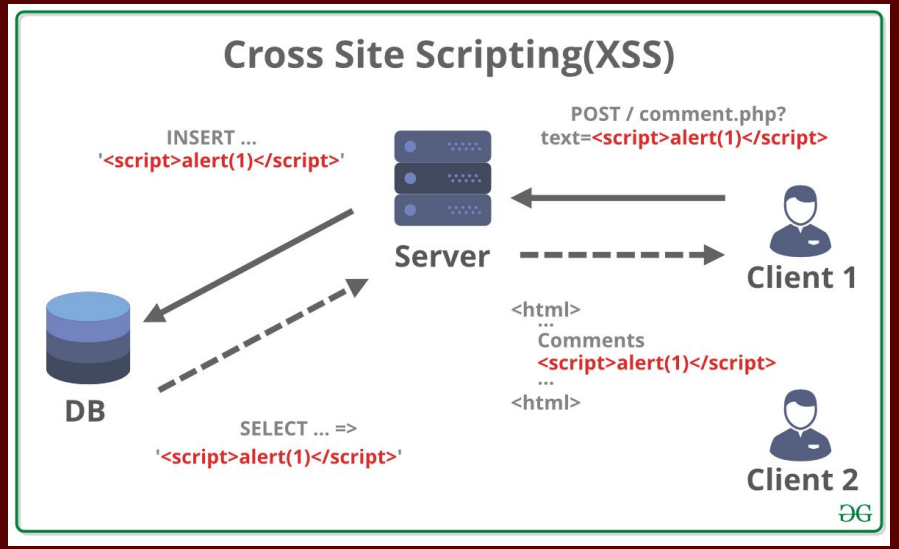
Web Application Vulnerabilities

Trey Atwood

October 27, 2025

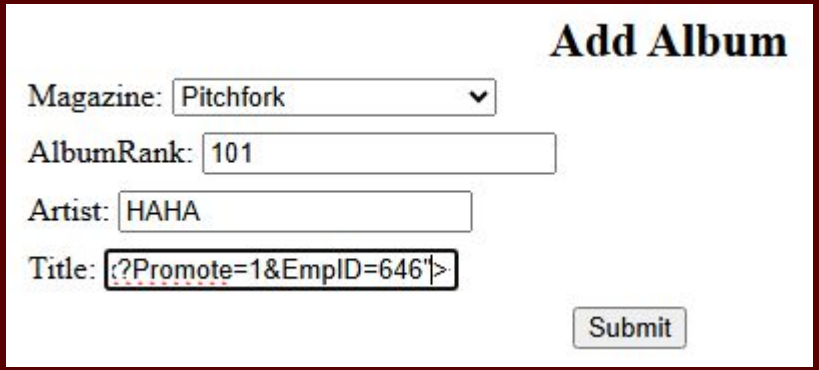
What is cross site scripting?

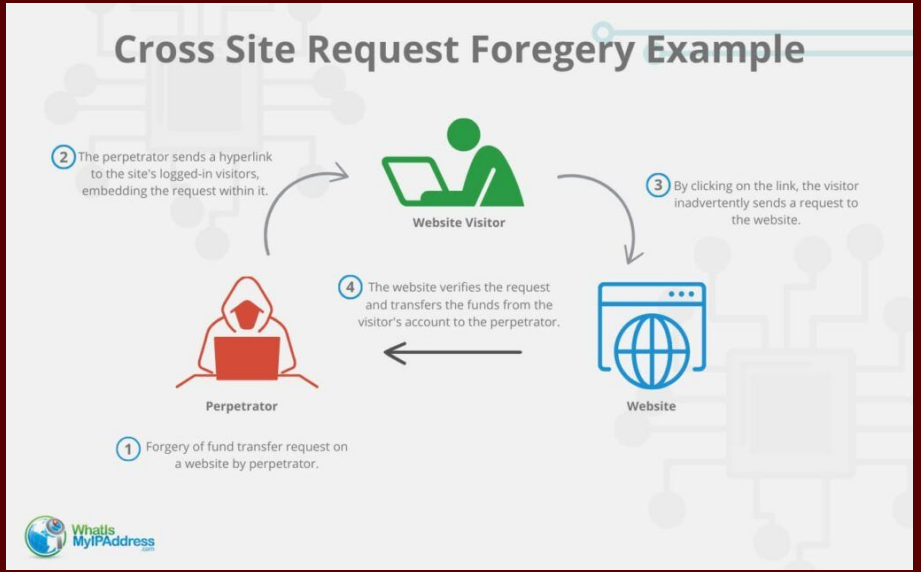
Cross Site Scripting or XSS, is a type of web vulnerability that allows a user to inject malicious scripts like JavaScript into a webpage that is viewed by others. A web application fails to safely sanitize user input, and the malicious user is then able to insert malicious code or scripts into their input. When another user visits the page whether it be the admin or just another user, the browser executes the scripts as if it came from a trusted source. The dangers of this are large, especially if you are able to get the Admins cookies, and you can steal their session tokens and login as the admin. This also comes as a threat to other users, as you can redirect them to malicious websites without them realizing what had happened.



What is a cross site request forgery?

Cross site request forgery or CSRF, is a type of web attack where a malicious website triggers a logged-in users browser to make unauthorized actions on another site. Say you're the admin of a web server, and someone unknowingly to you uploaded a script that makes you promote another employee in the system to have higher permissions than they should. Unknowingly you have just promoted someone in the company to have permissions they shouldn't and then can take control of systems if they act maliciously.

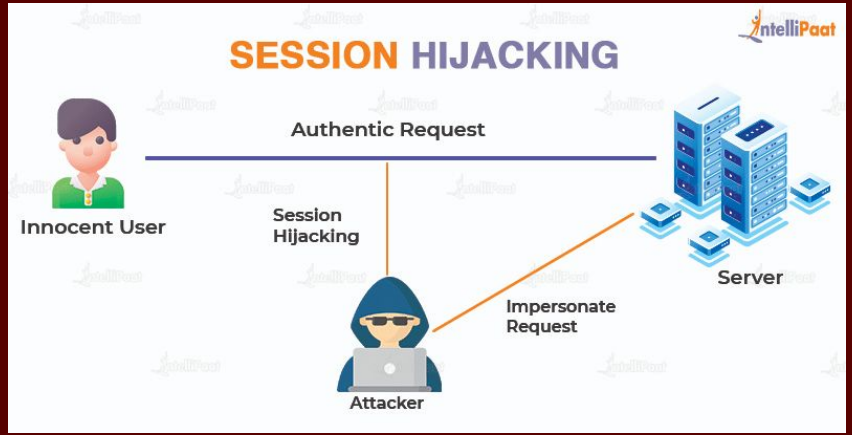




What does session hijacking mean?

Session hijacking is a cyberattack where the attacker takes over a legitimate users active web session. Stealing a users cookies, and then copying them into your own browser to browse as the legitimate user yourself. Each time you send a request your browser includes the session ID so the site knows that it's still you. If you're able to get these session IDs you can browse as someone you're not.

```
- ::1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+
 - ::1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)
- 80 - 10.1.0.1 Mozilla/5.0+(Windows+NT+10.0;+Wi
- 80 - 10.1.0.1 Mozilla/5.0+(Windows+NT+10.0;+Wi
LoginInfo=SessionID=pugwrvsqnljftv4ol1urnnmu 80
```



What is a unrestricted file upload?

This is a serious web application vulnerability that occurs when a website allows users to upload any file they'd like without validating them. Attackers are able to upload files that contain scripts that then can be executed on the server. This often leads to a full server compromise really damaging the webpage.

Here are the contents of each file
addalbum.aspx

```
<%@ Page validateRequest="false" %>
<%@ Import Namespace="System.Data.Odbc" %>
<%

Dim connString = "DSN=MariaDB;DATABASE=BestAlbums; User Id=root; Password=CIS@Room2015"
Dim conn = New OdbcConnection(connString)
conn.Open()
Dim SQL, cmd
Dim ValidForm=1
Dim Magazine, AlbumRank, Artist, Title
Magazine = request.Unvalidated().QueryString("Magazine")
AlbumRank = request.Unvalidated().QueryString("AlbumRank")
Artist = request.Unvalidated().QueryString("Artist")
Title = request.Unvalidated().QueryString("Title")
```

Let's start this exercise by downloading some necessary files from ASULearn. Onto our host machine where we are hosting the web server, we download the files "web.config", "login.aspx", and "upload.aspx"

Once we add those to the server, lets test to make sure that it all works

Magazine: Dazed

AlbumRank: 101

Artist: Trey

Title: Trey's So Cool

Submit

You are viewing the list for: Dazed

1	LANA DEL REY	DID YOU KNOW THAT THERE'S A TUNNEL UNDER OCEAN BLVD
2	KELELA	RAVEN
3	YEULE	SOFT SCARS
4	YAEJI	WITH A HAMMER
5	BOYGENIUS	THE RECORD
6	CAROLINE POLACHEK	DESIRE, I WANT TO TURN INTO YOU
7	SZA	SOS
8	TROYE SIVAN	SOMETHING TO GIVE EACH OTHER
9	MITSKI	THE LAND IS INHOSPITABLE AND SO ARE WE
10	AMAARAE	FOUNTAIN BABY
11	SPACE AFRIKA, RAINY MILLER	A GRISAILLE WEDDING
12	OLIVIA RODRIGO	GUTS
13	NONAME	SUNDIAL
14	JIM LEGXACY	HOMELESS N*GGA POP MUSIC
15	DJ GIGOLA	FLUID MEDITATIONS
16	STRANGE RANGER	PURE MUSIC
17	SUFJAN STEVENS	JAVELIN
18	100 GECS	10,000 GECS
19	CASISDEAD	FAMOUS LAST WORDS
20	OVERMONO	GOOD LIES
101	Trey	Treys So Cool

We are successfully able to add a new entry into the "Dazed" list. Confirming that everything is working as it should.

Last thing to make sure works is the upload.aspx file.

Upload Image

Choose File | No file chosen

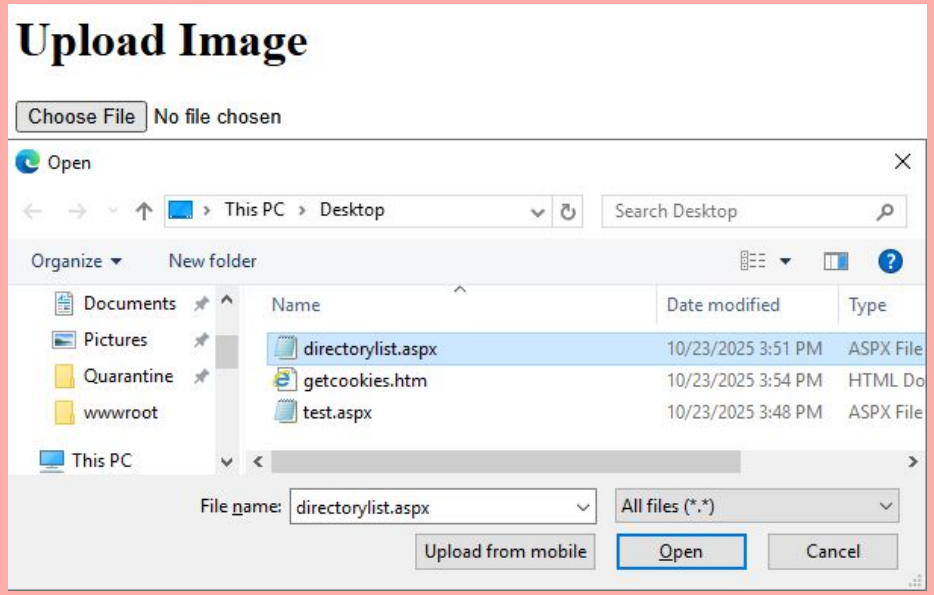
I uploaded a .jpg file of a grilled cheese sandwich, and it successfully was added onto the entry in the list!

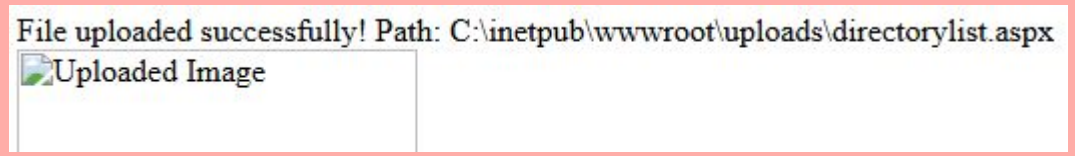
Everything is good!

Treys So Cool

Let's now switch over to our Azure machine and see if we can hack this web server.

To do this, we will download a couple more files from ASULearn but this time download it onto our Azure machine. The files "directorylist.aspx" and "getcookies.aspx" are what we'll download.





We naviage to "192.168.1.2/upload.aspx", and choose the file "directorylist.aspx". Once we upload the file we see File uploaded successfully! But then it reveals the file path of where the file is stored, it just gave us the root directory path.

Let's investigate this further.

Now that we have the file downloaded into the root folder of the directory, let's see what information we can get out of it. On the same page, we see contents of /uploads, followed by the file we just uploaded click onto the file and see what it displays.

Here are the contents of each file
addalbum.aspx

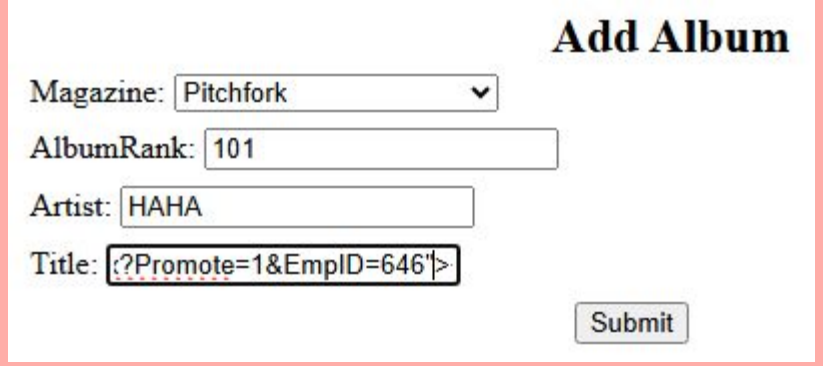
```
<%@ Page validateRequest="false" %>
<%@ Import Namespace="System.Data.Odbc" %>
<%

Dim connString = "DSN=MariaDB;DATABASE=BestAlbums; User Id=root; Password=CIS@Room2015"
Dim conn = New OdbcConnection(connString)
conn.Open()
Dim SQL, cmd
Dim ValidForm=1
Dim Magazine, AlbumRank, Artist, Title
Magazine = request.Unvalidated().QueryString("Magazine")
AlbumRank = request.Unvalidated().QueryString("AlbumRank")
Artist = request.Unvalidated().QueryString("Artist")
Title = request.Unvalidated().QueryString("Title")
```

In this file we exposed sensitive information regarding the web servers database. We got both the username and password to the database. That is real bad news for the host of the server, as we now can take control of their database and do whatever we please with the information.

For the next hack, we are going to see if we are able to force the admin to change my friend who works in the company to an admin.

In the addalbum page, we will add a new submission but hide the contents on the list. Using the command “<iframe Style="position: absolute; width:0; height:0; border:0;" src="hello.htm"></iframe>” we put it in the title box, but replace the hello.htm with “login.aspx?Promote=1&EmpID=646”. Emp=646 is my friends ID Glayds.



49	Sweeping Promises	Good Living Is Coming for You
50	André 3000	New Blue Sun
101	YLOLO	trey
101	HAHA	

This entry on the album looks normal. There isn’t anything that sticks out and says it’s something dangerous. In order for this to promote my friend Glayds, we need the current admin of the web server to access this page. Now it’s a waiting game.

Now as the web server host I'm doing my rounds on the website to make sure that everything is looking and working fine.

40	Ryuichi Sakamoto	12
41	Sexyy Red	Hood Hottest Princess
42	Youth Lagoon	Heaven Is a Junkyard
43	Kali Uchis	Red Moon in Venus
44	Jess Williamson	Time Ain't Accidental
45	Tomb Mold	The Enduring Spirit
46	Blue Lake	Sun Arcs
47	Parannoul	After the Magic
48	Purelink	Signs
49	Sweeping Promises	Good Living Is Coming for You
50	André 3000	New Blue Sun
101	YLOLO	trey
101	HAHA	

Reviewing the Pitchfork album list nothing immediately sticks out as wrong. No warnings or popups to say that you've just promoted an employee to have admin permissions.

635	Sol Mcconville	9971 Lost Dale	4348924922	4346350584	bartak@mac.com	
641	Danelle Sweetland	9099 Honey Timber Lane	7031655397	4347450757	bryanw@sbcglobal.net	
646	Glayds Guerrant	7623 Quaking Butterfly Forest	2767435406	8041932244	eabrown@icloud.com	1
653	Bennett Sanchez	3613 Little Crest	5404486741	7576154409	richard@live.com	
663	Neomi Yerkes	1881 Merry Grove	5719090217	8047135868	mpiotr@verizon.net	

But looking here on the login page, you can see my friend Glayds has the "1" at the end of his column. Meaning that he is now an administrator for the web server. Without the real admin knowing that this happened we sneakily promoted my friend.

Now acting as the web server admin, how does this vulnerability take place?

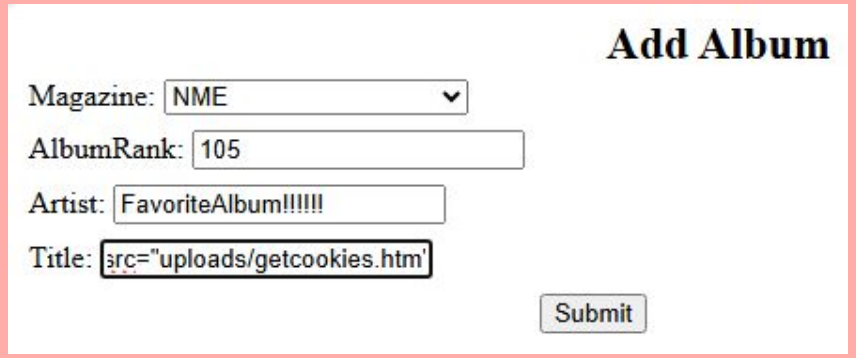
Well looking into the login.aspx file, lines 62 - 67 are where our problems lie.

```
If reader.Read()
    if request("Promote")="1" then
        dim UpdateEmpID = replace(request("EmpID"),"'","''")
        SQL = "UPDATE Employee SET Administrator = 1 WHERE EmployeeID = " & UpdateEmpID
        cmd = New OdbcCommand(SQL, conn)
        cmd.ExecuteNonQuery
    end if
```

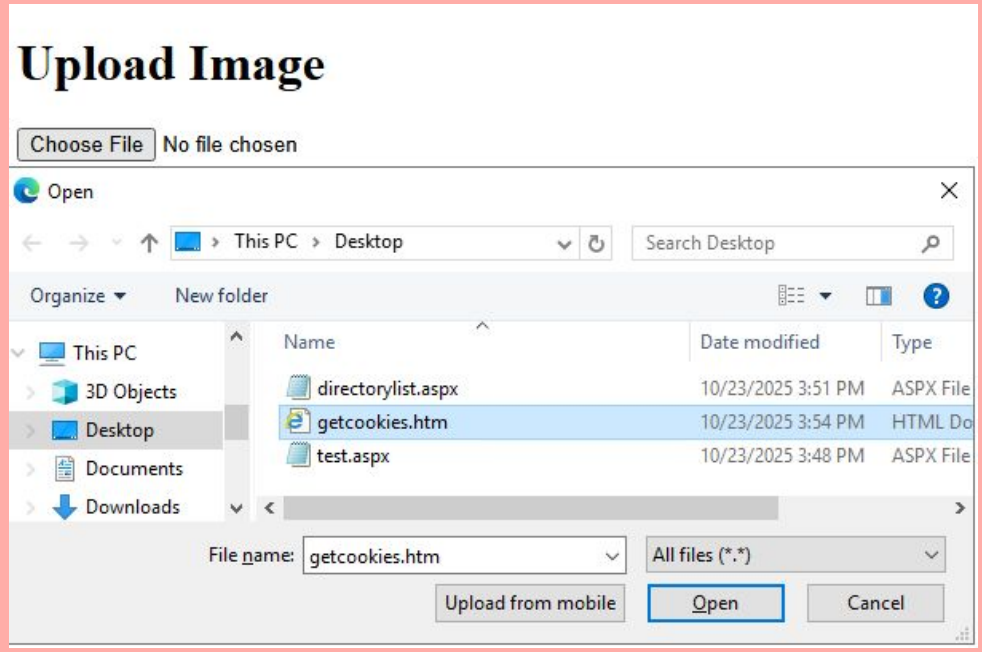
Using VS Code to inspect the file we see this is the problem. This is wrong because the line “UpadeEmpID = ... request(“EmpID)” is concatenated into the SQL string. This unsafely builds a SQL command based off of user input. This allows a hacker to do what we did and embed an iframe into the webpage and unknowingly update a user to Admin.

To fix this we need to change the code so it isn’t a “request” but instead a “post” method so it doesn’t allow just anyone to embed a hidden iframe into the web server and update employee information.

Next, we are going to try to steal the Admins cookies. In order to do this we need to download another file called “logit.aspx”. We need to set this up on our Azure machines web server, so download the file into the wwwroot folder on the Azure Machine.



We’ve successfully uploaded the file onto the server, but we aren’t done yet. We now need the Admin to access the file so we can steal her cookies. To do this we use the same iframe command from earlier except we replace the “hello.htm” with “uploads/getcookies.htm” the file path to the file we just uploaded.

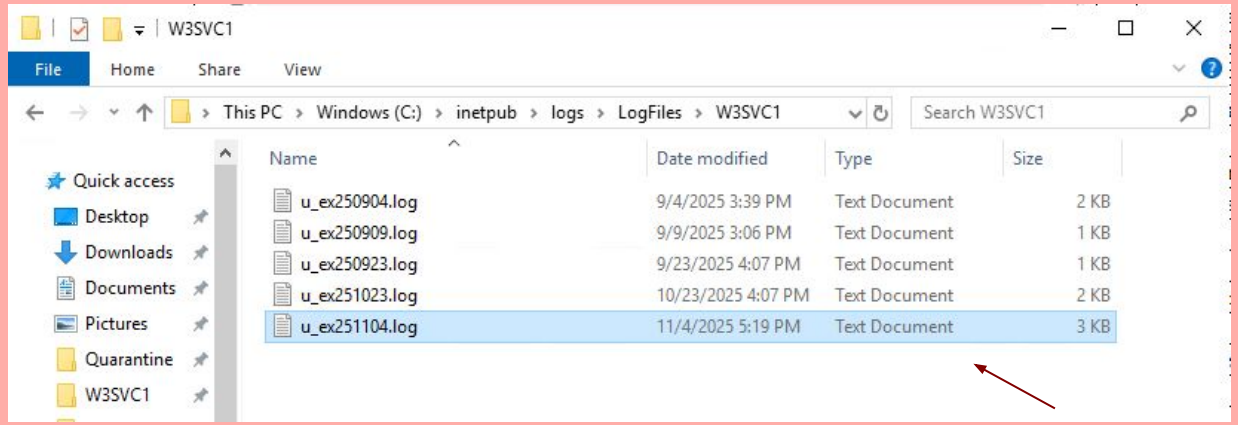


On the Upload Image tab, we are going to upload the file we downloaded earlier called “getcookies.htm”.

Here we can see that we have added the new entry into the list NME.

105	FavoriteAlbum!!!!!!	WOWZERS	

Now we just need the Admin to access this NME list and we are golden.



We just got an updated log file in our web server on our Azure machine, let's inspect it to see if the Admin fell into our trap!

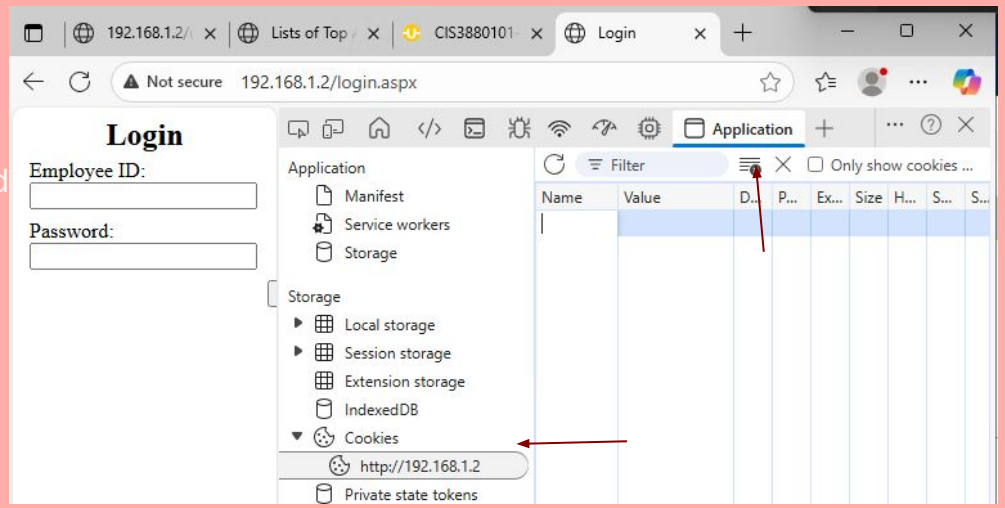
Let's inspect the new log file!

```
2025-11-04 17:11:43 ::1 GET /logit.aspx - 80 - ::1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+
2025-11-04 17:11:43 ::1 GET /favicon.ico - 80 - ::1 Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)
2025-11-04 17:13:03 10.1.0.1 GET /logit.aspx - 80 - 10.1.0.1 Mozilla/5.0+(Windows+NT+10.0;+Wi
2025-11-04 17:17:12 10.1.0.1 GET /logit.aspx - 80 - 10.1.0.1 Mozilla/5.0+(Windows+NT+10.0;+Wi
2025-11-04 17:18:36 10.1.0.1 GET /logit.aspx LoginInfo=SessionID=pugwrvsqnljftv4ol1urnnmu 80
```

Look at that! We got the admins cookies! Lets see what we can do with this information.

On our Azure machine go to "192.168.1.2/login.aspx" and do ctrl + shift + i to access developer tools, navegate to the Application toolbar and we will add a new cookie.

We want to add our own saved cookie. To do this under the name value we enter Logininfo. Then for the Value entry we want to enter the SessionID= … .



Once we finish entering the Admins cookie information onto our own browser refresh the page.

We are now the admin, Alyssa Sprinkle. Now we can see all the user information, phone numbers, addresses, emails anything we want we have it. Just for fun we can promote everyone to have Admin permissions.

This exercise shows the danger of having a poorly put together web server, allowing files that shouldn't be allowed to be added to the server allows a hacker to basically get or do anything that they want. Making the web server strictly only accept one file type is a basic addition but an important one at that. Not allowing hackers to upload any file type they want will prevent things like this from happening.