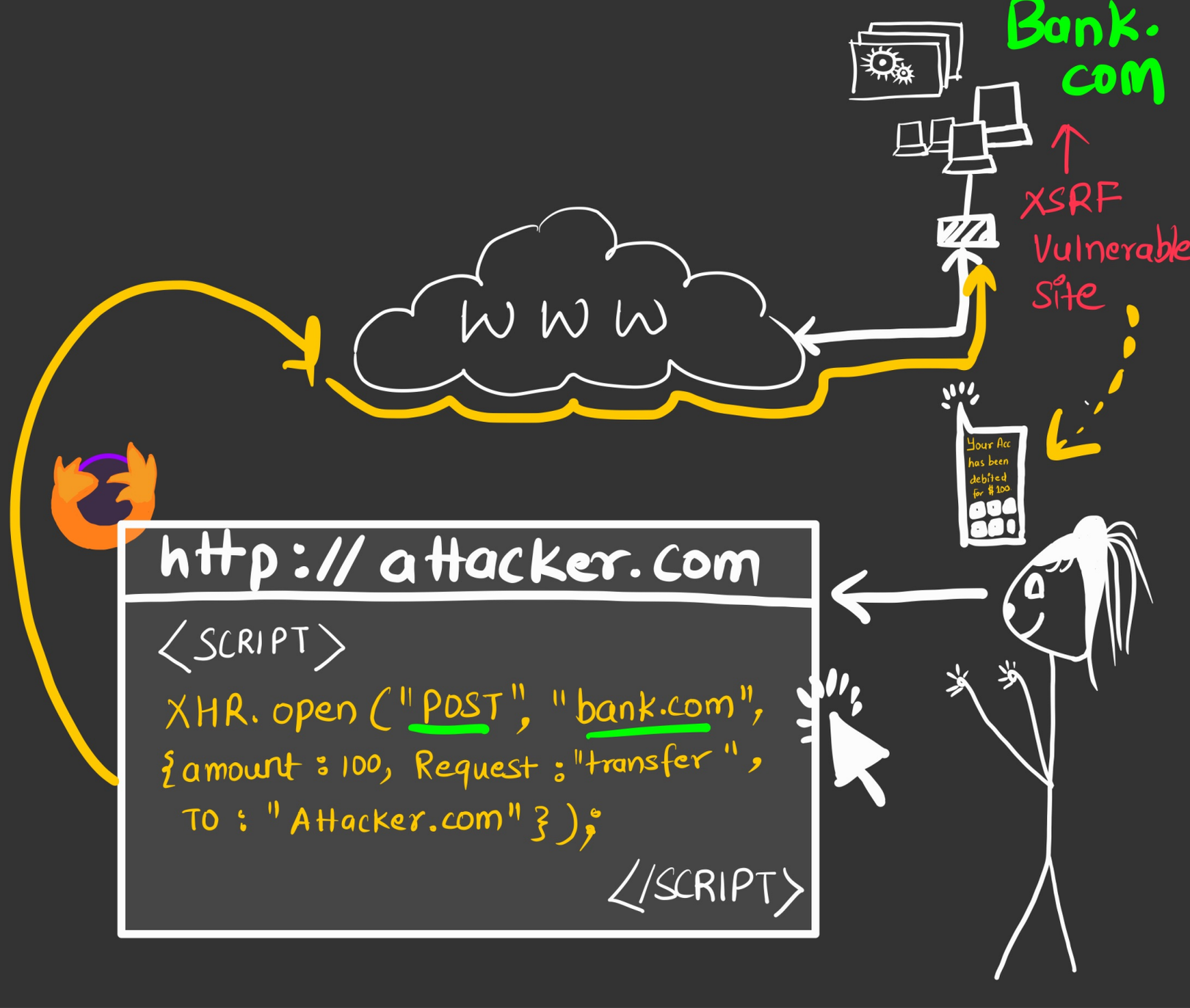
Security Zine #4

CSRF

Cross Site Request Forgery

OR

XSRF



What is this ?

Hey !!

I am Rohit
Your Mentor

And Today I want to tell u about

' CSRF '

* CSRF ✓
* How this Can be exploited.
* Consequences
* How to Fix it ?

sec_r0

'CSRF'

Money has been deducted from my Account

because you clicked the link you trusted

But what it has to do with my bank

Bank website was Vulnerable to CSRF

Now, what the Hell is that

sec_r0

CSRF

* Attack Exists when State change Requests can be automated.

→ POST, PUT DELETE

* AMOUNT TRANSFER REQUEST is State change. (From Server side)

* Bank.com had Amount transfer Request Endpoint (POST) which attacker automated through Javascript in Attacker.com

→ Because CSRF protection was not there.

sec_r0

CSRF Flow

You click on Random link on some website!

1

http://attacker.com

```
<SCRIPT>  CSRF PAYLOAD
XHR.open("POST", "bank.com",
{amount : 100, Request : "transfer",
 TO : "AHacker.com"});
</SCRIPT>
```

2

Cookies : Bank.com
For Current User

Bank.com

3

4

Bank see Request as : User Wants to perform tranfer to attack.com account

5

Cookie proves user identity = transfer amount

6

7

Amount transfer done, Send OK Response

8

9

200 OK

Response Blocked due to SOP

★ If you see, before Server Send back response to Script initiating transfer, Amount was alredy transferred.

★ Even if SOP block response Damage was already done.

sec_r0

* SOP will block the Response but damage was done at Server,

Even before Server responded.

* The Attack happenes because in Cross site Request, browser will pass Cookies 🍪. Which Identifies user.

* The Attack was exploitable as Server was not checking if the Request was automated from script.

sec_r0

CSRF

Consequences

Only Constraint is User should be logged in to CSRF vulnerable Site.

* Attacker Can make Request on behalf of user

* GET Request makes no sense here, as Response of Such request will be blocked by browsers.

→ due to SOP * Refer SOPZine

* No data theft possible.

sec_r0

Fixing CSRF

CSRF TOKENS

* Add a token to each form.
* Make Sure each request to Server contains the same token & validate it.
* Token Can be stateful or Stateless
* Include the token in form as HIDDEN Field.

SAME SITE COOKIE Attribute

* If cookies are set with SAME-SITE attribute
* Browser will prevent sending 🍪
* If cookies are not shared in Cross Site Request, CSRF is prevented.

Source Verification through Origin/ Refer Headers

* These headers are set by browser
* check the Starting point of Request and validate on server

CAPTCHA

* CAPTCHA prevents automated form submission
* there by preventing CSRF

Reauthentication

* Again validating password on each State change request.
* prevents automated form submission ⇒ prevents CSRF

Double Submit Cookie.

* In CSRF token based approach, tokens needs to be maintained by Server
* Adds extra load at Server
* Rather, add the token in Cookie & in hidden field.
* Since cookies are sent, check token with 🍪

sec_r0


I will make sure to disclose this to bank.com & remediation steps too!!
You will be Rewarded a bug bounty
Oh wow I can recover my money
Keep Smiling :)
sec_r0

Thanks for Reading

sec_r0

Join for more updates

Read more Zines

@

securityzines.com