

# **CROSS-SITE SCRIPTING (XSS)**

Security Vulnerabilities

# **CROSS SITE SCRIPTING**

INTRODUCTION



HOW XSS  
WORKS



TEST FOR  
XSS



WHAT IS  
XSS



TYPES IN  
XSS



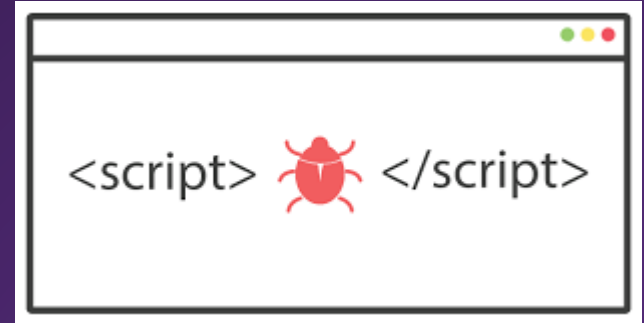
PREVENTION  
FOR XSS


## WHAT IS XSS

- Cross site scripting attack is a code injection attack that is executed on the client-side of a web application.
- It is a software or a browser that is used to interact With the web application.



- In cross site scripting we inject a malicious code onto the web browser to make the web application do something that it is not supposed to do.
- The malicious script gets executed on the web application after its get injected on the web browser. This malicious script is executed when the victim visits the web page or the web server.



- 
- This method is mainly used to steal sensitive Information like cookies, session tokens and other sensitive information like usernames and passwords.
  - Cross-site scripting is also used to modify the contents of the website by injecting malicious code onto the web server or the web browser.

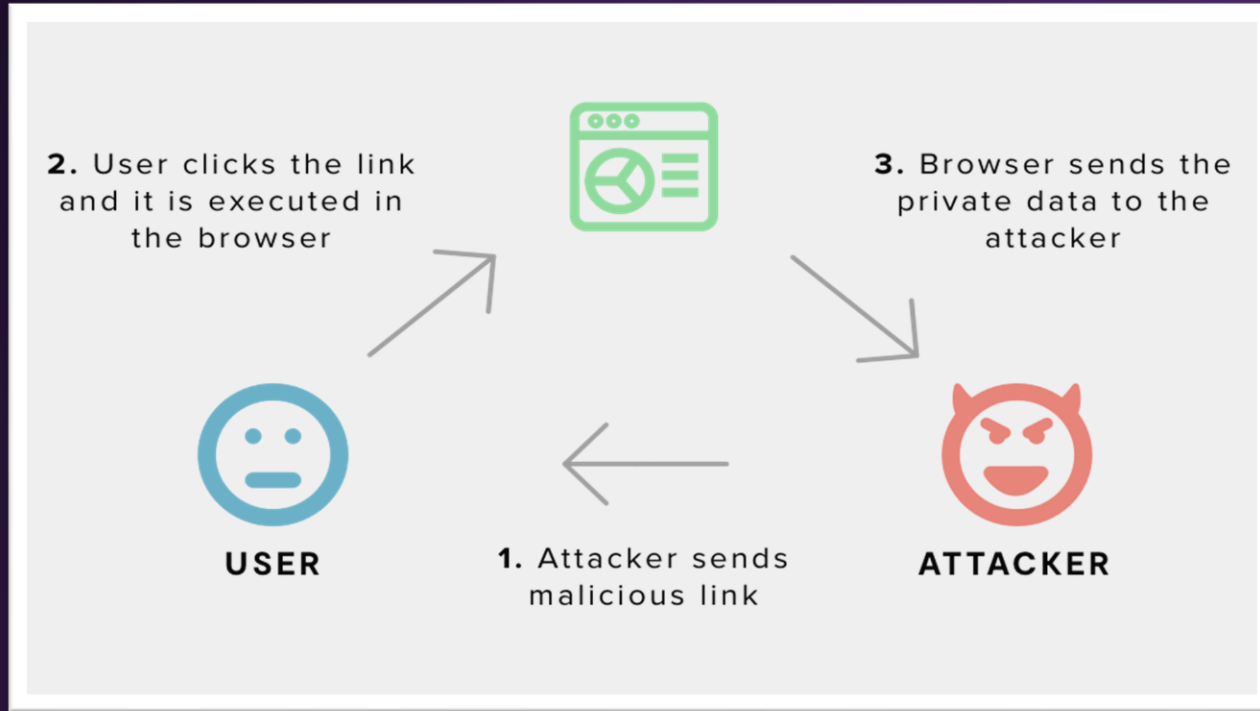
HOW

— **CROSS SITE SCRIPTING**

WORKS



# How XSS works



## — PROCEDURE

The process user enters into a website after typing the user name and password and click the submit button.

That website that data can be transferred into web browser.

Browser response can be back to the user.

After that, the attacker sends a malicious link. User click that link and it is executed in the browser. Then browser sends the private data to the attacker



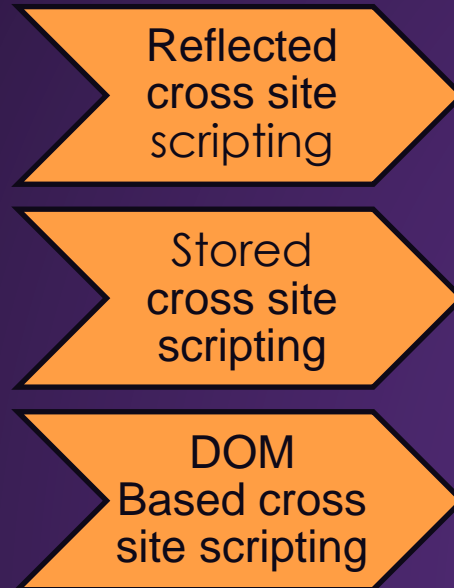


# TYPES OF — **CROSS-SITE** SCRIPTING (XSS)



# TYPES OF CROSS SITE SCRIPTING

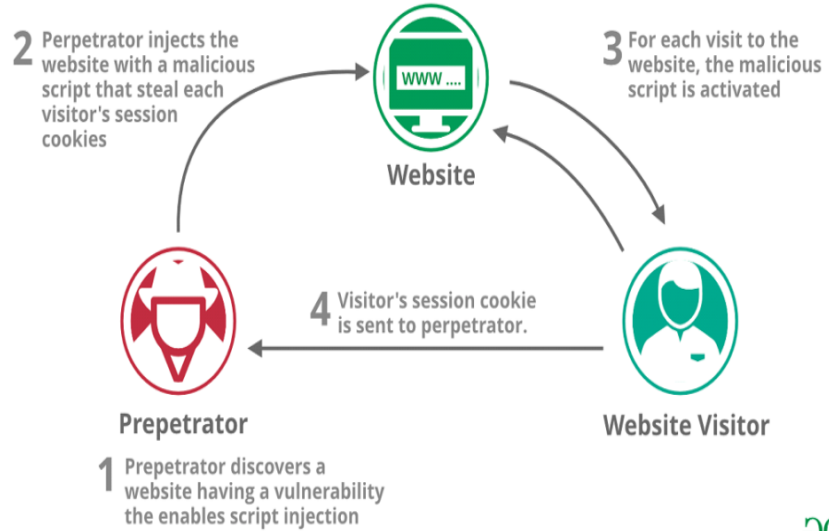
There are three types of cross site scripting



# REFLECTED CROSS-SITE SCRIPTING

- Reflected Cross Site Scripting is also called a Non-Persistent / Type-1 order XSS.
- Data is not stored in the database or web application.
- Where the malicious script comes from current HTTP.
- Script is activated through the link.

## Reflected XSS



# STORED CROSS SITE SCRIPTING

- ▶ Stored Cross Site Scripting is also called as Persistent / Type-2 order xss.
- ▶ Data is stored in the database or web application.
- ▶ Where the malicious script comes from the website's database.
- ▶ The payload is stored permanently on the target application.

# Stored XSS



1. Attacker gets malicious data into the database (no social engineering required)

2. Entirely innocent request

4. Response includes malicious data as active content

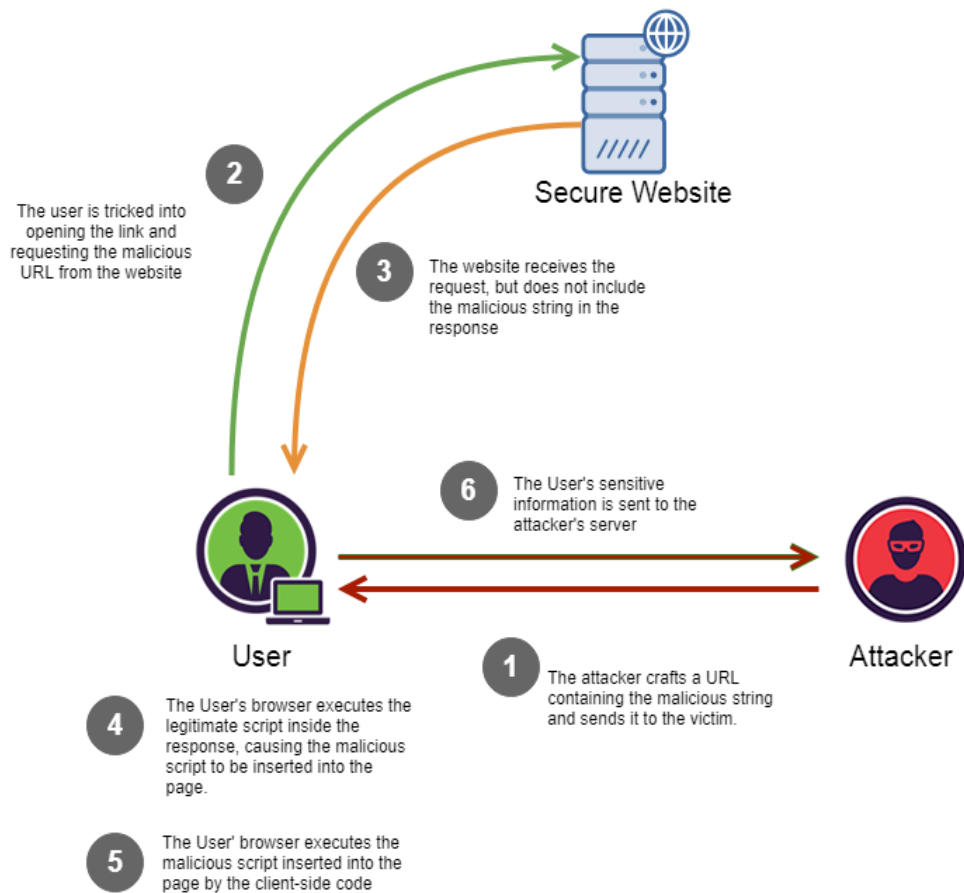
3. Bad app retrieves malicious data and uses it verbatim

5. Bad thing happens



# DOM BASED CROSS SITE SCRIPTING

- DOM Based Cross Site Scripting is also called Type-0 order XSS.
- DOM stands for Document Object Model.
- Where the vulnerability exists in client-side code rather than server-side code.
- DOM Based XSS is both persistent and non-persistent XSS.





# HOW TO TEST FOR XSS USING PwnXSS



# PwnXSS(TOOL)

**PwnXSS** is a free and open-source tool available on Github. This tool is specially designed to find cross-site scripting. This tool is written in python. You must have python 3.7 installed in your Kali Linux. There are lots of websites on the internet which are vulnerable to cross-site scripting(**XSS**). This tool makes finding cross-site **scripting** easy. This tool works as a scanner. The Internet has millions of websites and web apps a question comes into mind whether your website is safe or not. Security of our websites plays an important role. Cross-site **scripting** or **XSS** is a vulnerability that can be used to hack websites. This tool helps to find such vulnerability easily.



# — HOW TO INSTALL PwnXSS?

Create a new directory

>>mkdir pwnxss

>>cd pwnxss

Now install bs4

>>pip3 install bs4

Now install requests

>>pip3 install requests

```
(kali@kali)~$ sudo su
[sudo] password for kali:
(root@kali)~$ cd Desktop
(root@kali)~/Desktop$ mkdir pwnxss
(root@kali)~/Desktop$ cd pwnxss
(root@kali)~/Desktop/pwnxss$ pip3 install bs4
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: beautifulsoup4 in /usr/lib/python3/dist-packages (from bs4) (4.11.1)
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... done
  Created wheel for bs4: filename=bs4-0.0.1-py3-none-any.whl size=1272 sha256=04cce13afacd396bab147e516f40743925a2fe277e63ae1551bc583d3a3494d4
  Stored in directory: /root/.cache/pip/wheels/25/42/45/b773edc52acb16cd2db4cf1a0b47117e2f69bb4eb30e0e70
Successfully built bs4
Installing collected packages: bs4
  WARNING: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
  crackmapexec 2.1.2 requires nmap4<1.9.0, >4.1.1, but you have nmap4 1.7.0.dev0 which is incompatible.
  crackmapexec 2.1.2 requires python34<4.0.0, >3.6.0, but you have python3 3.4.3 which is incompatible.
Successfully installed bs4-0.0.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

(root@kali)~/Desktop/pwnxss$ pip3 install bs4
Requirement already satisfied: bs4 in /usr/local/lib/python3.10/dist-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in /usr/lib/python3/dist-packages (from bs4) (4.11.1)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

(root@kali)~/Desktop/pwnxss$ pip3 install requests
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (2.27.1)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

(root@kali)~/Desktop/pwnxss$ git clone https://github.com/pwn0sec/PwnXSS-
Cloning into 'PwnXSS-' ...
fatal: unable to access 'https://github.com/pwn0sec/PwnXSS-': The requested URL returned error: 400
```

# CLONING PwnXSS

Now you have to install the tool. This means you have to clone the tool from github using the following command.

```
>>git clone
```

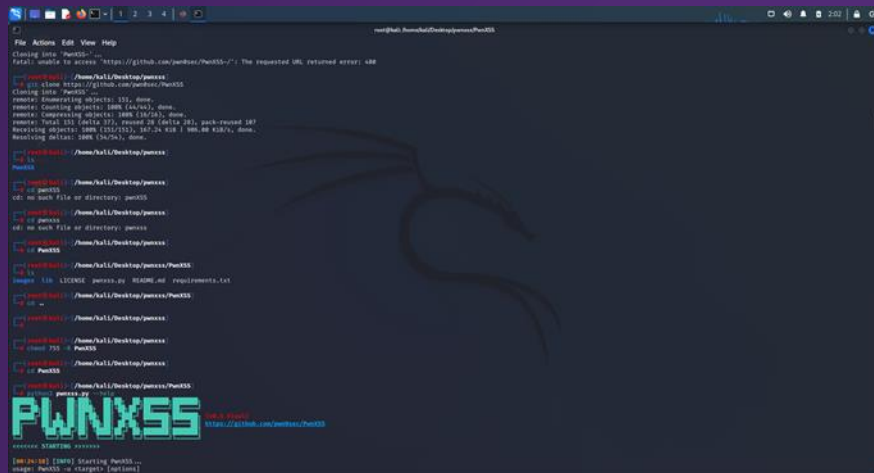
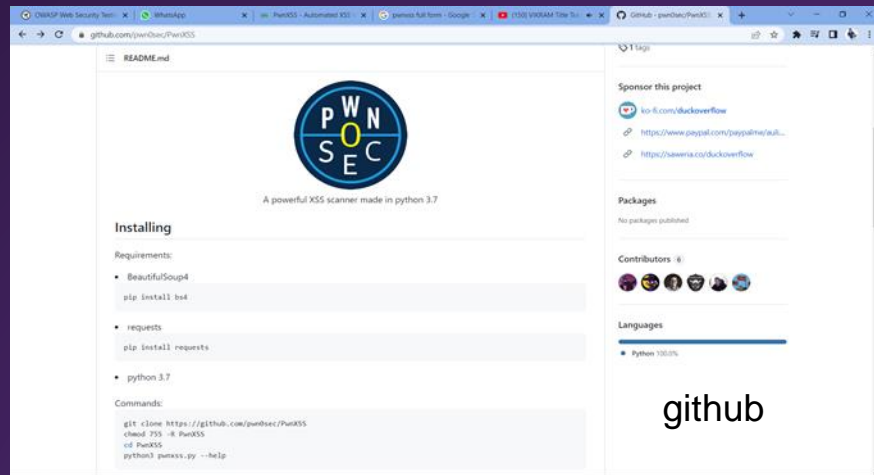
```
https://github.com/pwn0sec/PwnXSS
```

now a PwnXSS file will be created.

This pwnxss will be stored in

```
>>root@kali-
```

```
>>/home/kali/Desktop/pwnxss/PwnXSS
```



You can see the tool in PwnXSS. Now give permissions to that tool

```
>>chmod 755 -R PwnXSS
```

Use the following command is used to see the help index of the tool.

```
>>python3 pwnxss.py --help
```



```
PWNXSS (v0.5 Final)
https://github.com/pwn0sec/PwnXSS

<<<<< STARTING >>>>>

[02:15:36] [INFO] Starting PwnXSS...
usage: PwnXSS -u <target> [options]

Options:
  --help            Show usage and help parameters
  -u                Target url (e.g. http://testphp.vulnweb.com)
  --depth           Depth web page to crawl. Default: 2
  --payload-level   Level for payload Generator, 7 for custom payload. {1...6}. Default: 6
  --payload         Load custom payload directly (e.g. <script>alert(2005)</script>)
  --method          Method setting(s):
                   0: GET
                   1: POST
                   2: GET and POST (default)
  --user-agent      Request user agent (e.g. Chrome/2.1.1/...)
  --single          Single scan. No crawling just one address
  --proxy           Set proxy (e.g. {'https':'https://10.10.1.10:1000'})
  --about           Print information about PwnXSS tool
  --cookie          Set cookie (e.g {'ID':'1094200543'})

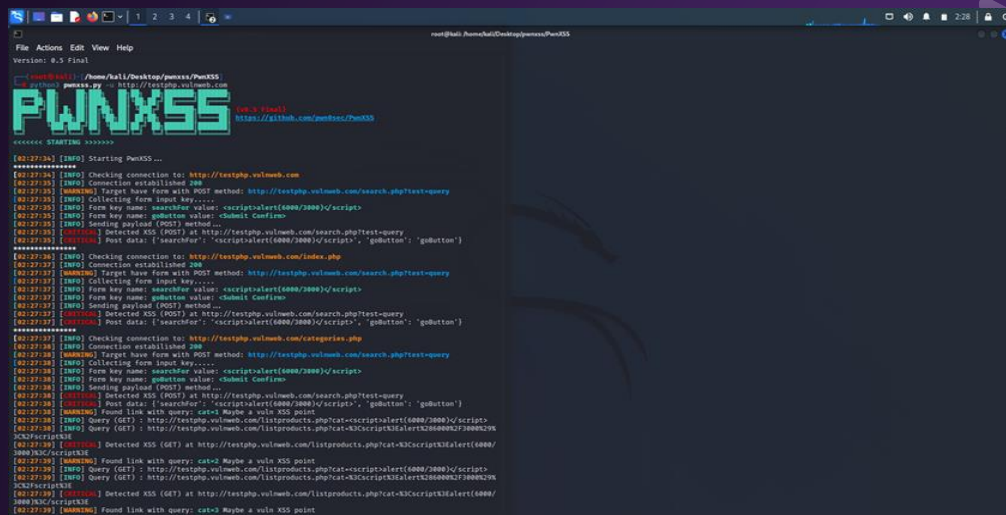
Github: https://www.github.com/pwn0sec/PwnXSS
Version: 0.5 Final

root@kali:~/Desktop/pwnxss/PwnXSS
```

# Tool usage

The tool has been downloaded successfully using this tool you can easily check the cross-site scripting vulnerabilities of the websites and webapps. Now here are some examples to use the PwnXSS tool.

```
>>python3 pwnxss.py -u http://testphp.vulnweb.com
```



```
root@kali: /home/kali/Desktop/pwnxss/PwnXSS
File Actions Edit View Help
version: 0.5 Final

PWNXSS 0.5 (Final)
https://github.com/pwn0x0/PwnXSS

##### STARTING #####

[02:27:34] [INFO] Starting PwnXSS...
#####
[02:27:34] [INFO] Checking connection to: http://testphp.vulnweb.com
[02:27:35] [INFO] Connection established OK
[02:27:35] [INFO] Target have form with POST method: http://testphp.vulnweb.com/search.php?test-query
[02:27:35] [INFO] Collecting form input key....
[02:27:35] [INFO] Form key name: searchbar value: <script>alert(6000/2000)</script>
[02:27:35] [INFO] Form key name: gobutton value: <Submit Confirm>
[02:27:35] [INFO] Sending payload (POST) method ...
[02:27:35] [INFO] Detected XSS (POST) at http://testphp.vulnweb.com/search.php?test-query
[02:27:35] [INFO] Post data: {'searchbar': '<script>alert(6000/2000)</script>', 'gobutton': 'gobutton'}
#####
[02:27:36] [INFO] Checking connection to: http://testphp.vulnweb.com/index.php
[02:27:37] [INFO] Connection established OK
[02:27:37] [INFO] Target have form with POST method: http://testphp.vulnweb.com/search.php?test-query
[02:27:37] [INFO] Collecting form input key....
[02:27:37] [INFO] Form key name: searchbar value: <script>alert(6000/2000)</script>
[02:27:37] [INFO] Form key name: gobutton value: <Submit Confirm>
[02:27:37] [INFO] Sending payload (POST) method ...
[02:27:37] [INFO] Detected XSS (POST) at http://testphp.vulnweb.com/search.php?test-query
[02:27:37] [INFO] Post data: {'searchbar': '<script>alert(6000/2000)</script>', 'gobutton': 'gobutton'}
#####
[02:27:37] [INFO] Checking connection to: http://testphp.vulnweb.com/categories.php
[02:27:38] [INFO] Connection established OK
[02:27:38] [INFO] Target have form with POST method: http://testphp.vulnweb.com/search.php?test-query
[02:27:38] [INFO] Collecting form input key....
[02:27:38] [INFO] Form key name: searchbar value: <script>alert(6000/2000)</script>
[02:27:38] [INFO] Form key name: gobutton value: <Submit Confirm>
[02:27:38] [INFO] Sending payload (POST) method ...
[02:27:38] [INFO] Detected XSS (POST) at http://testphp.vulnweb.com/search.php?test-query
[02:27:38] [INFO] Post data: {'searchbar': '<script>alert(6000/2000)</script>', 'gobutton': 'gobutton'}
[02:27:38] [INFO] Found link with query: cat=0 Maybe a vuln XSS point
[02:27:38] [INFO] Query (GET) : http://testphp.vulnweb.com/listproducts.php?cat=K3CscriptX3falerT260000JF3000Z9X
[02:27:38] [INFO] Query (GET) : http://testphp.vulnweb.com/listproducts.php?cat=K3CscriptX3falerT260000JF3000Z9X
[02:27:39] [INFO] Detected XSS (GET) at http://testphp.vulnweb.com/listproducts.php?cat=K3CscriptX3falerT(6000/
[02:27:39] [INFO] Found link with query: cat=0 Maybe a vuln XSS point
[02:27:39] [INFO] Query (GET) : http://testphp.vulnweb.com/listproducts.php?cat=K3CscriptX3falerT(6000/2000)</script>
[02:27:39] [INFO] Query (GET) : http://testphp.vulnweb.com/listproducts.php?cat=K3CscriptX3falerT260000JF3000Z9X
[02:27:39] [INFO] Query (GET) : http://testphp.vulnweb.com/listproducts.php?cat=K3CscriptX3falerT260000JF3000Z9X
[02:27:39] [INFO] Detected XSS (GET) at http://testphp.vulnweb.com/listproducts.php?cat=K3CscriptX3falerT(6000/
[02:27:39] [INFO] Found link with query: cat=0 Maybe a vuln XSS point
```

Now we can see here at <http://testphp.vulnweb.com/search.php?test=query>  
The script "<script>alert(6000/3000)</script>" got executed.  
With that script we can inject our malicious script and can hack the victim system.

```
[INFO] Sending payload (POST) method ...  
[CRITICAL] Detected XSS (POST) at http://testphp.vulnweb.com/search.php?test=query  
[CRITICAL] Post data: {'searchFor': '<script>alert(6000/3000)</script>', 'goButton': 'goButton'}  
[WARNING] Found link with query: pic=6 Maybe a vuln XSS point  
[INFO] Query (GET) : http://testphp.vulnweb.com/product.php?pic=<script>alert(6000/3000)</script>  
[INFO] Query (GET) : http://testphp.vulnweb.com/product.php?pic=%3Cscript%3Ealert%286000%2F3000%29%3C%2Fs
```

# HOW TO PREVENT — **CROSS-SITE** SCRIPTING (XSS)





## PREVENTION METHODS

- User Input Escaping
- Consider Every Input As A Threat
- Data Validation
- Sanitize Data
- Encode Output
- Right Response Headers
- Content Security Policy



## — USER INPUT ESCAPING

- In user input escaping, we remove the special feature of the characters such as greater than symbol > or lesser than symbol < that can be used in tags or in malicious scripts.
- The user input is escaped by treating these characters only as a text character



## — **CONSIDER ALL INPUT AS A THREAT**

- We have to consider every input as a threat as the user has complete control over what input he gives you.
- At the point where user input is received, filter as strictly as possible based on what is expected, sanitize, and handle every input with care.



## — DATA VALIDATION

- Data validation is done when you know the generic format of the input.
- Assume that we have a field of input for email. An input should only be allowed if it sustains in the specific format of a username, @, domain address.
- By using regular expressions, data validation can take place.



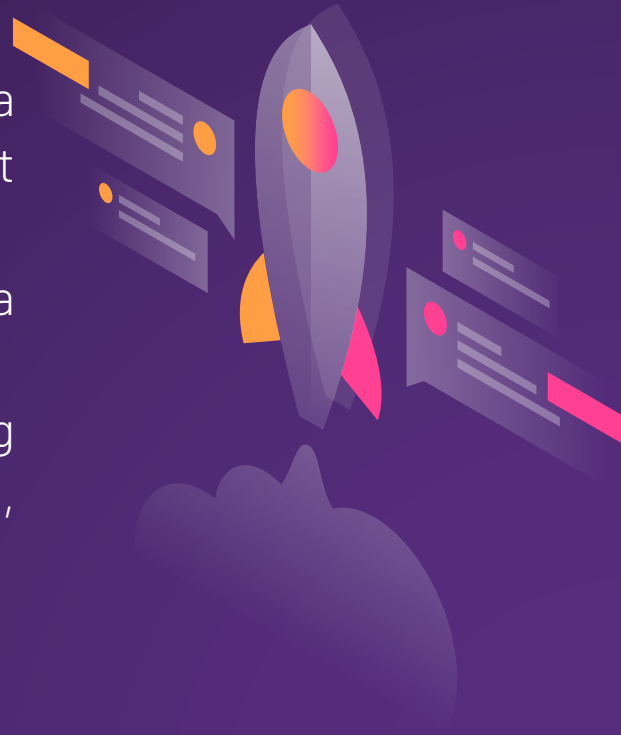
## — SANITIZE DATA

- Sanitization is typically performed by using either a whitelist or a blacklist approach.
- Basic tags for changing fonts are often allowed, such as `<b>`, `<i>` while more advanced tags such as `<script>`, `<embed>`, and `<link>` are removed.
- Also potentially dangerous attributes such as the `onclick` attribute are removed in order to prevent malicious code from being injected.



## — ENCODE OUTPUT

- Output Encoding is used to safely display data exactly as a user typed it in. Variables should not be interpreted as code instead of text.
- When you encode data, it's no longer can act as a malicious script.
- There are many different output encoding methods because browsers parse HTML, JS, URLs, and CSS differently.



## — RIGHT RESPONSE HEADERS

- There are different types of HTTP headers that are used to pass additional information with HTTP responses or HTTP requests.
- We can decide what the response headers should be, what data can be sent, or what data can be received.
- For such, X-XSS-Protection in HTTP header is a feature that stops a page from loading when it detects XSS attacks



## — CONTENT SECURITY POLICY

- As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.
- It works by restricting the resources (such as scripts and images) that a page can load and restricting whether a page can be framed by other pages.
- To enable CSP, a response needs to include an HTTP response header called Content-Security-Policy with a value containing the policy.





— **THANKS!**

