pentest report example

- Heading page
- Summary
- Attack narrative
- Findings
  - BAC on viewusers.php
    - Steps to reproduce
    - Expected result
    - Actual result
  - BAC on new.php
    - Steps to reproduce
    - Expected result
    - Actual result
    - Score
  - IDOR on the editing of posts
    - Steps to reproduce
    - Expected result
    - Actual result
    - Score
  - IDOR on the deletion of posts
    - Steps to reproduce
    - Expected result
    - Actual result
    - Score
  - No HTTPOnly flag defined on session cookie
  - Missing CSP header
  - Reflected XSS on search page leads to account takeover
    - Steps to reproduce
    - Expected result
    - Actual result
    - Score
  - No CORS headers defined
  - Self XSS chained into full XSS with CSRF
    - Steps to reproduce
    - Creation of PoC
    - Expected result
    - Actual result
    - Score
  - There is no rate limiting on the login/register pages. Same goes for anything.
    - Score

Conclusion
- Recommendations
- Cleanup/retention steps
- GO/NOGO

Extra examples

Heading page

Cheesebook

15/12/2021

Testers:

The XSS rat - info@thexssrat.com

BE0508.999.580

0476876632

Summary

We were able to identify several high severity vulnerabilities, some of which can easily lead to account takeover. Security seems to be lacking in a few key areas and it's recommended to go through each item and see if remidiation is required.

Attack narrative

In this assignment, we are testing in accordance to the OWASP top 10 web vulnerabilities. We are looking for each item on the list and following the OWASP web testing guide:

https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion

Findings

BAC on viewusers.php

There seems to be a BAC since everyone can open up viewUsers.php and see the contents of the user list.

Steps to reproduce

- Make sure to be logged out.
- Navigate to https://hackxpert.com/pentest/viewUsers.php

Expected result

- Only admins can view a list of users

Actual result

- One does not have to be logged in to view the list of users

BAC on new.php

There seems to be a BAC since everyone can open up viewUsers.php and see the contents of the user list.

Steps to reproduce

- Make sure to be logged out.
- Navigate to https://hackxpert.com/pentest/new.php

Expected result

- Only logged in users can make a post

Actual result

- One does not have to be logged in to make a post

Score

10/10 PII is leaked

IDOR on the editing of posts

By directly going to the page to edit a post and entering any identifier, we can edit posts that do not belong to us.

Steps to reproduce

- Log into the application
- Go to https://hackxpert.com/pentest/edit.php?id=108
- Fill in the ID of a post of another user

Expected result

We get an error message

Actual result

We can see the edit page and actually edit a post

Score

7/10

IDOR on the deletion of posts

By directly going to the page to edit a post and entering any identifier, we can edit posts that do not belong to us.

Steps to reproduce

- Log into the application
- Go to https://hackxpert.com/pentest/delete.php?id=108
- Fill in the ID of a post of another user

Expected result

We get an error message

Actual result

We can delete a post that is not ours

Score

8/10 This can be very annoying to users driving them away

No HTTPOnly flag defined on session cookie

This always needs to be defined if possible, see the next issue.

Missing CSP header

This always needs to be defined if possible, see the next issue.

Reflected XSS on search page leads to account takeover

When searching for a XSS attack vector, no posts will be found but a reflected XSS will be triggered due to the reflection of the search parameter.

Steps to reproduce

- Search "<img src=x onmouseover=alert()>"

Expected result

We get a message no results have been found but no reflection is done.

Actual result

An XSS attack vector will trigger. You can steal the cookies due to not having the HTTPOnly flag enabled. **This can lead to full account takeover. This is made even worse because there is no CSP header which makes it very easy to smuggle out data.**

Score

10/10 account takeover of anyone

No CORS headers defined

These should always be defined when possible

Self XSS chained into full XSS with CSRF

In the user panel, if you make a post with a XSS attack vector title, it will pop up but since this is only in your own account panel, this is self XSS.

We can however upgrade this because there is no CSRF token on the creation of a post. This means we can make anyone create a post with a XSS attack vector via CSRF and when they visit their account panel, we can again steal their session cookies.

> https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b6534a7b-3ec9-4dbd-a8d0-e4a8d4390646/csrfPoc_(9).html

CSRF PoC with XSS provided.

Steps to reproduce

- Log in
- Click on the PoC
- Go to the user panel
- You now have a XSS attack pop up

Creation of PoC

XSS Vector is encoded with https://www.urlencoder.org/

XSS PoC is generated with https://security.love/CSRF-PoC-Genorator/

Settings:

- Data
  title=%3Cimg%20src%3Dx%20onerror%3Dalert%28%29%3E&description=test&cat=3&submit=Post
- URI: https://hackxpert.com/cheesebook/new.php
- Method: POST
- Encode: application/x-www-url-encoded

Expected result

The HTML is filtered properly and the user sees their post. It's not possible to insert a post via CSRF due to a token being present that is needed to be verified. This token is not there.

Actual result

An XSS attack vector will trigger. You can steal the cookies due to not having the HTTPOnly flag enabled. **This can lead to full account takeover. This is made even worse because there is no CSP header which makes it very easy to smuggle out data.**

Score

10/10 Account takeover

There is no rate limiting on the login/register pages. Same goes for anything.

This allows attackers to easily perform attacks such as credentials stuffing in which they will attempt to log in with a list of usernames and passwords they gained from hacking other applications. Attackers can also spam by sending a lot of requests to create new posts. In combination with new.php not being secure and the attacker not even having to log in, this issue is severe.

Score

8/10

Conclusion

The application contains a lot of weaknesses of high severity still and being able to chain them can get use some very impactful results. This application is not production ready.

Recommendations

- Add authorization and authentication checks to viewUsers.php and new.php
- Add authorization checks to edit.php and delete.php
- Do not reflect user values, if you must, filter them properly

- Add CSRF tokens to every form that is put behind a login screen and verify it
- Add CSP and CORS headers
- Add rate limiting
- ...

Cleanup/retention steps

We deleted all posts that were used in the pentest and not required for the report.

GO/NOGO

No go based on previous advice

Extra examples

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5fd66925-1247-4d10-9098-5c7c190cf93c/Pentesting.docx