===============================

“웹 보안 및 해킹” 실습 보고서

===============================

Burp Suite를 활용한 패킷 분석 및

DVWA를 이용한 웹 취약점 실습

AI를 활용한 보안전문가 양성 과정

최형회

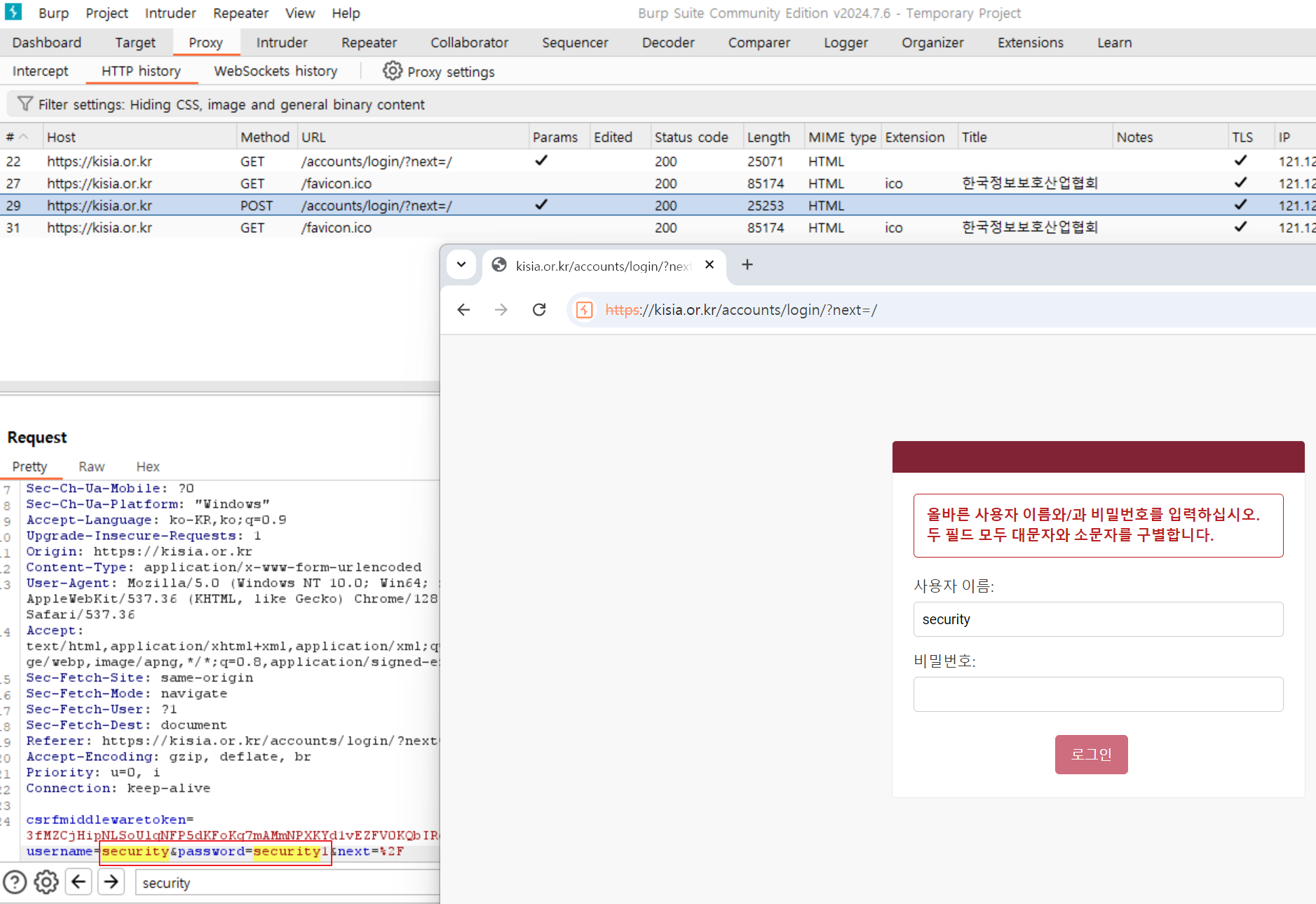
2025년 7월 25일

장대현 강사님

===============================

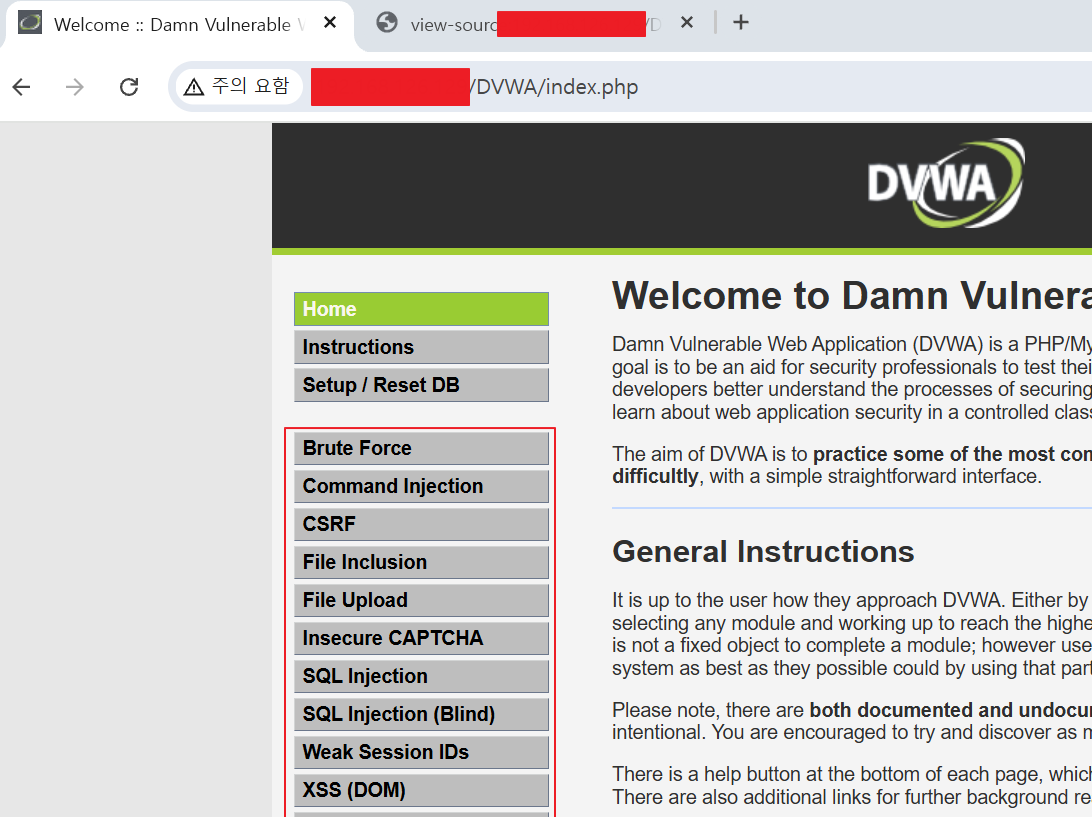
서론

본 실습에 앞서 Burp Suite 프로그램을 사용하여 프록시에서 클라이언트와 서버 간 HTTP 패킷을 가로채고 분석 및 수정해보는 실습을 진행하였습니다.



“BurpSuite”는 웹 프록시 도구로 패킷을 캡처하거나 정지시켜놓고 해당 패킷에 대한 분석 혹은 조작하여 테스트 하는 데에 사용합니다. Proxy, Repeater, Intruder 등의 기능을 사용하여 패킷을 분석해보고 패킷을 조작하여 보내보기도 하고 반복적으로 무작위 대입 공격을 실습해보았습니다. 그리고 여기서 프록시는 클라이언트와 서버 간의 통신 사이에 통신을 중개하는 서버나 프로그램 등을 의미합니다. 이러한 과정을 통해 Burp Suite 프로그램 활용법과 프록시의 역할에 대해 익힐 수 있었습니다.

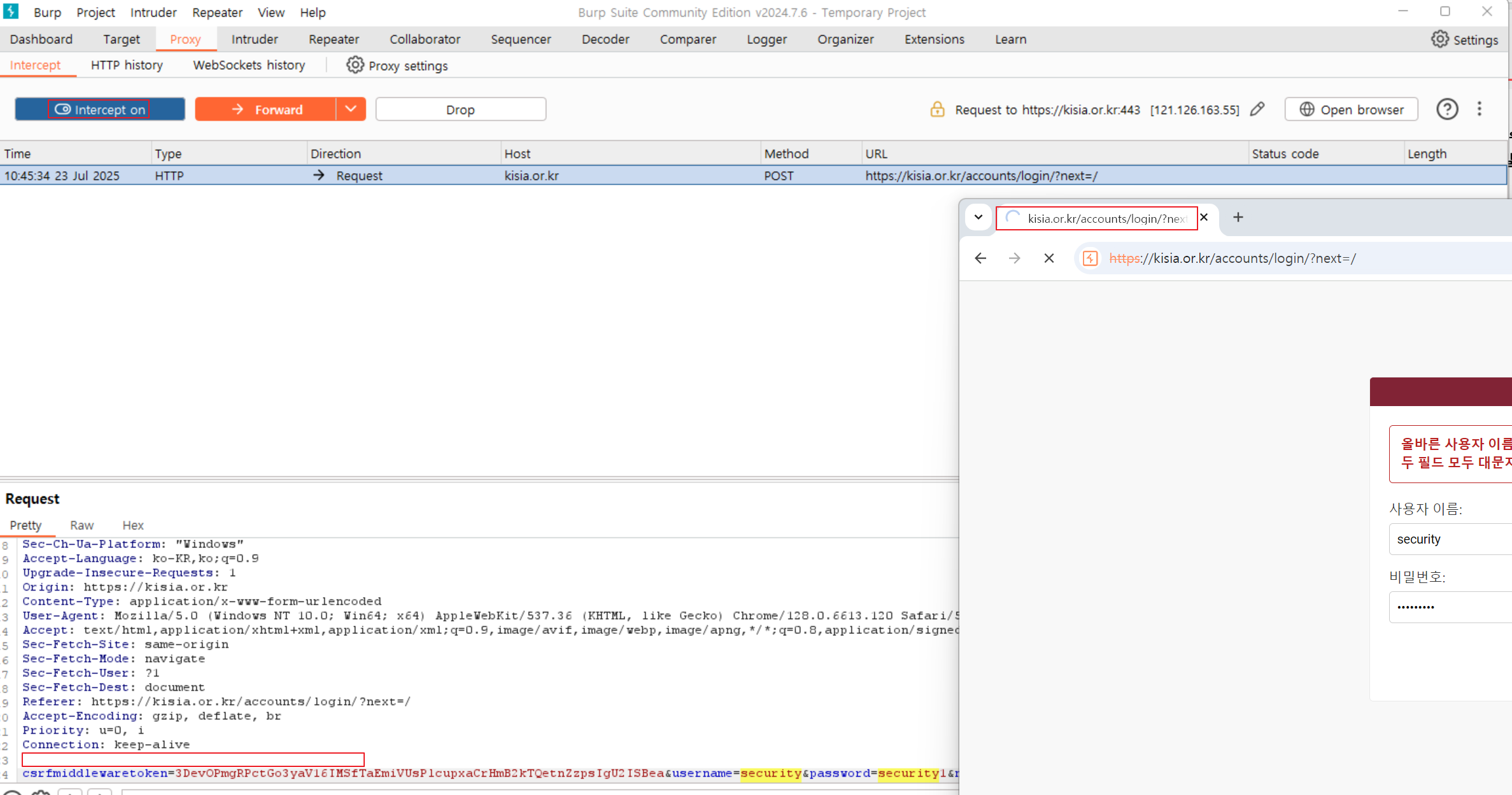
그리고 본 실습에서는 가상머신에 kali 이미지를 올려 “DVWA” 서비스 환경을 구성하고 해당 사이트의 웹 보안 취약점 문제들을 풀어보았습니다. “DVWA”는 모의해킹 테스트를 위해 제공된 환경으로 많은 강의 혹은 교육 등 공부에 사용됩니다. 본론에서 이 “DVWA” 중 몇 가지 문제풀이의 과정 및 분석 결과에 대해 다뤄보고자 합니다.



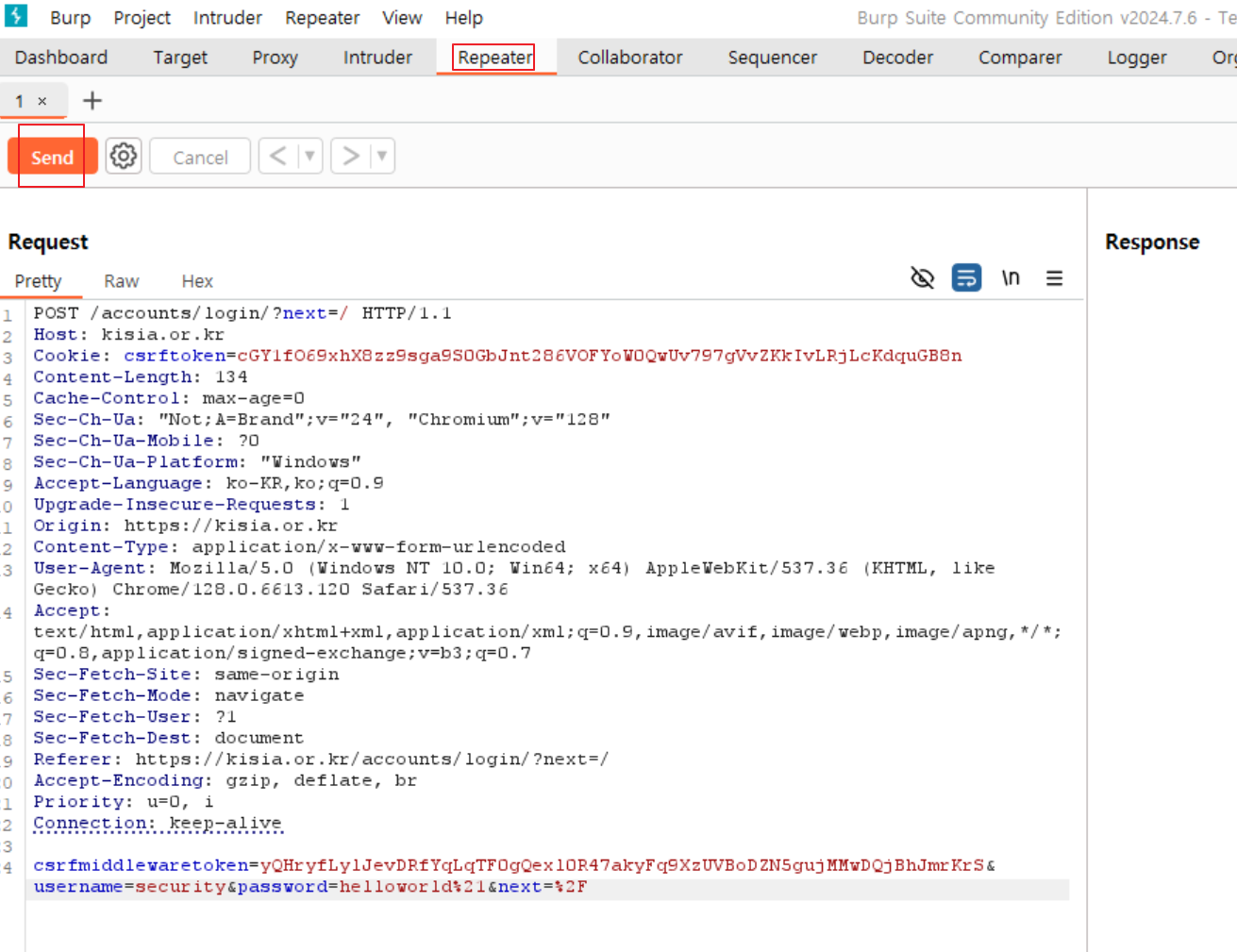
본론

1. “BurpSuite”의 기능

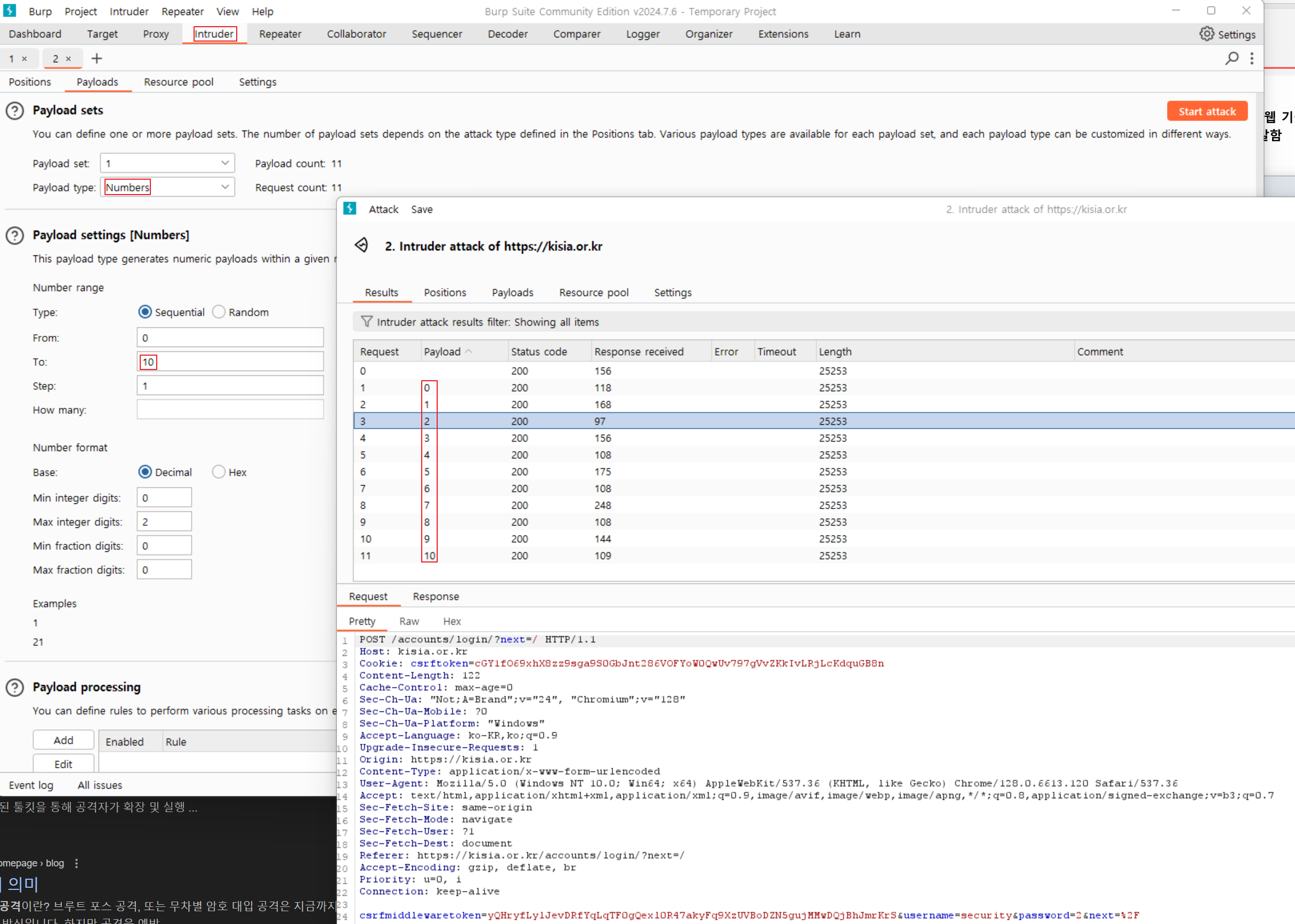
실습에서 활용된 기능 중 첫 번째로 Proxy 는 패킷을 수집 및 관찰하는 기능입니다. “Intercept”를 사용하여 패킷을 막아놓고 패킷 내용을 확인하거나 단계별 전송에 따른 응답 패킷을 대응되는 요청 패킷과 함께 확인할 수 있습니다. 그리고 “Intercept” 기능을 사용하지 않아도 “HTTP history” 탭을 이용하여 오가는 패킷을 스니핑만 하여 내용을 확인할 수 있습니다.



두 번째 기능은 Repeater입니다. Proxy 탭에서 얻은 패킷 정보를 이 Repeater 탭으로 넘겨 패킷 내부 값을 변경해 볼 수 있습니다. 그리고 내부 값이 변경된 요청 패킷을 전송하여 그 결과로 전송되어 오는 응답 패킷과 비교하여 확인할 수 있습니다. 이 기능을 “DVWA” 서비스에 여러 가지 방식의 공격을 진행해보고 어떤 방식이 유효한지 테스트 해보는 용도로 사용했습니다.



세 번째로는 “Intruder” 기능입니다. 실습 과정에서 많이 사용하진 않은 기능이지만 아래 사진과 같이 Payload를 작성하여 반복적으로 패킷을 전송해보는 기능입니다. 아래 사진에서는 값을 변경할 파라미터를 지정하고 해당 파라미터에 0부터 10까지 1씩 증가하며 총 11번 패킷을 전송해보는 식으로 기능을 확인하였습니다. 실습이라서 아주 적은 수의 케이스만 테스트 하였지만, 이를 사용해 업무를 진행하거나 공격을 수행할 때(?)에는 자세한 Payload를 작성하여 크게보면 브루트 포스 공격을 수행하는 도구로 활용할 수 있는 것처럼 보입니다.

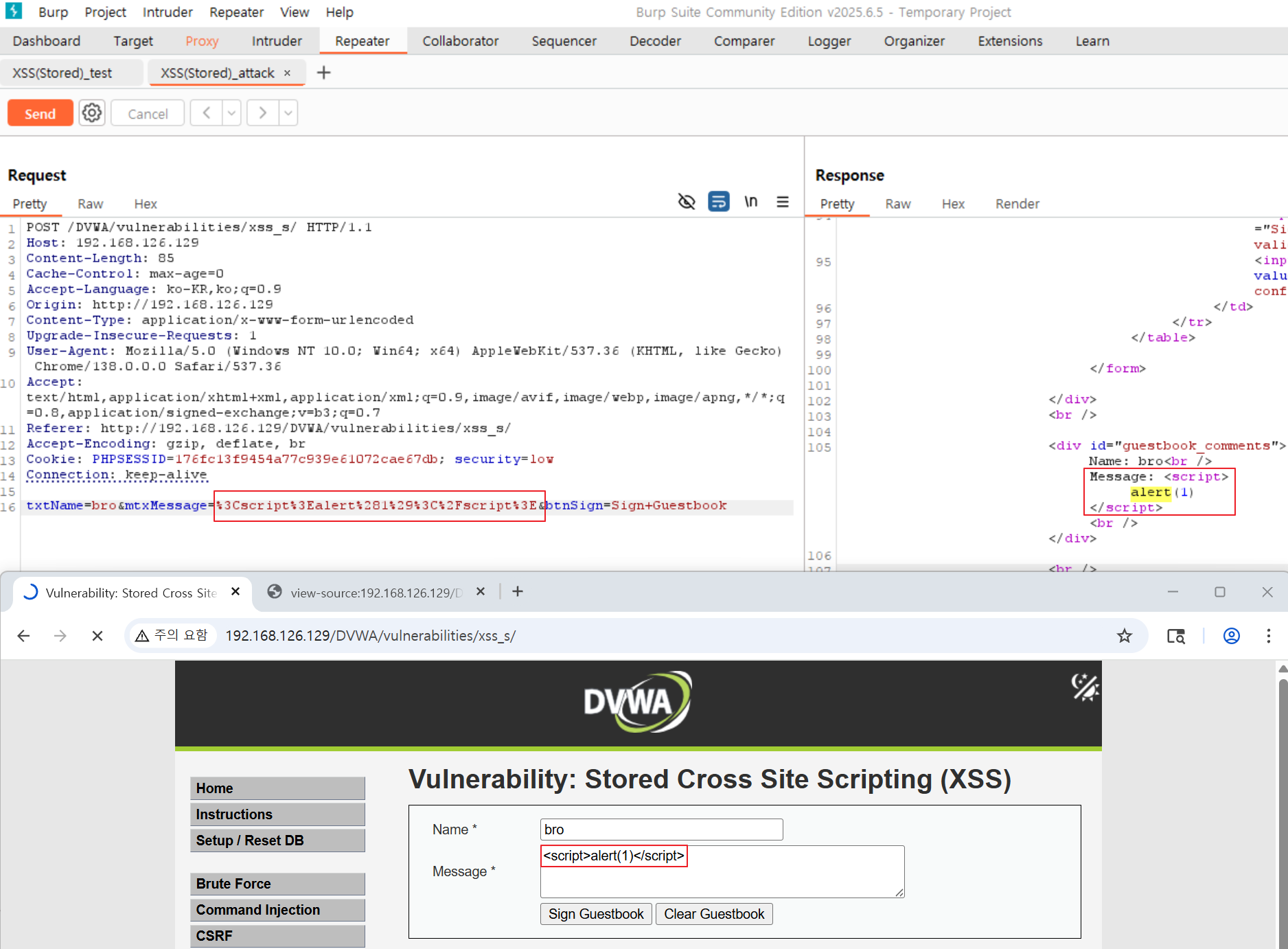


설명해주신 기능은 위의 3가지이지만 실제로는 “Froxy”, “Repeater”를 이용하여 패킷을 탐지 및 조작 해보고, 그 외 가상머신에서 DB를 확인하거나 DBWA에서 제공하는 source code 등을 확인하여 실습을 진행하였습니다.

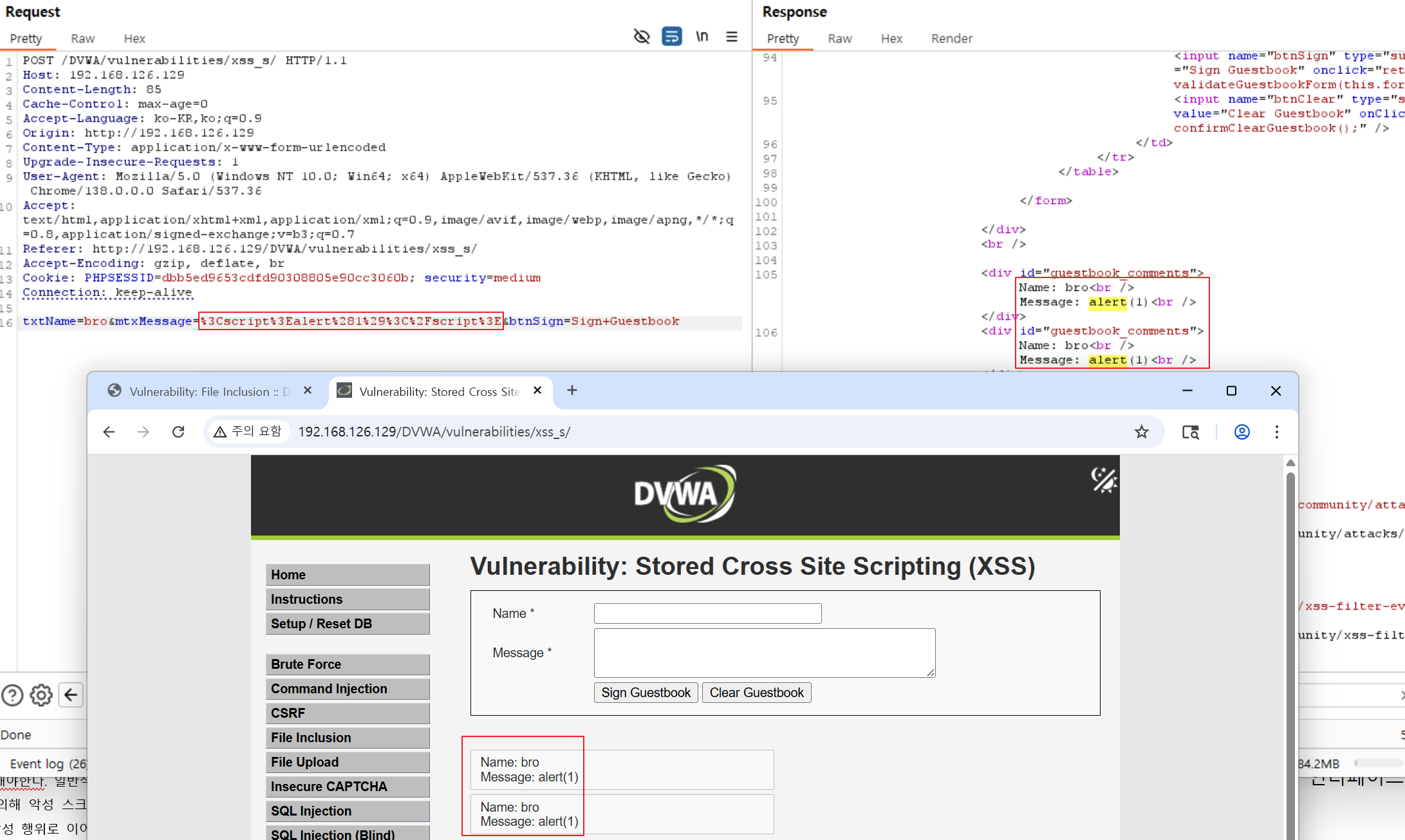
2. Stored XSS

XSS는 Cross Site Scripting의 약자로 입/출력 값 검증 부재로 공격자가 게시글 또는 URL을 통해 삽입한 악성 스크립트가 사용자의 요청에 의해 사용자 측(브라우저)에서 실행되는 취약점입니다. 예로 들어 공격자가 웹 서버 게시판에 악성 스크립트를 삽입한 채로 게시글을 작성하여 두었다고 가정해보겠습니다. 해당 게시글을 클릭한 타 사용자의 브라우저에서 실행된 악성 스크립트로 인해 중요정보가 공격자에게 전달됩니다.

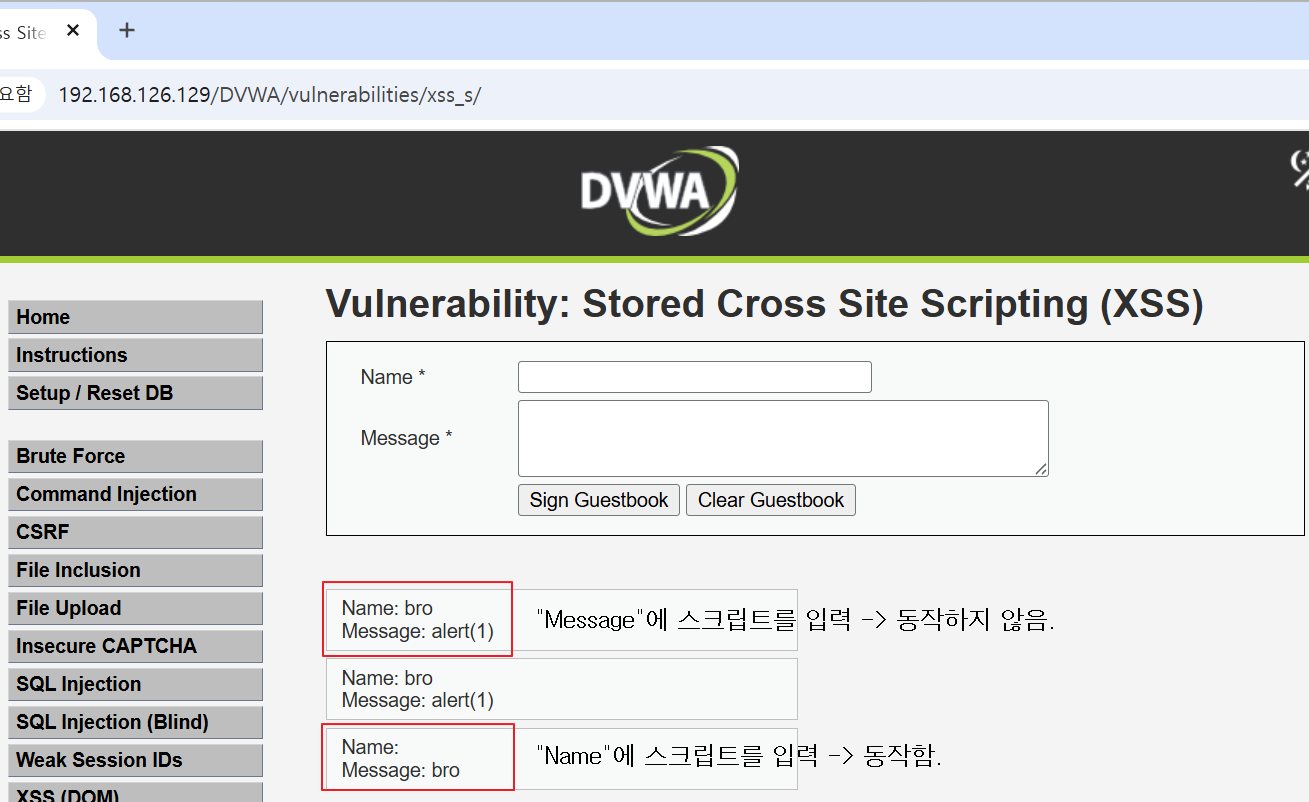
XSS는 악성 스크립트가 전달되는 위치에 따라 3가지 정도로 분류됩니다. 첫 번째로 “Stored XSS”는 위에서 설명한 내용에 해당합니다. 웹 서버의 DB에 악성 스크립트를 삽입하고, 해당 웹 서비스를 이용하는 타 사용자에게 악성 스크립트가 실행되도록 합니다. 아래 캡처와 같이 게시글 “Message” 입력칸에 악성 스크립트를 입력하여 서버 측에 요청 패킷이 전달되면 해당 파라미터의 내용이 응답 패킷에 그대로 노출되어 악성 스크립트가 실행되는 방식입니다.



위의 방식은 DVWA의 low 난이도 문제이고, XSS 취약점에 대한 필터링이 전혀 되어 있지 않아 원하는 어떤 악성 스크립트라도 그대로 실행되도록 만들 수 있습니다. 그에 반해 medium 난이도 XSS에 대한 대응이 되어 있는 것 같습니다. 아래 캡처와 같이 위와 똑같은 스크립트를 게시하고자 하였을 때 입력한 값과 다른 결과 값이 게시글에 저장되어 있는 것을 확인하였습니다. 이는 서버 측에서 패킷에 대해 처리하는 구문이 추가된 것으로 보입니다.



그래서 파라미터를 바꿔 “Name” 필드에 스크립트를 입력하였더니 아래 그림과 같이 “</script>” 구문은 사라지지 않은 것을 확인하였습니다. 그래서 “Name” 필드를 공략해 스크립트를 실행할 수 있지 않을까 생각하였습니다. ”Name” 필드에 maxlength가 10으로 설정되어 있어 크롬 개발자 도구를 활용하여 제한을 늘려주고, 여러가지 방식으로 필터를 우회하고자 시도해보았습니다. 띄어쓰기를 입력하거나 다른 문자를 섞어 쓰는 등 필터를 피해보고자 하였지만 이 경우에 스크립트 문법에 맞지 않아 스크립트가 동작하지 않았습니다. 그래서 강사님이 알려주신 대문자를 섞어 쓰는 방법을 적용해보았더니 필터를 우회하여 스크립트가 동작하는 것을 확인하였습니다. 왜 이런 방식이 가능한지 생성형 AI를 통하여 확인해보니 HTML 파싱의 특성이라고 합니다. 브라우저의 HTML 파서는 태그명을 대소문자 구분없이 처리한다고 합니다. 그래서 “<script>”,”<Script>”,”<SCRIPT>” 등을 모두 동일하게 인식한다고 합니다. 간단한 문자열 매칭 기반 필터들은 정확히 “<script>”만 필터링 하기 때문에 대소문자가 섞인 변형은 놓칠 수 있다고 합니다. 그래서 테스트를 위하여 “Name” 필드가 아닌 “Message” 필드에 해당 스크립트를 입력하였을 때에는 스크립트가 동작하지 않는 것을 확인하였습니다. 각각의 입력에 대하여 다른 필터가 적용되어 있고, 이 중 “Name”에는 대문자로 변형된 스크립트 태그명에 대한 필터링이 적용되어 있지 않음을 확인할 수 있었습니다.



문제를 해결한 뒤 “DVWA”에서 제공하는 소스코드를 확인하였습니다.

|  |
| --- |
| <?php  if( isset( $\_POST[ 'btnSign' ] ) ) {  // Get input  $message = trim( $\_POST[ 'mtxMessage' ] );  $name = trim( $\_POST[ 'txtName' ] );  // Sanitize message input  $message = strip\_tags( addslashes( $message ) );  $message = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], $message ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));  $message = htmlspecialchars( $message );  // Sanitize name input  $name = str\_replace( '<script>', '', $name );  $name = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"], $name ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));  // Update database  $query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";  $result = mysqli\_query($GLOBALS["\_\_\_mysqli\_ston"], $query ) or die( '<pre>' . ((is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_error($GLOBALS["\_\_\_mysqli\_ston"]) : (($\_\_\_mysqli\_res = mysqli\_connect\_error()) ? $\_\_\_mysqli\_res : false)) . '</pre>' );  //mysql\_close();  }  ?> |

코드를 확인하여 본 결과 “Message” 입력에 대해서는

* addslashes() : 특수문자 앞에 백슬래시를 추가하여 SQL injection 공격 방지
* strip\_tags() : HTML, PHP 등의 태그를 제거하여 XSS 공격 방지
* mysql\_real\_escape\_string() : 특정 특수문자 앞에 백슬래시를 추가하여 SQL injection 방지
* htmlspecialchars() : 입력받은 값을 HTML Entity Code로 변환하여 Text Data로 처리

등의 기능을 이용하여 입력 값에 대한 필터링이 되어 있습니다. 반면에 “Name” 입력에 대해서는 “str\_replace()”를 통해 단순히 “<script>” 태그만 제거하고 있음을 확인할 수 있습니다.

위 두 가지 난이도에 대한 실습을 진행한 뒤 추가적으로 “Impossible” 난이도에도 도전해 봤습니다. 위의 표의 코드에서 “strip\_tags(addslashes())” 구문이 “stripslashes()”로 대체되었고, “Name” 과 “Message”에 같은 필터가 적용되었습니다. 이에 더해 Prepared Statement를 사용하여 SQL injection에 대한 방지가 추가되어 있습니다. 각각의 메소드 들의 기능을 살펴본 결과 물론 각각의 메소드에도 취약점으로 지적되는 부분은 있습니다. Prepared Statement의 경우 동적 SQL 문이 사용되면 사용자 입력이 쿼리 구조 자체를 변경할 수 있어 보호 효과가 없어지게 됩니다. 그리고 stripslashes()의 경우 PHP 5.4 이하의 “magic\_quotes\_ gpc"가 활성화 되어 있는 환경에서 꼭 포함되어야 하는 구문이지만 지금은 거의 사용되지 않는 구문이며, 현재 대상 시스템은 php 8.4.6 버전으로 동작됩니다. 하지만 HTML Entity code 디코딩에 도전하는 등 여러 가지 방식으로 시도해봤지만 문제를 풀어내지는 못했습니다.

이 실습으로 알 수 있는 내용은 공격자 측에서 파라미터를 선택하는 것도 중요하며, 서버 측에서는 각각의 필드에 대해 개별적으로 필터를 적용하는 것이 아니라 모든 입력 파라미터에 대해 충분한 필터를 적용해야 한다는 것입니다. 시간 남으면 그 외 검색한 XSS 방지 관련 자료 추가하기

3. File Upload

다음으로는 2일차에 진행된 File Upload 웹 취약점에 대한 실습입니다. File Upload 취약점은 파일 업로드 기능이 존재하는 웹 서비스에서 업로드 되는 파일에 대한 검증이 존재하지 않는 경우 악성 파일을 업로드 하는 것이 가능해지는 문제입니다. 보통 이 공격의 경우 “웹셀”(Web Shell) 파일이 업로드 되며 웹셀은 시스템에 명령을 내릴 수 있는 코드 혹은 그 코드를 포함한 파일을 의미합니다. DVWA에서 low 난이도를 확인해보니 아래 첫번째 캡처와 같이 Upload 버튼을 누르면 POST 방식으로 서버에 업로드 됩니다. 여기에서 발견된 관련 취약점은 총 3가지입니다.

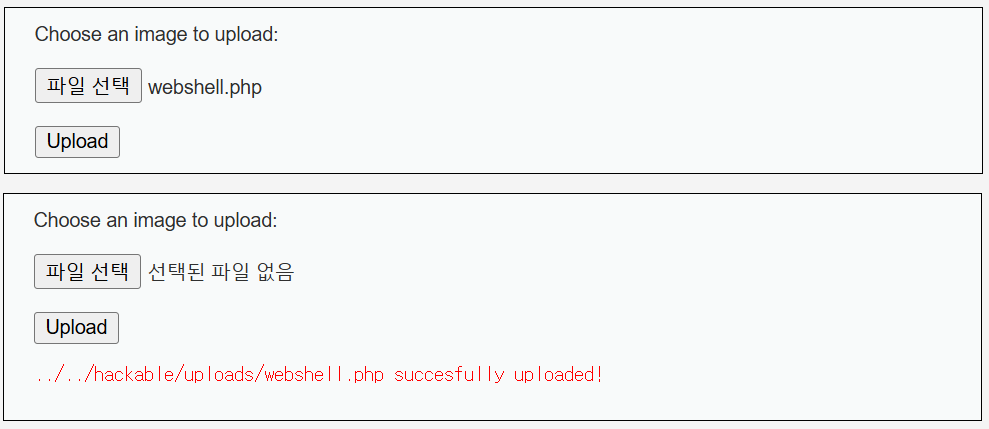
* 첫 번째 캡처를 보면 .php 파일이 업로드 됩니다. 실습용으로 만든 webshell.php 파일이 그대로 업로드 되었습니다.

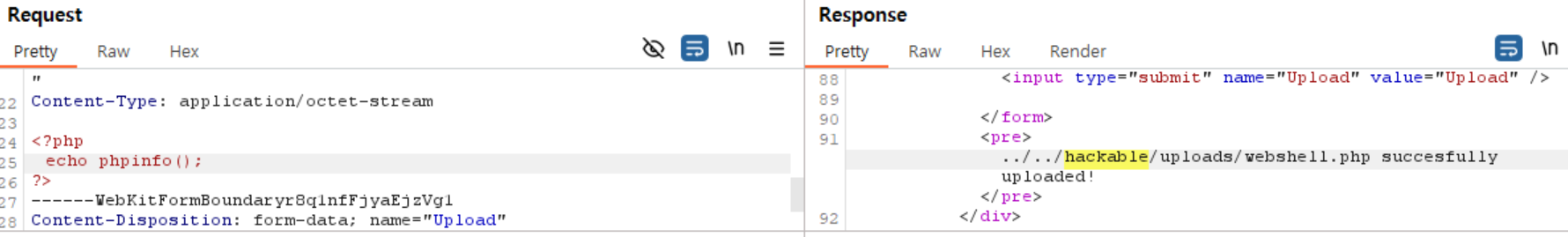
이는 업로드 된 파일에 대한 검증 과정이 없음을 의미합니다.

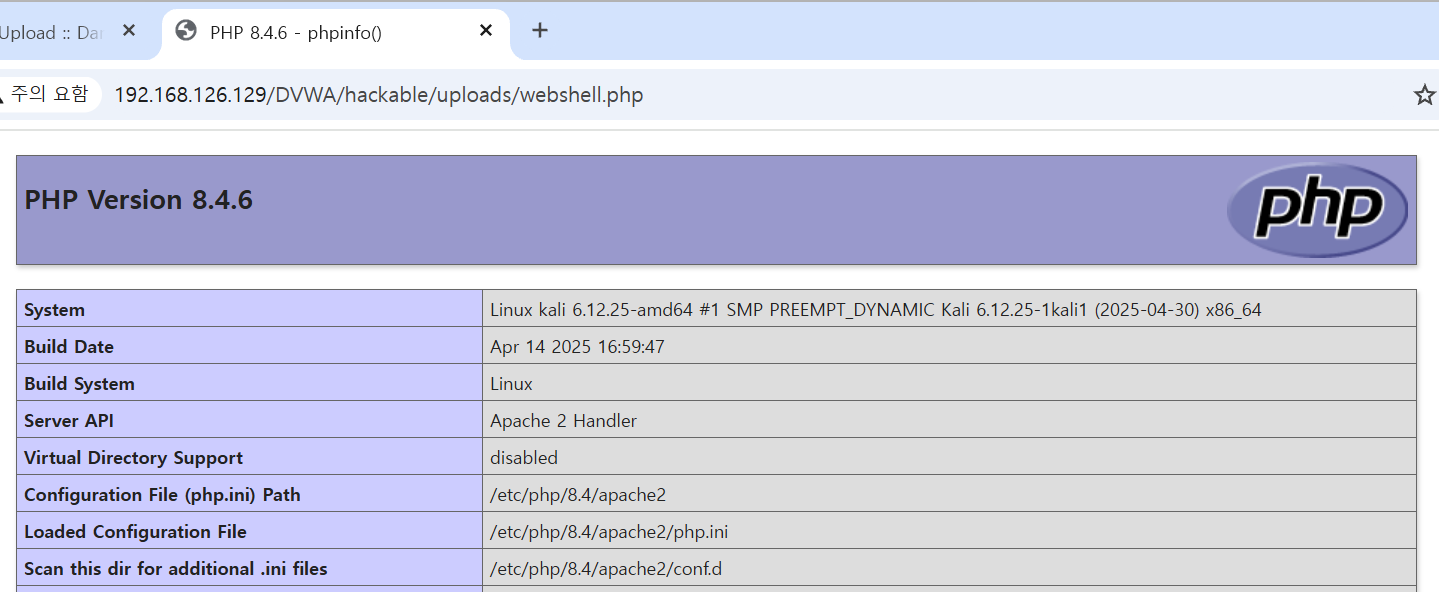
* 업로드 되는 파일이 저장되는 디렉터리 경로가 노출되어 있습니다. 업로드 성공을 알리는 메시지에 해당 파일의 상대경로가 포함되어 있습니다.

첫 번째 캡처 및 두 번째 캡처에서 확인할 수 있습니다.

* 세번째 캡처에서 확인할 수 있듯이 웹셀 파일의 경로를 URL에 담아 해당 파일을 실행시킬 수 있습니다. PHP 코드로 작성한 phpinfo() 메소드가 그대로 동작하는 것을 확인할 수 있습니다.





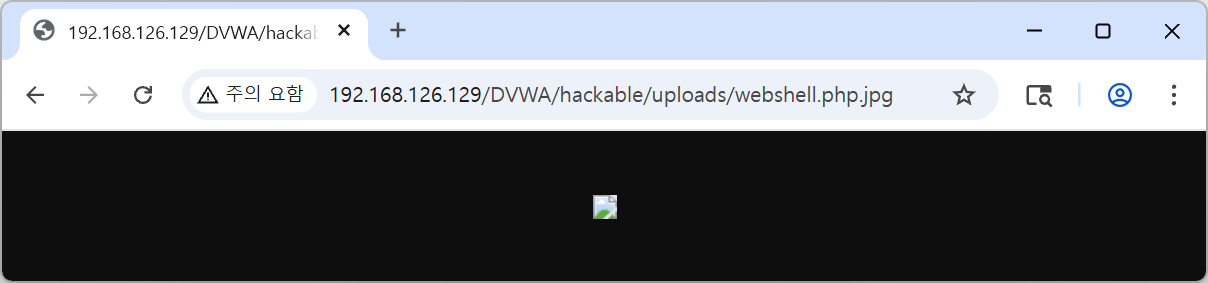


이 low 난이도 문제에는 해당 취약점에 대한 방어 기제가 전혀 없는 것 같습니다. 실제로 코드를 확인해보니 파라미터로 입력 받은 값을 그대로 서버에 업로드 하는 것을 확인할 수 있었습니다.

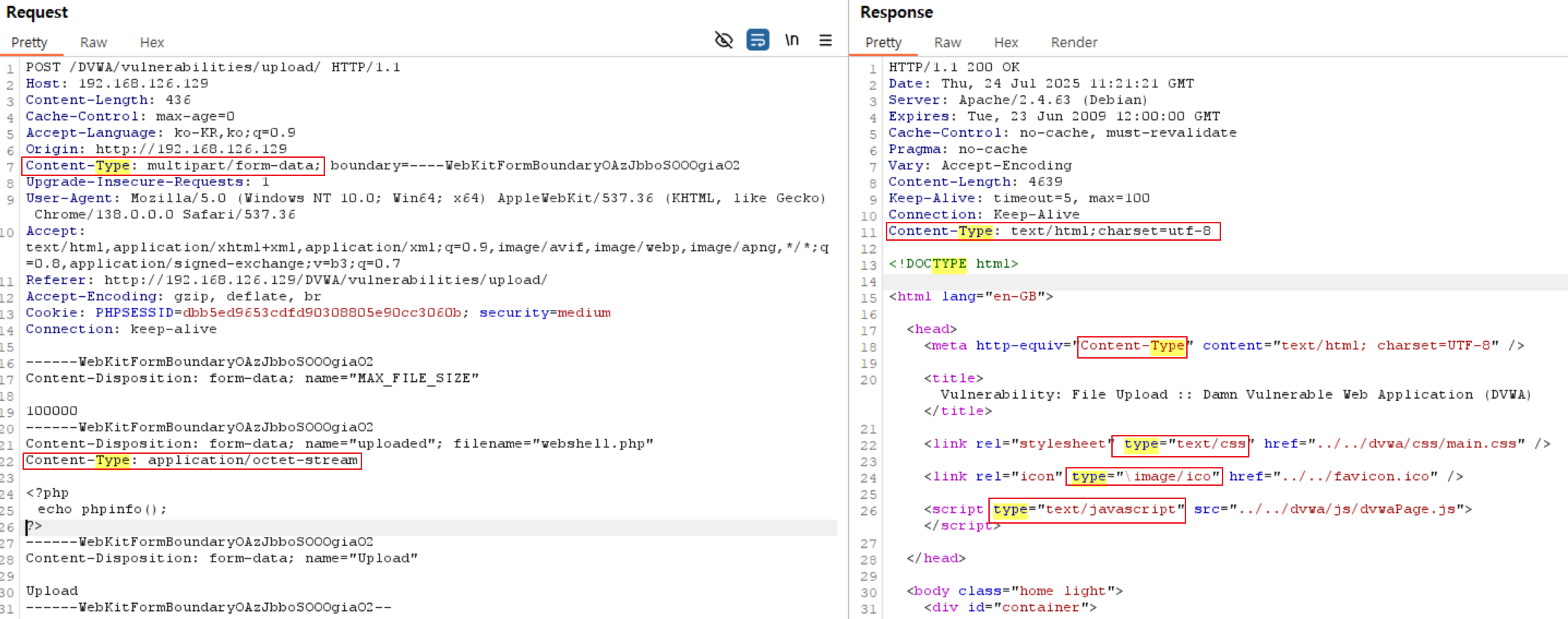
다음으로 medium 난이도 문제에서 동일한 웹셀 파일을 업로드하고자 하니 에러 메시지가 출력되며 업로드 되지 않은 것을 확인할 수 있었습니다. 에러 메시지 내용을 확인해보니 JPEG 혹은 PNG 등 이미지 파일에 한해 업로드 된다고 나와있습니다. 그러면 특정 방법을 통해 파일의 형식을 검증하여 필터링 하고 있음을 알 수 있었습니다.



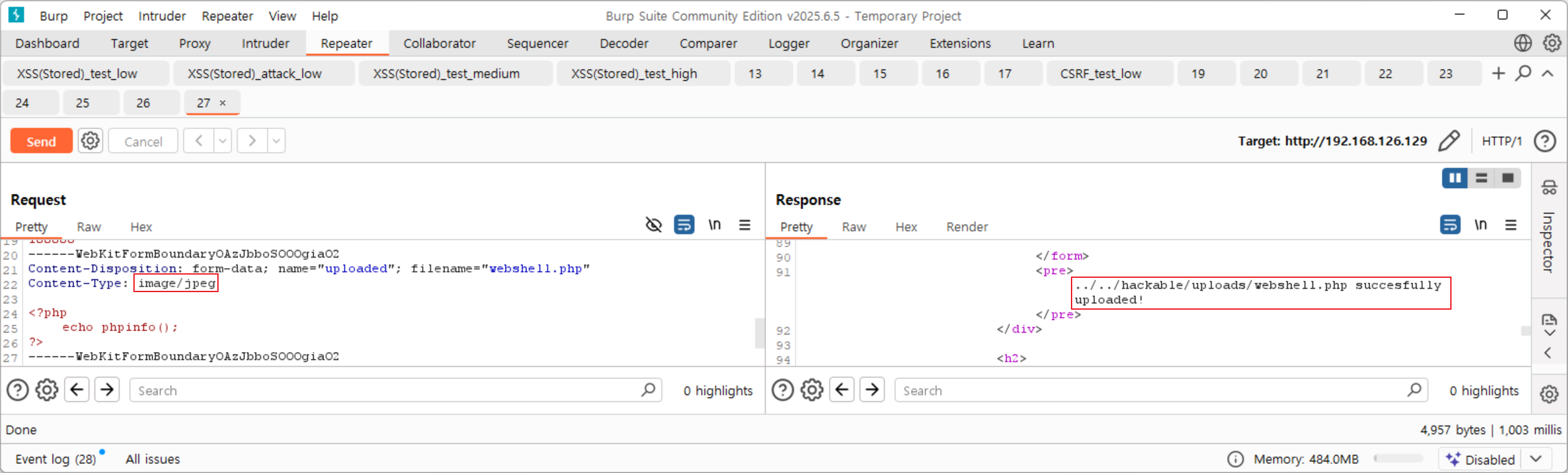
이 검증을 우회하기 위해 파일의 확장자를 바꾸어서 업로드 해봤습니다. 파일 업로드는 성공적으로 되었지만 같은 방식으로 URL을 통해 접근해 보았으나 파일이 실행되지 않았고, 해당 파일의 확장자를 변경하거나 다른 방법으로 실행시킬 방법을 찾지 못 했습니다.



그래서 서버 측 혹은 클라이언트 측에서 파일의 형식을 검증하는 부분을 찾기 위해 아래 캡처와 같이 패킷을 캡처하여 확인해보았습니다.



해당 패킷에 php라고 검색해봤지만 원하는 정보가 나오지 않았고, type이라고 검색하니 HTTP 패킷에서는 “content-type” 혹은 “type=” 라는 구문으로 파일의 형식을 지정하고 있음을 알 수 있었습니다. 그래서 요청 패킷의 Content-Type 구문 2가지 중 헤더에 포함된 것은 패킷의 형식 혹은 포함된 바디의 형식을 표현하는 것으로 보입니다. 그래서 아래 바디의 Content-Type에 대해 검색하여 “image/jpeg”라는 입력 값을 넣어야 함을 확인하였습니다. Burp Suite의 Repeater 기능을 이용하여 아래 캡처와 같이 요청 패킷을 전송한 결과 성공적으로 업로드 되는 것을 확인하였습니다. 그리고 URL을 통해 접근하여 확인한 결과 동일한 위치에 파일이 업로드 됨을 확인하였습니다.(Low 난이도 문제와 구분하기 위해 DVWA 서비스를 구동중인 kali 서버에 접근하여 파일을 삭제한 뒤에 진행하였습니다.)



이외 High 난이도 및 Impossible 난이도 문제에 대해서는 DVWA 에서 제공하는 소스코드를 통해 내용만 확인하였습니다. High 난이도는 기존 medium 난이도에서 Content-Type을 통해 파일의 형식을 확인하던 것에서 substr() 메소드를 사용하여 파일명에서 확장자를 분리해 파일의 형식을 확인하는 것을 보입니다. 이 경우 위의 medium 난이도에서 확인했듯이 파일의 확장자를 변경해서 업로드하면 해당 파일을 실행하기위해 추가적인 작업이 필요합니다. 그래서 File Upload 취약점의 방어법으로 제 역할을 할 것으로 보입니다. 그리고 파일의 디렉터리 경로는 여전히 결과 출력문에 노출되어 파일에 대한 접근시도가 가능해 보입니다. 그리고 URL을 통한 파일을 실행하는 것은 난이도에 상관없이 가능한 것으로 보입니다.

Impossible 난이도는 이에 더해 CSRF 토큰 검증으로 XSS 취약점에 대한 반복적인 테스트 난이도를 어렵게 하였으며, 업로드된 파일명을 처리과정에서 MD5 암호화된 채로 진행하기 때문에 파일명을 알 수 없게 되어있습니다. 그리고 getimagesize() 메소드를 통해 실제로 이미지 파일인지 확인하는 과정을 거치며, 이미지 재인코딩을 통해 메타데이터 제거 후 이미지 데이터만 처리합니다. 하지만 여전히 파일 경로가 노출됩니다. Alink를 사용하여 화면에는 출력되지 않지만 공격자가 단순히 해당 링크를 클릭하는 것으로 정확한 경로를 확인할 수 있습니다. 악성 웹셀 파일을 업로드하는 데에는 보안적인 요소가 추가되었지만 파일의 업로드에만 성공한다면 쉽게 파일이 저장된 디렉터리에 접근하여 해당 파일을 실행시킬 수 있습니다.

실습 대상 DVWA 서비스의 File Upload 취약점 대응방안으로는 아래와 같은 내용들이 제안됩니다.

* 이미지(jpg, gif, bmp등), 문서(hwp, docx, xls 등)등의 파일을 제외한 모든 확장자 제한
* 경로가 노출되지 않도록 성공 출력 메시지에서 제외
* 서버의 권한 설정을 통해 공격자의 파일 및 디렉터리에 대한 접근 및 실행 권한을 제한

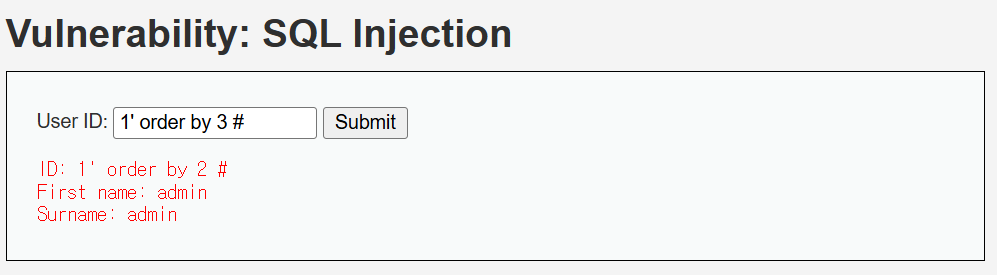
4. SQL injection

SQL이란 DB에서 데이터를 관리, 조작, 추출 등을 하기 위한 DB 표준 프로그래밍 언어 입니다. DBMS 시스템에 따라 NULL 값 처리, 데이터 조회 방식, 함수 등에서 조금씩 차이점이 있지만 DML, DDL, DCL 등 기본적인 언어는 전부 동일하게 사용됩니다. 그 중 DML은 데이터 관리, DDL은 테이블 관리, DCL은 권한 관리를 위해 사용됩니다.

SQL injection이란 입력 값 검증 부재로 악성 SQL 쿼리문을 서버로 전달하여 DB가 비정상적인 동작을 하도록 조작하는 공격 방식입니다. 공격에 따라 서버 내 DB의 데이터가 조작 혹은 추출되거나, 혹은 테이블이나 스키마를 생성/수정/삭제할 수도 있습니다. OWASP에서는 일반적으로 이 공격이 높은 심각도를 가지고 있다고 판단하며 주기적으로 발표하는 웹 보안 취약점에 항상 포함시키고 있습니다.

SQL injection은 크게 Error 기반 공격과 Blind 기반 공격 방식으로 나눌 수 있습니다. 에러 메시지에 에러 원인에 대한 내용이 포함되어 있다면 이를 통해 삽입문을 변경해가며 원하는 방식으로 공격을 진행해나갑니다. 이를 막기 위해 에러 메시지에서 에러의 유형을 구분할 수 없도록 특정 에러 메시지를 준비하여 해당 메시지만 출력되도록 할 수 있습니다. Blind 기반 공격은 앞서 말한 것과 같이 에러 메시지에 사용할 수 있는 정보가 없거나 에러 메시지가 아예 출력되지 않을 때 사용되는 방식입니다. 참/거짓을 의미하는 데이터 타입 Boolean 정보를 기반으로 참일 때 결과와 거짓일 때의 결과를 가지고 판단하여 공격을 수행하는 방식이 있습니다. 그리고 이 Boolean 정보도 노출되지 않을 때, 입력에 따른 응답에 걸리는 시간으로 판단하거나 응답을 지연시키는 Sleep 함수등을 사용하여 공격하는 방식이 있습니다. 위 두가지 방식 외에 UNION 구문을 활용하여 공격자가 원하는 데이터를 조회하는 방식이나 다량의 SQL 코드를 삽입하여 DB를 조작하는 방식 등의 공격이 있습니다.

관련하여 UNION을 사용하여 DB 관련 정보를 획득하는 실습을 진행하였습니다. 난이도는 low 로 설정하고 진행하였습니다. 첫 번째로 “’”(작은따옴표)를 활용하여 입력한 값이 쿼리문에 그대로 삽입됨을 확인하였습니다. 아래 캡처와 같이 Order by 구문을 이용하여 칼럼 수를 확인하였습니다. 3일 때 에러가 발생해 칼럼 수가 2임을 확인하였습니다. 이에 더해 User ID가 1인 계정은 admin이라는 것을 확인할 수 있었습니다. 그리고 User ID가 단순한 숫자임을 알았기 때문에 다른 숫자를 넣어볼 수도 있지만 현재 실습과는 다른 내용이기 때문에 무시하고 넘어가겠습니다.



다음으로는 DB, table, column에 대한 정보를 획득하는 과정입니다. User ID 정보를 확인하기 위해 DB 내의 어떤 테이블에 유저 정보가 담겨있는지 확인한 후 해당 테이블의 데이터를 조회하여 원하는 정보를 얻어내기 위함입니다. 아래와 같은 2개의 구문으로 “dvwa” 스키마의 “users” 테이블에 유저 정보가 담겨있음을 확인하였고, 마지막 구문으로 “user” 및 “user\_id”에 대한 정보를 확인하였습니다.

|  |
| --- |
| ' union select 1,schema\_name from information\_schema.schemata #  ' union select 1,table\_name from information\_schema.tables where table\_schema = "dvwa" #  ' union select 1,column\_name from information\_schema.columns where table\_schema = "dvwa" and table\_name = "users" #  ' union select user, user\_id from dvwa.users # |



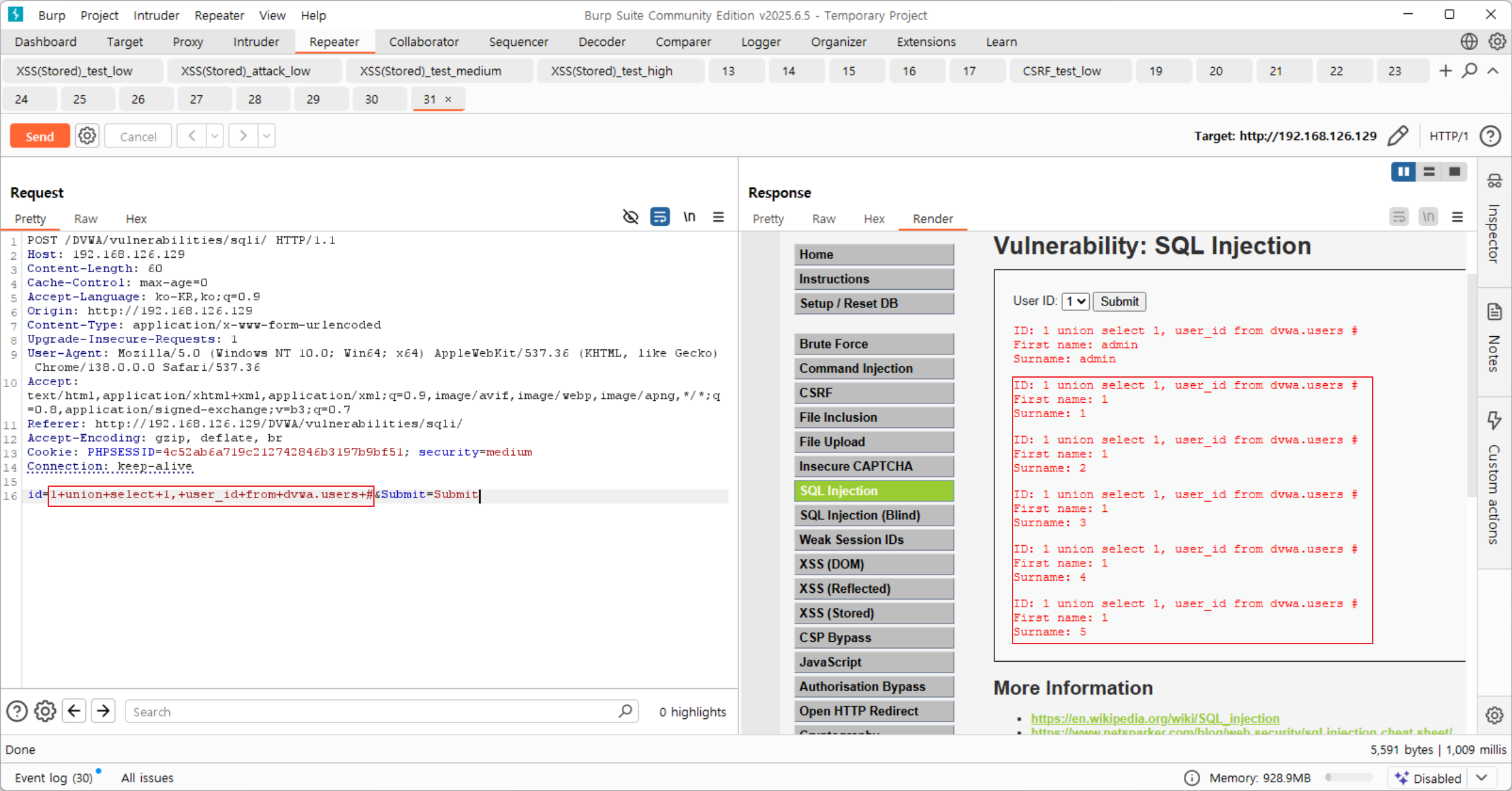
위와 같이 low 난이도 문제는 입력 값에 대한 검증 혹은 필터링이 전무하여, 원하는대로 DB에 접근 및 데이터 조회가 가능하였습니다. Medium 난이도 문제는 입력 칸이 텍스트 입력이 아닌 드롭다운에서 선택하는 식으로 변경되어 있습니다. 원하는 구문을 입력하기 위해 아래 캡처와 같이 Burp Suite에서 패킷을 캡처하여 보았습니다.



POST 방식으로 파라미터가 body에서 전달되고 있으며, 쿠키에 담겨있는 세션 토큰 외에 유저 토큰은 없는 것으로 보아 CSRF 토큰 설정은 없는 것을 확인하였습니다. 해당 id 파라미터에 다양한 값을 입력하여 본 결과 1~5까지 입력 시 데이터가 출력되는 것을 확인하였으나, 6 이상의 숫자를 입력하였을 때는 출력 메시지가 없고, 다른 문자나 특수문자 등을 입력하였을 때에는 Syntax 에러가 발생하는 것을 확인하였습니다. 그래서 DVWA에서 제공하는 소스코드를 확인해보니 “mysql\_escape\_string()” 메소드가 적용되어 공격에 활용되는 특정 특수문자가 입력된 경우 이스케이프문자를 붙여 필터링하는 식의 기능이 구성되어 있었습니다. 그래서 html entity code로 인코딩 하거나, URL 형식으로 인코딩 해서 입력해보는 등 다양한 방법을 시도해보았습니다. 하지만 문제는 다른 방식으로 해결되었습니다. 소스 코드를 다시 확인해보니 아래와 같이 Where 절에 작은 따옴표가 빠져있었습니다.

|  |
| --- |
| $query = "SELECT first\_name, last\_name FROM users WHERE user\_id = $id;"; |

그래서 Fatal Error 가 발생하지 않는 integer 형식의 데이터(예를 들어 1,2,3,…)를 맨 앞에 넣어주고 뒤에 UNION SELECT 구문을 넣어주니 원하는 정보가 출력되었습니다.



작은따옴표는 앞의 Where 조건문을 닫기 위해 항상 사용해야 된다고 생각해왔었는데 이 부분이 문제 풀이과정을 돌아가게 만들었던 것으로 보입니다. High, Impossible 난이도에 대해서는 시간이 부족하여 다음에 개인적으로 확인해보도록 하겠습니다..

위에서 실습한 것과 같이 SQL injection은 적절한 보안 조치가 되어 있지 않으면 대상 서버의 DB에 포함되어 있는 모든 데이터를 확인할 수 있습니다. 그리고 이 데이터는 다른 공격의 수단으로 사용되어 시스템에 큰 영향을 끼칠 수 있습니다. 이러한 SQL injection에 대한 대응방안으로는 잘 알려진 “Prepared Statement”가 있습니다. 입력 받은 값을 그대로 쿼리에 넣는 것이 아닌 다른 변수를 설정합니다. 그리고 해당 변수에 입력 값을 저장하고, 입력 값에 대한 검증 과정을 거친 뒤 쿼리문에 넣어 동작하도록 합니다. 위의 실습에서는 Prepared Statement가 없거나, 특정 특수문자 앞에 백슬래시를 추가하는 정도(mysql\_escape\_string)의 보안 요소만 적용되어 있었기 때문에 SQL injection 공격에 성공할 수 있었습니다.

결론

이전 DB 강의 시간에 잠시 활용해본 SQL injection에 더해 XSS, CSRF, File Upload, File Inclusion 등에 대해 관련 개념을 공부해보고 직접 실습해볼 수 있는 시간이었습니다. 이에 더해 각 취약점에 대한 대응방안이 강의에 포함되어 있어, 취약점을 확인하고 해당 취약점을 대상으로 공격해보고, 이에 대한 대응방안까지의 스토리라인을 이해할 수 있는 시간이었다고 생각합니다. 그리고 DVWA라는 웹 취약점 테스트 사이트, Burp Suite라는 프록시 프로그램 등을 사용해본 경험도 분명 이후 교육 및 취업 후 실무에서 저를 더 이해하도록 도와주는 도구가 되리라 생각됩니다.