Session hijacking

Background to this idea of continuous authentication.

Backing up to concepts of networks ,one thing we should all understand is we have 2 kinds of communication when working with internet.

One is connection based, other is packet based.

When I make a phone call ,I dial a number and I create a connection until we close the connection.

Connection based

Start a connection, assume all traffic between the 2 points that are created and then close the connection.

Packet based is like sending little pieces of information, one way and then the other way.

That’s like sending letters, they are data packets ,see them as packages that postman carries.

Have these 2 basic methods.

HTTP -protocol with web is a packet based protocol.

Feels like a connection, when I connect to bank do stuff and then drop connection. But not.

On a underlying level HTTP is me sending packets and getting something in return.

When we make communication into sessions, we are working hard to make packages look like sessions.

That’s when we get idea of cookies and their use for making things like sessions,

Appear out of quickly running postman.

Little illustrations here

Me sitting at desk ,wanting to communicate with bank.

Doing this over the web .Some kind of communication going on.

Since its via the web, it is packet based at a low level , even though I can imagine it’s a connection.

Problem comes if connection between me and my bank is not as strong as it should be.

What can happen in that case is that somebody else can come along and usurp the connection and take over here and continue communicating with the bank.

We do have a problem with packet based communication , is this communication between the 2 parties might be weak enough for somebody to break the connection, jump in and assume (usurp)

The role of bank ,take over the role.

This is session hijacking.

Even if things are connection based shouldn’t be possible for someone to cut connection as I’m speaking to my bank.

Session hijacking isn’t just a problem for packet based protocols like HTTP,

Cross-site scripting attack (XSS attack)

A kid whose name was Bobby Tables

Soon after changing name to Bobby Script, he started to see special messages on any new website he signs up to.

Same origin Policy (SOP) – Stops one website from reading or writing data to another.

Policy checks for 3 different things in the origin.

1)Protocol

2)Host

3)Port

Only if 3 are same for 2 different origins, browser allows cross-origin read and write.

If we have 2 websites – <http://pwnfunction.com> and other http://hackfunction.com

Browser checks if protocol same in both cases – it is same (http)

Checks for host(obviously they’re different)

One is pwnfunction , other is hackfunction so browser blocks cross origin read and write.

This is great feature, ensuring basic web security.

What if we have control of javascript of another website

Javascript has access to HTML via DOM API’s which are provided by browser. Can manipulate DOM and make it different ,like defacing page.

Can steal csrf tokens or read some cookies. Having access to javascript on another website in a different user’s context can be very problematic.

Can we inject js into another website? Yes

That is exactly XSS ,a javascript injection technique.

When we click on button ,we send out HTTP request. Name is sent out as GET parameter. After server processes information , it returns a response.

Browser sees whole of response data as HTML which is also seen by content type: text/html.

Browser can’t know without extra information, input we send is being reflected back in response.

What if input is <script> alert(1337)</script>

This becomes reflected XSS – input reflected back in response and gets identified as script block and gets executed.

Stored XSS – input is stored in a DB, shown to user by pulling it out from where it was stored.

Can affect everyone who views the page depended on your input.

It seems you're inquiring about Cross-Site Scripting (XSS) attacks. An XSS attack is a type of security vulnerability typically found in web applications.

It allows attackers to inject malicious scripts into webpages viewed by other users.

These scripts can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. XSS attacks can also be used to deface websites or redirect the user to malicious sites.

There are three main types of XSS attacks:

1. **Reflected XSS**: The malicious script comes from the current HTTP request. It's often delivered via a link which, when clicked by an unsuspecting user, sends a request to a vulnerable website which then includes the script in the response.
2. **Stored XSS**: The malicious script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information.
3. **DOM-based XSS**: This type of XSS attack occurs when the attack payload is executed as a result of modifying the DOM environment in the victim’s browser, typically in a client-side script.

It attacks the user's browser by forcing a server to transmit malicious code

A close-up of a screen

Description automatically generated

Session

Session – time u stay active using a website. Technique used by servers to keep track of who you are during a period of using a website. HTTP is stateless. While session is active every request in browser will be identified as coming from you. Otherwise every time u make a new request , you have to tell server hey it’s me again.

This is my personal ID. This is not a good way to surf the web.

No way server could know when you’re done with a website.

Session terminates either by you logging out or through an expiration mechanism.

We need sessions as HTTP is stateless. Not built to keep information about users. Sessions help associate requests to other requests. Without sessions if you are on facebook page and want to visit profile, you would have to login again.

When you login to website using username and password ,server may use this info along with other stuff to generate a unique id or a key and pass it back to web browser. This marks the start of session. Id is what further requests will use to authenticate you and talk to the server. This is the session id.

Session hijacking – attacker fools website into thinking they are you and then can do anything you can do on the site.

Hackers know how servers make use of session id’s to identify users so they will try to steal those ids or trick users to use ids they have control over.

Anyway an attacker can gain control of user’s session without their knowledge.

Browsers keep track of session keys used to identify users.

Communication in internet is done using packets.

3 ways to steal session key

1)Guessing key

2)Tricking user to login with compromised key

3)Stealing key from client or server

Man in the middle attack