (IP)** |
| 1 | WEB | Juice Shop (<https://juice-shop.herokuapp.com/#/>) |

## 1.3. 수행 기간

모의 해킹 기간: 2024년 11월 05일 ~ 2024년 11월 12일 (6일)

## 1.4. 진행 방식

원, 도표, 스크린샷이(가) 표시된 사진

자동 생성된 설명

웹 서비스의 로그인 시스템, 검색 시스템, 그 외 시스템에 대한 사전 자료를 공유하고 진행하는 화이트 박스 방식으로 진행

## 1.5. 수행 인력

|  |  |  |
| --- | --- | --- |
| **Team** | **Name** | **Task** |
| Korea IT/ DevSecOps/ Team Firewalls. | 정재호  (Jae-ho Jung) | - 웹 서비스 모의 해킹  - 결과 보고서 작성 |

## 1.6. 점검 도구

|  |  |  |  |
| --- | --- | --- | --- |
| 새, 그래픽, 디자인이(가) 표시된 사진  자동 생성된 설명 | Nmap.org: /images/ | One of the best Burp Suite proxy server | 원, 다채로움, 그래픽, 그래픽 디자인이(가) 표시된 사진  자동 생성된 설명 |
| Kali Linux | Nmap | Burp Suite | Chrome |

# 2. 모의 해킹 결과

## 2.1. 총평

## 2.2. 결과 요약

# 3. 취약점 상세

## 3.1. Login page (http://ip:port/#/login)

### 3.1.1. SQL Injection 취약점

|  |  |  |
| --- | --- | --- |
| **구분** | **설명** | |
| **정보** | **URL** | **메뉴** |
| http://ip:port/#/login | 메인 페이지> Account> Login |
| **취약점 상세** | 로그인 시 사용하는 이메일 입력 필드에 SQL 명령어 주입 취약점이 있음을 확인했습니다. 이를 이용해 공격자는 웹 서비스에서 원하는 정보를 획득하거나 관리자 계정을 획득하여 정보를 위. 변조하여 2차 피해에 악용될 수 있습니다. | |

1) 로그인 페이지에 접속한다

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명.

2) 이메일 입력 필드에 (‘)를 입력해 본다.

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

자동 생성된 설명

* SQL Injection 취약점 확인

3) 간단한 SQL 구문(‘ or 1=1 --) 입력

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 모자, 스크린샷, 인간의 얼굴이(가) 표시된 사진

자동 생성된 설명

- 관리자 계정으로 로그인 되는 것을 확인했다.

4) SQL 구문을 수정해 관리자 계정 외 다른 계정으로 로그인 시도

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

- 웹 서비스의 임의의 사용자 계정으로 로그인

[보안 권고안]

1. 사용자의 입력을 파라미터로 사용할 때 문자열이 명령어로 인식되지 않도록 해야 합니다.
2. 에러 메시지를 노출하지 않도록 해야 합니다.
3. 최소 권한 원칙을 준수해야 합니다.

## Search Field(http://ip:port/#/search)

### 3.2.1. XSS(크로스 사이트 스크립트) 취약점

|  |  |  |
| --- | --- | --- |
| **구분** | **설명** | |
| **정보** | **URL** | **메뉴** |
| http://ip:port/#/search | 메인 페이지> 돋보기(검색) 버튼 |
| **취약점 상세** | 검색 함수에 xss 취약점이 존재하는 것을 확인했습니다. WAF나 그 외 방어 수단으로 방어했지만 에러를 유도해서 원하는 스크립트를 실행하도록 하는 스크립트에는 정상 반응함으로 xss 취약점 보완이 필요할 것으로 판단됩니다. | |

1) 검색 필드에 텍스트 입력

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

자동 생성된 설명

2) “<script>alert("XSS1")</script>” 입력

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

자동 생성된 설명

- 기본적인 xss 유도 스크립트는 안전하게 방어하는 것으로 확인

3) html 인코딩 된 xss 유도 스크립트 입력

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

자동 생성된 설명

- “%3Cscript%3Ealert(%22XSS1%22)%3C%2Fscript%3E”

- 스크립트로 인식되지 않고 문자열로 검색이 진행되는 것으로 확인

4) 에러를 유도한 변형 스크립트 입력

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

자동 생성된 설명

- “<img src=x onerror=alert('XSS1')>”

- WAF 와 같은 XSS 입력 방어 책을 우회하여 스크립트 실행 가능한 점 확인

- 이로 인해 공격자는 서버에 악의적인 코드 실행이 가능한 것을 확인

[보안 권고안]

1. 사용자의 입력 검증: HTML 태그 및 JavaScript 코드 제거
2. 정규 표현식을 활용해 안전한 패턴만 허용
3. 출력값 인코딩: HTML 컨텍스트 출력되는 값은 반드시 escape 처리
4. CSP, WAF 설정

## 3.3. 게시판 시스템

# 4. 보안 권고안

## 4.1. Login page

|  |  |
| --- | --- |
| **취약점 항목** | SQL Injection |
| **취약점 개요** | 로그인 시 사용되는 Email 입력 필드에 SQL 명령어 입력 취약점이 있음을 확인.  해당 취약점을 이용해 서비스의 데이터 베이스 구조, 사용자 계정, 서비스 데이터 등에 권한이 없는 사용자가 접근할 수 있게 되어 치명적인 2차 피해로 이어질 우려가 있음. |
| **보안 조치 방법** | |
| **<Prepared Statements 사용>**  SQL 쿼리문을 작성할 때 Prepared statements 또는 Parameterized Queries를 사용하여, 쿼리와 입력 데이터를 분리해야 합니다. 사용자가 입력한 값이 SQL 쿼리의 일부가 되지 않고, 데이터로서 처리되도록 할 수 있습니다.  **<사용자의 입력 검증 및 필터링>**  사용자의 입력을 받을 때 반드시 입력을 검증하고 허용된 형식인지 확인해야 합니다. Email 형식에서 사용가능한 특수문자 외 다른 문자들은 필터를 걸어 제외하는 것이 좋습니다.  **<최소 권한 원칙>**  서비스의 데이터베이스에 접근할 때 사용하는 계정의 최소한의 권한만 부여해야 합니다. 로그인 검증용 쿼리에는 SELECT 권한만 필요하며, 데이터를 수정할 필요가 없음으로 UPDATE, DELETE 권한을 제외해야 2차 피해를 최소화할 수 있습니다.  **<에러 메시지 노출 방지>**  에러 메시지에 구체적인 데이터베이스 정보를 노출하면 공격자의 SQL Injection 공격에 이용당할 수 있습니다. 에러 메시지는 노출하지 말고, 일반화된 메시지를 출력하거나 로그만 기록하는 것이 좋습니다. | |

## Search Field

|  |  |
| --- | --- |
| **취약점 항목** | XSS (크로스 사이트 스크립트) |
| **취약점 개요** | 사용자가 원하는 상품을 찾기 위해 사용하는 검색 필드에 XSS (크로스 사이트 스크립트) 취약점이 있음을 확인. 공격자는 해당 취약점을 이용해 서버에 악의 적인 코드를 실행하거나 2차 공격에 악용할 여지가 있습니다. |
| **보안 조치 방법** | |
| **<사용자의 입력 검증 및 필터링>**    사용자의 입력을 받을 때 반드시 입력을 검증하고 허용된 형식인지 확인해야 합니다.  function sanitizeInput(input) {  return input.replace(/<|>|"|'/g, '');  }  입력 값에 포함된 특수문자를 제거하는 함수를 사용하거나 예상되는 입력 형식과 일치하지 않는 경우 입력을 거부하는 방법이 좋을 것입니다.  **<출력값 인코딩>**   |  |  | | --- | --- | |  | 사용자 입력값을 HTML 페이지에 출력하기 전 HTML 엔티티로 변환하여 브라우저가 스크립트로 인식하지 못하게 만듭니다.  const escapeHtml = (str) => {  return str.replace(/&/g, '&amp;')  .replace(/</g, '&lt;')  .replace(/>/g, '&gt;')  .replace(/"/g, '&quot;')  .replace(/'/g, '&#039;');  };  위 코드와 같이 HTML 컨텍스트에서 <와 >를(그 외 script를 구성하는 특수문자들 포함) &lt; &gt; 로 변환하거나, JavaScript 컨텍스트에서 문자열을 JSON.stringigy() 함수로 변환하거나 URL 컨텍스트에서는 encodeURIComponent()를 사용하는 것이 좋습니다. |   **<HTTPS 사용>**  공격자가 악성 스크립트를 삽입하지 못하도록 보안이 강화된 http 프로토콜을 사용하는 것이 좋습니다. 추가로 SSL/TLS 인증서를 설정하여 HTTPS로만 서버에 연결되도록 강제하는 것이 좋습니다.  **<WAF & CSP>**  WAF는 사용자의 악의적인 입력을 감지, XSS를 탐지하여 차단하는 역할을 수행합니다. CSP는 신뢰할 수 있는 도메인에서 제공되는 스크립트만 실행되도록 시스템을 제한하는 서비스입니다. 두 서비스를 적용하여 공격자의 악의적인 스크립트 입력을 미연에 차단하고 방어할 수 있습니다.  **<사용자 세션 보호>**  XSS 공격으로 인해 서비스의 사용자(관리자 계정 포함) 세션 정보를 탈취당할 가능성이 있습니다. 이를 사전에 HttpOnly, Secure 플래그를 사용해 보호하는 것이 좋습니다. | |