

Chapter. 1

Computer security def by NIST ::

The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and Confidentiality of information system resources.

The CIA Triad ::



Confidentiality :: Information that is sensitive must be kept private. Only people who can access it must be able to view the data.

Differential privacy falls under this.

Integrity :: Unauthorised users should not be able to modify the data.

Availability :: The people who have access must be able to access it. That is data must be available to them.

Authenticity ::

Accountability :: If someone deletes something in system, then we must be able to see who deleted. Must not be able to delete.

Web application attacks :-

1. CSRF :- (Cross site request for)

visits
victim ~~hits~~ a malicious n/w bad guy.com but when he visits this website he is logged on to some sensitive website like BoA.com (logged in). The webpage badguy.com is in victims browser that ^{makes request to} login BoA.com. Server. By this money may transfer to malicious account. A page from badguy.com has made request to BoA.com. So this is cross site request.

If request is made to same server this is called Same-site request.

Cross site request is necessary in many applications.

Suppose your website has a image. This image is in other server. Then when you see image we go to other website. This is good application of cross site request.

- Advertisements use inline frame (iframe), here entire webpage is shown in one webpage. This is also cross site request.

In this case, cookies are used. Cookies are sent to server.

Two types of cookies

1. Persistent Cookies - Can last longer,
2. Session Cookies - last when you close browser.

Counter questions:

1. what if user is not logged into BoA.com. If not logged in then attack fails.
2. Once action taken on BoA if it shows success message, User will get doubt why success message is shown.
3. How does money get transfer without giving all the field data.

↓

Get - to open URL

They use Get or Post Commands Post - to enter data in URL.
Get:- Give HTTP header - using this they can ask questions.

↓

1. Anchor Tags can be used.
 2. we can use Javascript. In this victim don't need to click anywhere.
 3. `` can be used. If we use `` in source that has URL. The moment user visits badguy.com it will request URL. That request to BoA will trigger what to do. For sure no image will be returned as there is no image there.
- If the website is ^{not} vulnerable then we can't attack.

``

we know that there is no image, but whatever is there is or we need hide, victim must not be able to see what ever comes back for this we use infinite small size.

Defence for GET Command is

To attack Post Service :-

we are going to make use of web form.

How to fill form in webpage? -

One will not submit the form knowing that is not realised right. Then how?

Solutions for this :-

1. Make the form invisible (Because we don't want user to see form). But if we give invisible form how will we get data.

2. Populate inputs programmatically. (Then how to submit, after populating automatically).

3. Javascript Submit (Using Javascript we can Programmatically Submit). Once the webpage opens we can Programmatically Submit.

| |
|---------------------------------------|
| Receiver's acct: 1234 |
| Amount: 6000 |
| <input type="submit" value="Submit"/> |

→
<form action = "http://www.bca-con.com/mm.php" method = "post">
receiver-acct <input type = "text" name = "acct" value = "1234">

Amount <input type = "text" name = "amount" value = "6000">

<input type = "submit" value = "Submit">

</form>

Once it is submitted, post request is send to server.

Use Javascript To programmatically fill data code is in 1st notes check that.

The above form is just example of form, just to show how a form looks like.

 tags have great threat.

alt → allows to intercept with website.

form

- Javascript

```
<script type = "text/javascript" >
```

```
function Attack () {
```

```
    var form = details;
```

```
    form-details = "<input type = 'hidden' name = 'Account' />
```

```
    form-details += "value = '1234'> <input type = 'hidden'>";
```

```
    form-details += "name = 'amount' value = '6000'>";
```

```
    var Attack-form;
```

```
    Attack-form = document.createElement ("form");
```

```
    Attack-form.action = "https://...";
```

```
    Attack-form.innerHTML = form-details;
```

```
    Attack-form.method = "post";
```

```
    document.body.appendChild (Attack-form);
```

```
    Attack-form.submit();
```

```
    window.onload () = function ()
```

2. XSS (Cross-site scripting attack):

OwASP - open web application security project
XSS is basically code injection attack.

Two types of XSS attack

1. Non-persistent (Reflected XSS):

You enter some input to website, the website does operation and gives output. But output contains some exact part of input in it.

eg:- Enter, Random text in Google. We get "no results found in Random text".

Instead of random script, if we give JavaScript, that means now browser is going to execute JavaScript. But it's like attacking my self. Me giving script and attacking my self.

If we use GET new url doesn't open, we can give script as input and get output containing script. Copy URL and send to other person. when that person clicks link.

```
http://www.ABC.com/search?input = <script>  
alert("attack") </script>
```

It is non-persistent because it is not saved permanently. It's reflected because attack reflects on us also.

2. Persistent attack:

In this case attacker sends JavaScript directly to server web application. It is permanent.

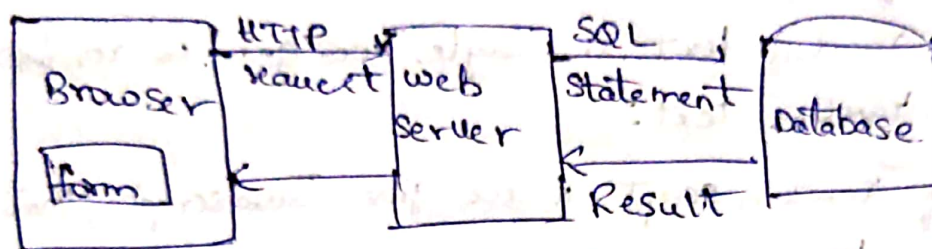
This is done by posting javascript in any social media, who ever views that post get effected.

eg:- enter javascript in comment (or) post in facebook.

What can XSS do:

1. Morphing a website
2. Steal information
3. Send malicious spoofed requests

3. SQL injection:-



```
<>php
```

```
$R-num = $_GET ['R-N']
```

```
$PwD = $_GET ['PwD']
```

SQL statement

```
SELECT Name  
From users
```

```
WHERE R-num = '$ R-num' and PwD = '$PwD'
```

```
F1: 57814 # 57814#
```

```
F2: 11235
```

is comment. It is commenting out everything.

By this we can select users even without knowing password, as password is commented.

```
f1= 57814'OR 1=1 #
```

1=1 → means whatever records have R-number

So we get all users data having R-number

4. Click Jacking:-

we click a button that does xyz, thinking it does abc.
Also called UI redressing as it plays games with UI.

They run HTML command called `iframe`. Using this we display one webpage in other page.

Two features

1. Transparency (opacity)

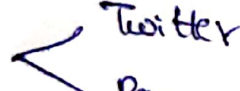
2. Overlap.

Code in 1st notes, check it.

- Commonly done through 'like' feature in website
- Also called like jacking.

Twitter (liking)

- Attract Victim

- 2 iframes  Twitter
Page (Inserted by attacker)

opacity = 0 → Totally transparent.

This can be done where ever we have single click button.

~~IF~~ user get doubt why he is not being directed to correct page (as he is clicking bad guy button). This is risk of using transparent i-frame.

So we use non transparent i-frame.

we expose the bad guy almost 100%, but not its location.

→ what if attacker wants to steal key strokes.

11235

Click Jacking: This attack can be done on actions
UI Redressing: that require one click.

iframe (Inline frame) This attack done by using transparent iframe.

```
<iframe id="A" Src="http://www.A.com/index.html"> </iframe>
<iframe id="B" Src="http://www.B.com/index.html"> </iframe>
<style type="text/css">
#A { Position: absolute; top: 100px; left: 20px; width: 100px;
    height: 100px; z-index: 1 }
#B { Position: absolute; top: 100px; left: 30px; width: 100px; height: 100px;
    z-index: 2; opacity: 0.5 }
```