LUNDS UNIVERSITET
Lunds Tekniska Högskola

WEBSHOP UNDER ATTACK

October 12, 2016

Tim Dolck dat11tdo@student.lu.se
Alexander Olsson dat12aol@student.lu.se
Anton Andersson vov13aan@student.lu.se
Emil Hasslöf dic14eh1@student.lu.se

Contents

1 ARCHITECTURAL OVERVIEW

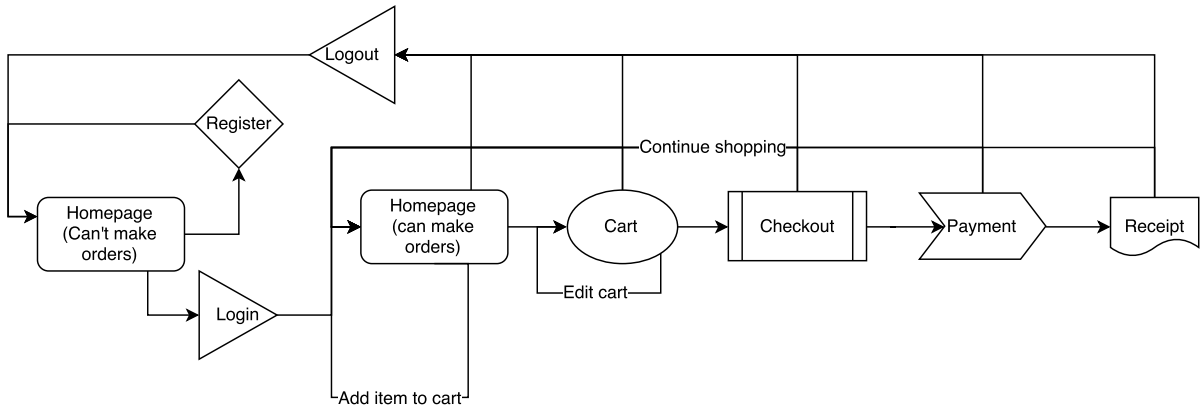

Figure 1: Architectural overview of the web page.

The workflow of the webpage, as seen in Figure 1, is relatively simple. The customer has to register an account to be able to order products. Even though a visitor doesn't have an account it's still possible to view and add products to the cart. To be able to checkout and pay, the user has to be logged in. If the user decides to add products to the cart and then log in, the items will still be saved in the cart after the log in. When a user is logged in, the header displays a text "*Logged in as: username*" in the top right corner.

1.1 Database design

The webshop relies on a MySQL database consisting of three tables: *Users*, *Products* and *Orders*. These are the most vital things to keep track of in the webshop.

1.1.1 Users

Updated when a visitor registers and is used to control that the user credentials are correct when logging in.

1.1.2 Products

Used when displaying all the products on the homepage as well as in the cart. When editing the *Products*-table in the database, the webshop will automatically be updated and make the changes available on the web page.

1.1.3 Orders

Used to log orders that have been made. It contains informations about which user ordered which products and when it happened.

2 SECURITY CONSIDERATIONS

2.1 Secure transmission

A self signed certificate is used to set up a TLS tunnel to encrypt all traffic. The 2048 bit keys are generated using the RSA algorithm. This makes traffic sniffing and man-in-the-middle attacks very difficult for potential attackers.

2.2 Authentication and password handling

Chosen passwords are required to be at least 8 characters long and must consist of both upper- and lower-case letters as well as at least one digit. This makes all kinds of brute-force attacks harder as the easiest passwords are ruled out. It is not, however, sufficient protection by itself. To protect against online brute-force one user is limited to 10 login tries in a short timespan. If too many invalid login-tries is detected, the user is blocked to do more login tries for a period of 10 minutes.

The passwords are stored in the database. If a someone manages to steal them, they need to be hard to crack. Therefore, the passwords are hashed with a randomly genererated salt and with a strong and empirically tested hash-function. The hashing function chosen in this case is BCRYPT[1]. This will prevent the attacker from cracking the passwords with time-memory tradeoff (TMTO) or Rainbow tables.

2.3 Session management

The session key token is stored in a cookie, and is not visible in the URL. This makes session hijacking harder and does not allow a malicious user to distribute a link containing a session key to other users.

If a user presses the "logout" button, the session is ended. Next time that user logs in, a new session key is generated.

When a user signs in the session id is regenerated. This helps to prevent session fixation attacks.

2.4 Server configuration

One of the most important considerations when working with web-security is to have the server properly configured. We have focused on two parts off the server configuration: apache and php.

2.4.1 Apache configuration

Apache configuration is controlled through the httpd.conf and other linked .conf files.[5] Specifically we have made the following changes:

- Disable directory listing
- Disable some unused modules
- Rewrite all request to https
- Enable SSL
- Set SSLrandom seed to good random quality
- Set a strong SSL cipher suite

2.4.2 Php configuration

The php configuration is mainly to prevent php to leak unecessary error messages and to set a strong hashing function for the session variable. This makes it harder to guess session id.

- display_errors ⟶ off
- display_startup_errors ⟶ off
- track_errors ⟶ off
- html_errors ⟶ off
- file_uploads ⟶ off
- allow_url_fopen ⟶ off

- disable smtp
- session.bug_compat_42 ⟶ off
- session.bug_compat_warn ⟶ off
- session.hash_function ⟶ sha256

2.5 Specific attacks

2.5.1 SQL-injections

Prepared statements are used whenever the user enters information that will be stored in the database, for example username and address etc. This efficently blocks sql-injections [4]

2.5.2 Cross-Site Request Forgery

The threat of Cross-Site Request Forgery is determined by a couple of factors. First off, the web browser that the client use affects the feasibility of the attack. Newer browsers have implemented the "Same-origin Policy" (SOP)[2] which prevents content hosted from one (malicious) origin to make requests to or access content from another (affected) origin. However, SOP allows cross-origin writes e.g. form submissions (POST/PUT/DELETE/...), and cross-origin embedding e.g. with the image tag (GET). Therefore it is still possible to perform Cross-Site Request Forgery attacks unless precautions have been taken by site owners of the affected site. Protection from CSRF attacks through form submission have been accomplished with the use of CSRF tokens[3]. A token is generated server-side to the session of a user every time a page that requires the token is requested. The token is included with all forms that are sent in response and then checked when a form is submitted. The malicious site would not be able to provide this token and the form submission would not be accepted by the server.

2.5.3 Cross site scripting

There is currently only one case where the website will show text submitted by regular users. This happens where the user registers their account. The username will then be shown in the page header when logged in. When the user checks out their purchase, a receipt is shown with all the users information, e.g. name, address, ZIP etc. A successful XSS attack was accomplished by registering a user with the forename `<script>alert("PppP")</script>`.

When this user made a purchase and the receipt was displayed, an alert box appeared. The username is limited to 30 characters, which severely limits what one can accomplish with the script, but it is a successful attack nonetheless.

To prevent this, regular expressions are used to validate the user input. No special characters are allowed anywhere except for the email field. This instead makes use of a built in PHP function for email validation, `filter_var($email, FILTER_VALIDATE_EMAIL);`.

2.5.4 DOS-Attack

Denial-of-Service comes in many variations, e.g. Distributed-Denial-of-Service, but the basic idea is to make a specific service unavailable. A DOS-Attack is carried out by flooding the targeted server/machine with unnecessary requests to overload the server, so some or all legitimate requests are prevented from being fulfilled.

Some symptoms of a DOS-attack is that the service starts being unusually slow or a specific web page isn't available etc.

Protecting a web server from the different DOS-attacks is tricky. The single-origin DOS-attack isn't that hard to prevent, e.g. by blocking the attackers IP. However, it's a lot harder to prevent for example a DDOS-attack, one of the reasons being that it's hard separating an attacker from a normal user.

REFERENCES

[1] PHP documentation
http://php.net/manual/en/function.password-hash.php
Accessed: 09-10-2016

[2] Same-origin policy (Mozilla developer network)
http://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
Accessed: 02-10-2016

[3] Synchronizer (CSRF) Tokens
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet#Synchronizer_.28CSRF.29_Tokens
Accessed: 02-10-2016

[4] W3Schools - Prepared Statements
http://www.w3schools.com/php/php_mysql_prepared_statements.asp
Accessed: 09-10-2016

[5] Apache - Configuration Files
http://httpd.apache.org/docs/current/configuring.html
Accessed: 09-10-2016

3 PEER REVIEWS

3.1 Review by group 14

Architectual overview

A short introduction to the purpose of the project would be fitting, that you are building a webshop and implementing attacks to compromise it.

It's a little bit difficult to understand the diagram you have drawn. An improvement would be to make the arrow from "Homepage (can make orders)", the one which shows that you can loop around when adding more items to the cart. The text below the figure is concise, easy to read and explains the procedure in a good manner. However, there are some sentences that may be improved such as "The first time visiting the web shop the user has to...". An formulation could be for example "The user has to register to be able to shop/set an order". Also, the sentence "To be able to checkout and pay, they have to log in" would sound less informal as "To be able to proceed to checkout and pay, the user has to log in".

It is good that you mention how the database is structured and used. However, you might want to make the first sentence shorter or use a colon. "The webshop relies on a MySQL database consisting of three tables:...". The word "bought" is misspelled in last paragraph.

Security considerations

The section on Secure transmission is well structured and informative. You mention how you protect yourselves against time-memory tradeoff and rainbow table attacks. You shouldn't use abbreviations for words the first time you use them, an example is when you write TMTO. It is better if you write "time-memory tradeoff (TMTO)" the first time it is mentioned. Also, great that you mention the vulnerability towards online brute-force attacks and addressing the problem with password quality.

The content in the CSRF section is well written. You might also add that you can protect your website from CSRF attacks by not using `$_REQUEST` and using `$_POST` instead. This is because `$_REQUEST` might use both `$_GET` and `$_POST`. `$_GET` requests send info in the URL, and CSRF attack also send arguments through the URL. However, the section feels hastily written since there are some grammatical errors.

The XSS has issues similar to the CSRF section. "When the user checks out their purchase, a receipt is shown with all the users information..." would be more correct as "When the user checks out a purchase, a receipt is shown with the user's information...". Content is good and informative. The DOS-attack section has very good structure. "from" is spelled wrong in the first sentence in the last paragraph and you should avoid starting sentences with "but".

The report does not provide information on what considerations have been made regarding PHP and Apache configurations. There are a number ways to modify these in order to accommodate security requirements. Also, descriptions are a bit inconsistent in how much detail they go into. Some sections describe only how the webshop is protected from the attack while some sections describe how the attack itself works as well.

3.2 Review by group 15

We think your report is generally good and thorough, but we have found a number of details that we think you should look into.

You have used a good language and structure though the whole report.

We think there should be a figure that explains the structure of the page, including for example the database, and not only the one showing how the website works from the user's perspective. Right now the part about the design choices is completely unrelated to the picture. You have never referenced the figure in the text. The figure and text does though give a good overview of the usage of the homepage so we don't think there's any reason you should remove it, but rather complement it.

Some smaller details: There is a word wrap missing in section 2.4.1. The usage of colons in some section titles looks a bit odd and we suggest you remove them.

In section 2.4.2 you describe the websites as malicious.com och vulnebare.com. This seems a bit informal. It would fit the general style better if you instead provided a description of the websites (such as for example “the website the user intends to visit”).

We like that you have divided the security considerations into four sections to make it clear. Generally you have followed the instructions but we have noticed that you have left out

references, which should be included.

4 IMPROVEMENT SHEET

The following improvements have been made in accordance with the peer reviews and after several proof-readings:

- Reference Figure 1 in the Architectual Overview section.
- Rewrote the first paragraph of the Architectual Overview section to make it clearer and more concise.
- Revised section titles.
- Added explanations for some abbreviations the first time they are used.
- Added Serever Configuration section.
- Rewrote parts of the Cross-site request forgery section to make it less informal and more informative.
- Added references.
- Fixed the grammatical errors stated in the peer reviews and several more, too many to list.

LUNDS UNIVERSITET
Lunds Tekniska Högskola

EITF05 Web Security
Functionality Review Form
Autumn 2016

How to perform the functionality review

The group being reviewed (reviewee) is denoted 'Group E' in the sequel. The group that is performing the review (reviewer) is denoted 'Group R'. The review process is as follows.

1. Group E demonstrates that their SSL connection works.
2. Group E demonstrates that their password handling is secure against TMTO/Rainbow attacks.
3. Group E demonstrates that usage of PHP sessions is reasonable.
4. For each attack that has been implemented, do the following:
   a. Group E explains to Group R how the attack works in general.
   b. Group E demonstrates (use-case and code) how their website can be made susceptible to the attack.
   c. Group E demonstrates how to protect the website so that the attack no longer works.

Review report

Group R fills out the relevant information in this section.

Group R number: 14 Group E number: 13

Attacks implemented and reviewed:
☒ SQL Injection
☒ Cross-Site Scripting (XSS)
☒ Cross-Site Request Forgery (CSRF)
☐ Remote File Inclusion

Group R hereby confirms that the project implementation of Group E is of sufficient quality and complies with the project requirements.

Group R signatures:

Lund, 2016-

LUNDS UNIVERSITET
Lunds Tekniska Högskola

EITF05 Web Security
Contribution Statement Form
Autumn 2016

Individual project contributions

Group number: 13

name	contribution
Anton Andersson	certificates, cart, attacks etc
Emil Hasslöf	PHP coding + report
Alexander Olsson	Database, PHP coding, report etc
Tim Dolck	Configuration, coding and report

All group members have actively taken part in and contributed to the project in sufficient part.

Member signatures:

Anton Andersson

Tim Dolck

Emil Hasslöf

Alexander Olsson

Lund, 2016-