리팩토링 41 주차

DVWA 실습 보고서

Command Injection, Brute Force (Medium)

목차

1. 개요

1.1 프로젝트 목적

본 프로젝트는 *DVWA(Damn Vulnerable Web Application)*를 이용하여 웹 애플리케이션 취약점인 *Command Injection*과 *Brute Force* 공격 실습과 그에 따른 공격 기법과 대응 방법을 학습하는 것을 목적으로 한다.

1.2 실습 환경

- VMware : 17.6.2
- Kali Linux: 2025.2
- Apache : 2.4.63
- PHP : 8.4.8
- MariaDB : 11.8.1
- DVWA : 2.5
- DVWA Security Level: Medium
- 브라우저: Firefox

2. Command Injection 실습

2.1 Command Injection

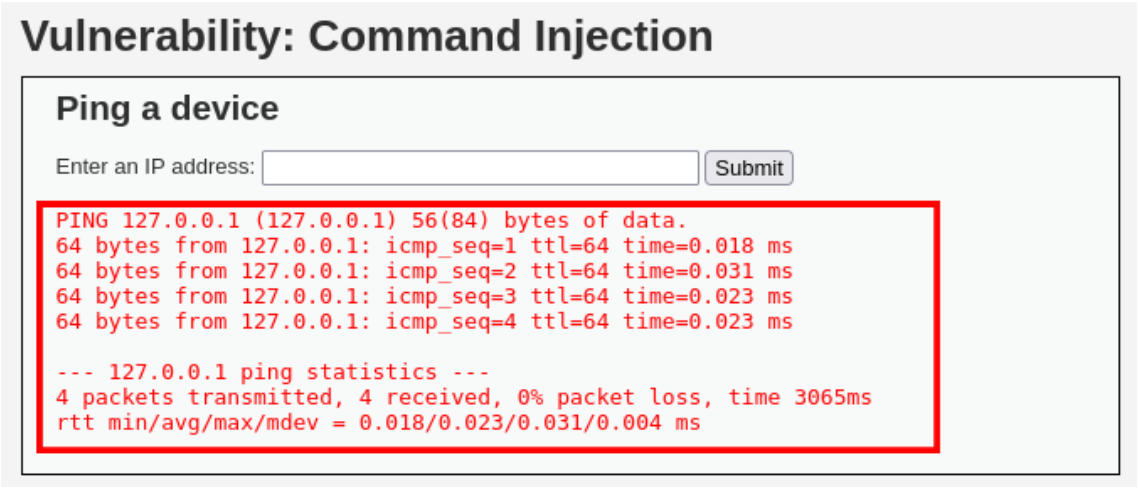
Command Injection 공격이란 웹 애플리케이션 등에서 사용자의 입력값이 OS 명령어를 실행하는 함수에 직접 전달되면서 공격자가 임의의 시스템 명령어를 실행할 수 있는 공격 방법이다.

사용자가 입력한 값이 검증 없이 시스템 명령어 실행 구분(system, exec 등)에 포함될 때 발생한다.

웹 애플리케이션에서 Command Injection 공격이 가능할 경우 해당 애플리케이션을 실행하고 있는 운영체제 계정의 권한을 공격자가 그대로 사용 가능하므로 파일 삭제, 정보 노출, 루트 권한 탈취 등의 피해가 발생할 수 있다.

2.2 Command Injection 실습

DVWA 실습 페이지의 왼쪽 탭에서 'Command Injection' 탭을 누르면 IP 주소를 입력하여 Ping 명령어를 실행하고 그 결과를 출력하는 페이지가 나온다.



시험삼아 127.0.0.1을 입력해보면 입력한 IP주소에 PING을 보내고 그 결과를 출력해주는 것을 확인할 수 있다.

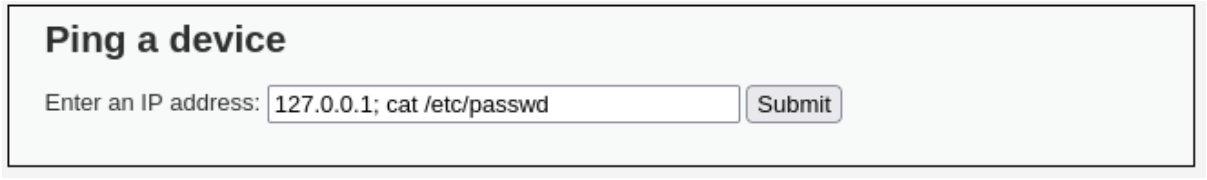
```php
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Set blacklist
    $substitutions = array(
        '&&' => '',
        ';'  => '',
    );

    // Remove any of the characters in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

Command Injection 페이지의 소스코드를 확인해보면 위와 같다.

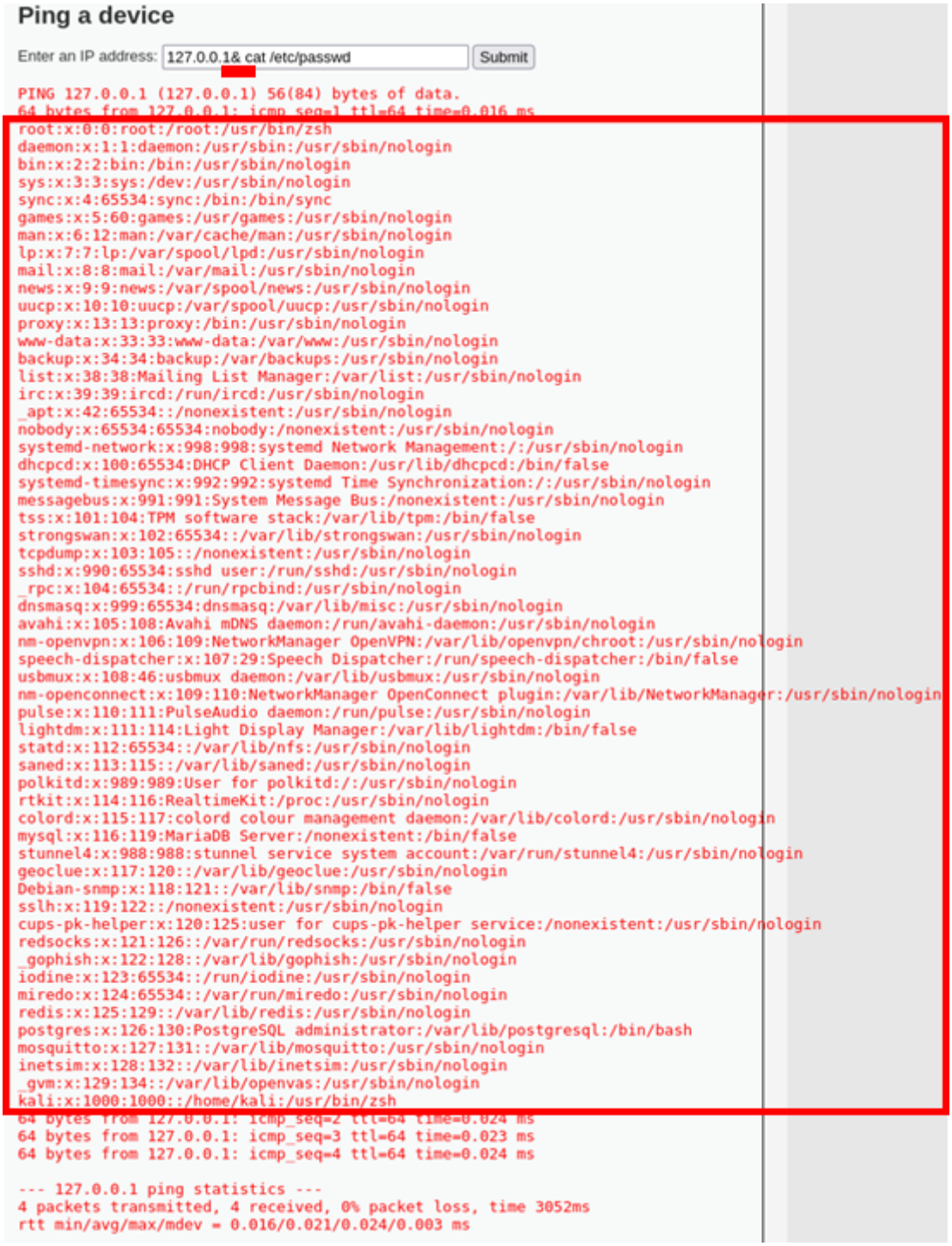
소스코드의 동작은 아래와 같다.

1. 입력창의 값이 저장되어 있는 $target과 '&&'와 ';'를 ''(공백)로 치환하는 블랙리스트가 존재한다.
2. str_replace 함수를 통해서 $target의 '&&'와 ';'문자를 공백으로 치환한다.
3. shell_exec 함수를 통해서 "ping $target값" 형식으로 쉘 명령을 실행한다.
4. 반환된 값을 <pre>...</pre>로 감싸 출력한다.

입력창의 값에서 '&&'와 ';'만 제거하고 그대로 shell_exec함수를 통해 쉘 명령을 실행한다. '&&'와 ';'가 쉘의 명령어 구분자이기 때문에 이를 필터링하려고 한 것을 알 수 있다. 하지만 그 외의 쉘 명령어 구분자인 &, |, ', (,), || 등은 필터링하지 않았기에 Command Injection 공격이 가능하다.



따라서 "127.0.0.1: cat /etc/passwd" 같이 Command Injection 공격 시에는 ';'이 필터링되어 쉘에서 오류가 발생해 아무것도 출력되지 않는다.



하지만 공백으로 치환되는 '&&', ';'를 제외한 구분자를 사용하여 공격하면 정상적으로 "cat /etc/passwd/" 명령어의 결과가 출력되는 것을 볼 수 있다.

본 프로젝트에서는 '&'를 사용하였다. '&'는 백그라운드 연산자로 "ping 127.0.0.1"은 백그라운드에서 실행되고 "cat /etc/passwd"는 포그라운드에서 실행된다.

2.3 Command Injection 대응방안

- 코딩 시 shell_exec, system, exec 등과 같이 OS 명령어를 직접 호출하는 함수를 사용하지 말고 언어에서 제공하는 안전한 함수를 사용한다.
- 사용자의 입력값을 신뢰하지 않고, 입력이 허용된 값만 통과시키는 화이트리스트 정책을 사용한다.
- 다중 명령어를 입력할 수 있는 특수문자 등의 입력을 제한한다.
- 해당 서버나 계정에 시스템 접근 권한을 최소로 부여해야 한다.

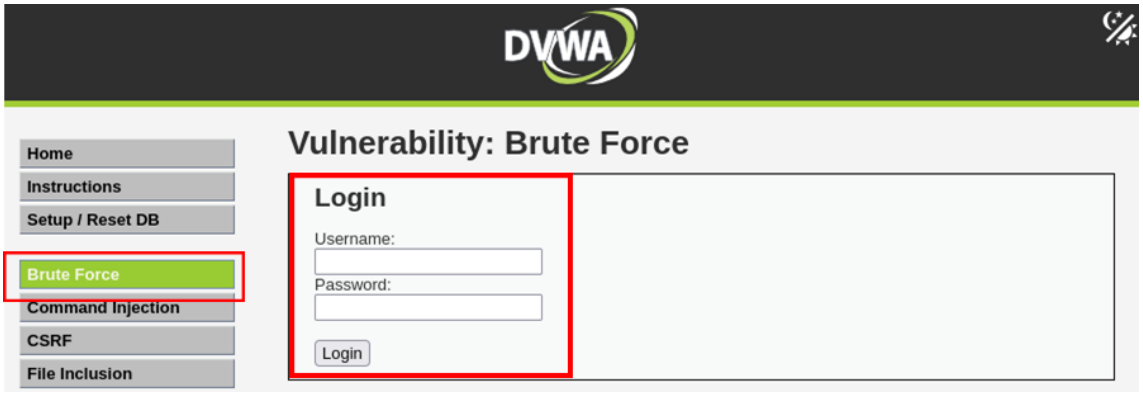
3. Brute Force

3.1 Brute Force

Brute Force(무차별 대입) 공격이란 비밀번호나 암호화 키, 로그인 정보 등 보호된 데이터를 알아내기 위해 가능한 모든 조합을 체계적으로 반복해서 시도하는 공격을 의미한다.

Brute Force 공격을 통해서 비밀번호나 인증 정보를 알아내어 계정 탈취가 일어날 수 있고 그에 따른 정보 유출, 시스템 장악, 금전적 피해 등이 발생할 수 있다.

3.2 Brute Force 실습



DVWA 실습 페이지의 왼쪽 탭에서 'Brute Force' 탭을 눌러보면 로그인 창을 볼 수 있다.

유저 이름과 비밀번호를 알고 있는 DVWA 관리자 계정으로 로그인을 해본 결과 로그인

성공 메시지가 뜬 것을 확인할 수 있다.

잘못된 비밀번호를 입력한 결과 약간의 딜레이 후에 비밀번호가 잘못되었다는 메시지가 하단에 출력되는 것을 확인했다.

```php
<?php

if( isset( $_GET[ 'Login' ] ) ) {
        // Sanitise username input
        $user = $_GET[ 'username' ];
        $user = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $user ) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

        // Sanitise password input
        $pass = $_GET[ 'password' ];
        $pass = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass ) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass = md5( $pass );

        // Check the database
        $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
        $result = mysqli_query($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>' );

        if( $result && mysqli_num_rows( $result ) == 1 ) {
                // Get users details
                $row    = mysqli_fetch_assoc( $result );
                $avatar = $row["avatar"];

                // Login successful
                $html .= "<p>Welcome to the password protected area {$user}</p>";
                $html .= "<img src=\"{$avatar}\" />";
        }
        else {
                // Login failed
                sleep( 2 );
                $html .= "<pre><br />Username and/or password incorrect.</pre>";
        }

        ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

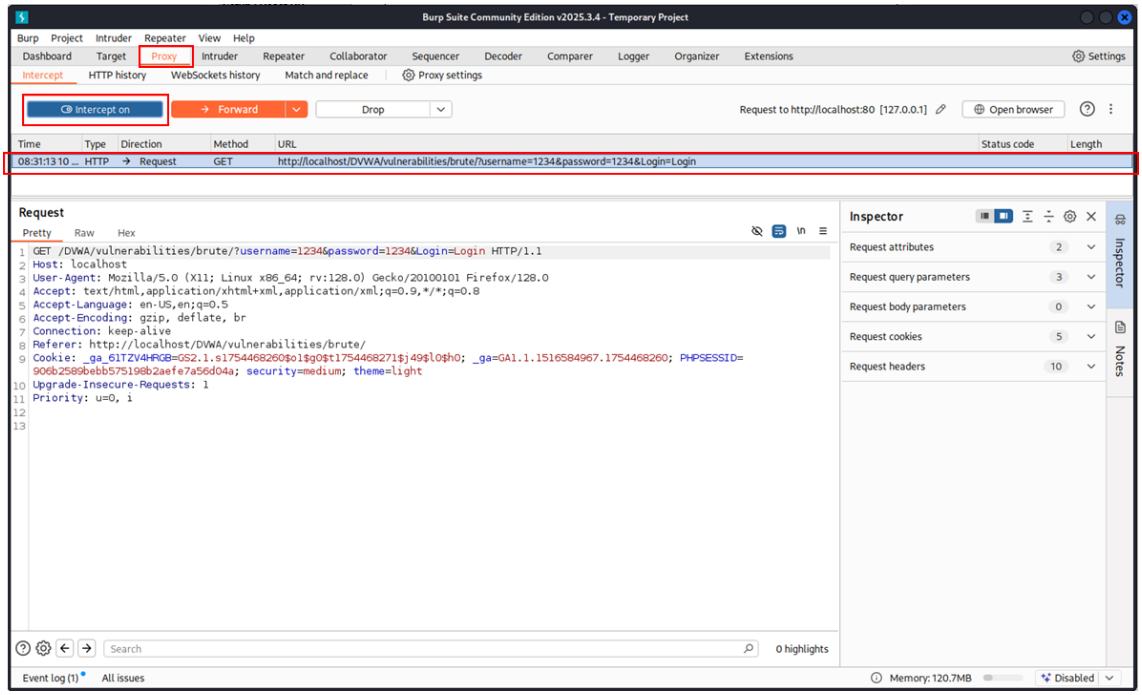
Brute Force 페이지의 소스코드를 확인해보면 위와 같다.

소스코드의 동작은 아래와 같다.

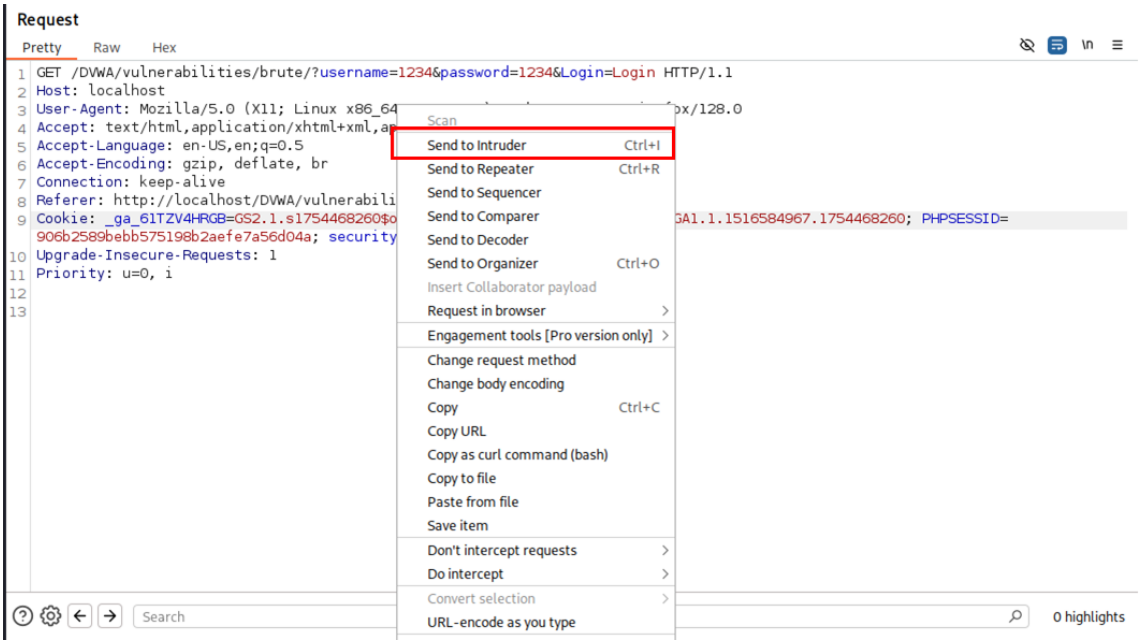
1. 유저명과 비밀번호를 가져와서 SQL 이스케이프 처리를 한다. 비밀번호는 MD5 해시로 변환한다.
2. users 테이블에서 유저명과 비밀번호를 동시에 만족하는 레코드를 검색한다.
3. 결과가 정확히 1개면 로그인 성공으로 간주하며 그 외는 실패로 간주한다.
4. 로그인 성공 시 성공 이미지와 메시지를 출력한다.
5. 로그인 실패 시 2초가 지연되며 로그인 실패 메시지를 출력한다.

이전 난이도인 Low에서는 로그인 실패에 따른 아무런 조치도 취하지 않았지만 medium 난이도에서는 로그인 실패 시 2초를 지연시켜 Brute Force 공격의 공격 속도는 늦춘다. 하지만 근본적으로 Brute Force 공격을 막을 수는 없다.

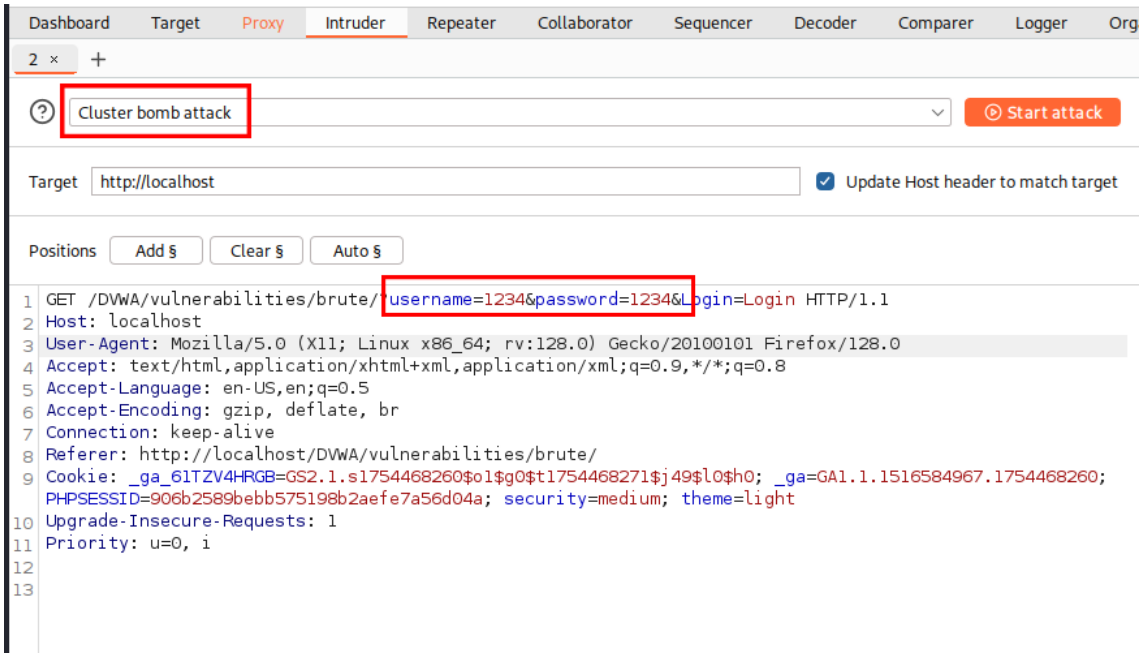
본 프로젝트에서는 Burp Suite의 Intruder 기능을 이용하여 Brute Force 공격을 시도할 것이다.



Burp Suite를 실행시키고 Proxy > Intercept 탭으로 이동하여 Intercept on으로 설정한다. 그리고 로그인 페이지에 아무 값이나 입력하고 로그인 버튼을 누르면 위와 같이 로그인 요청 패킷을 볼 수 있다.



Burp Suite의 Request 창에서 마우스 오른쪽 클릭을 하면 선택 메뉴가 나오는데 그 중에서 "Send to Intruder를 클릭하여 해당 패킷을 Intruder로 전송한다.

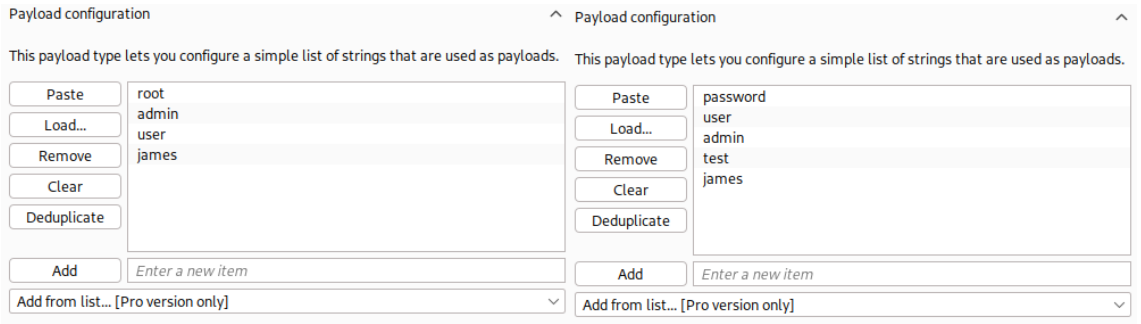


Intruder 탭으로 이동하여 상단의 Attack Type을 "Cluster bomb attack"으로 바꿔준다. Cluster bomb attack이란 패킷 내부의 페이로드를 지정하여 페이로드 포지션에 사전에 입력해 놓은 아이디, 비밀번호 리스트를 조합하여 전부 시도해보는 공격 타입이다.

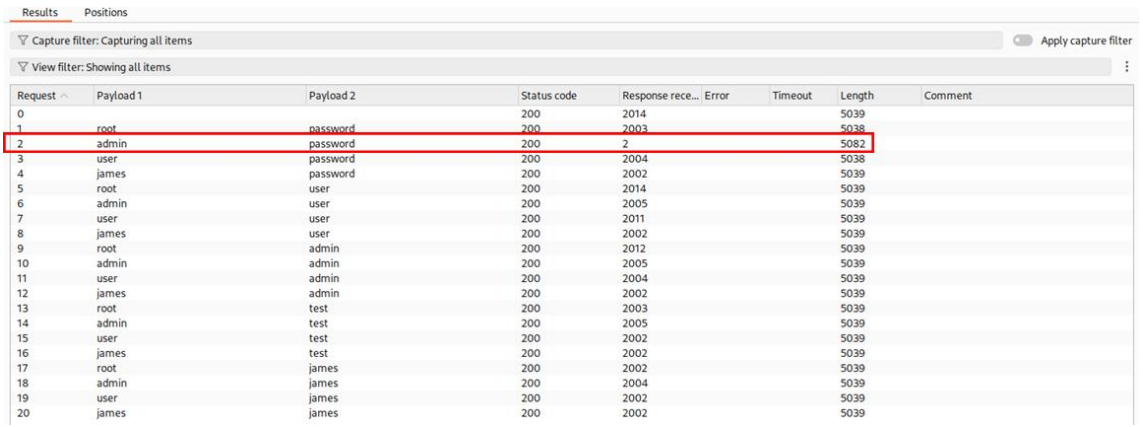
위의 사진을 확인해보면 아까 입력했던 유저 이름과 비밀번호를 확인할 수 있다. Brute

Force를 통해서 유저 이름과 아이디를 시도해봐야 하기 때문에 username과 password를 페이로드 포지션으로 설정해야 한다.

페이로드 포지션으로 설정하고 싶은 부분을 드래그하여 Add § 버튼을 누르면 설정된다.



설정한 페이로드 포지션을 클릭해보면 Payload configuration이라는 설정이 나온다. 여기에 예상되는 아이디, 비밀번호를 입력하면 된다. 본 프로젝트에서는 위 사진처럼 설정했다.



Request	Payload 1	Payload 2	Status code	Response rece...	Error	Timeout	Length	Comment
0			200	2014			5039	
1	root	password	200	2003			5038	
2	admin	password	200	2			5082	
3	user	password	200	2004			5038	
4	james	password	200	2002			5039	
5	root	user	200	2014			5039	
6	admin	user	200	2005			5039	
7	user	user	200	2011			5039	
8	james	user	200	2002			5039	
9	root	admin	200	2012			5039	
10	admin	admin	200	2005			5039	
11	user	admin	200	2004			5039	
12	james	admin	200	2002			5039	
13	root	test	200	2003			5039	
14	admin	test	200	2005			5039	
15	user	test	200	2002			5039	
16	james	test	200	2002			5039	
17	root	james	200	2002			5039	
18	admin	james	200	2004			5039	
19	user	james	200	2002			5039	
20	james	james	200	2002			5039	

Start attack 버튼을 눌러 공격을 시작하면 결과가 위처럼 나온다. 응답 패킷의 길이부분을 잘 보면 대부분의 패킷은 길이가 비슷한데 2번 요청의 패킷의 길이가 다른 패킷보다

40정도 긴 것을 알 수 있다.

2번 요청의 값인 admin, password를 유저 이름, 비밀번호 입력창에 입력하여 로그인에 성공했다.

3.3 Brute Force 대응방안

- 대소문자, 숫자, 특수문자를 조합한 긴 비밀번호(10자리 이상 등)를 사용하도록 비밀번호 정책을 강화한다.
- 일정 횟수 이상 로그인 실패 시 계정을 일시 잠금하거나, 추가 인증이 필요하도록 해야 한다.
- CAPTCHA를 적용하여 사람이 아닌 자동화 도구의 접근을 차단한다.
- 비밀번호 외에 OTP, SMS 코드 등의 추가 인증 절차를 요구해야 한다.
- WAF(Web Application Firewall)에 자동화 공격을 탐지하고 차단하는 시스템을 마련한다.

4. 결론

4.1 정리

공격 벡터	Command Injection	Brute Force
공격 표면	웹 사이트 사용자 입력폼 - 입력값이 검증/필터링이 미흡 - 서버 OS 명령에 직접 전달	로그인 폼, 인증 입력란, 비밀번호 입력창 등의 인증 정보를 입력받는 모든 접점
주요 피해	서버 루트권한 탈취, 정보 유출	계정 탈취, 인증 우회
대응 방안	- OS 명령어 직접 호출 함수 사용 자제 - 사용자 입력값 검증/필터링 - 서버 계정 시스템 접근 권한 최소한으로 부여	- 비밀번호 정책 강화 - 반복적인 인증 시도, 실패 시 차단 및 2차 인증 요구 - WAF 기반 자동화 공격 탐지, 차단 시스템 도입