LUNDS UNIVERSITET
Lunds Tekniska Högskola

WEBSHOP UNDER ATTACK

October 4, 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

1 ARCHITECTURAL OVERVIEW

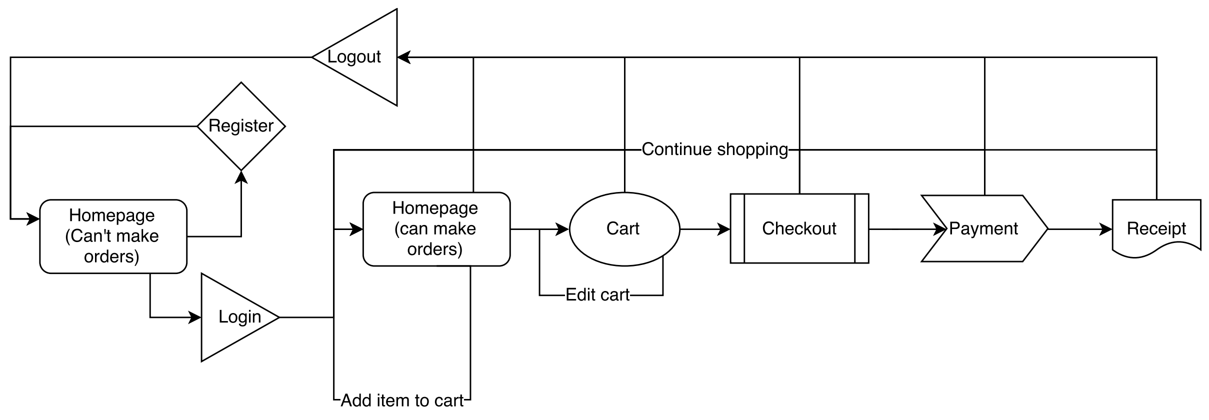

Figure 1: Architectural overview of the web page.

The first time visiting the webshop the user has to register an account to be able to shop. 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, they have to log 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.

It's clearly displayed when a user is logged in, displaying the message "*Logged in as: username*" in the top right corner.

1.1 Design choices

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.

The *Users*-table is updated when a visitor registers and is used to control that the user credentials are correct when logging in.

The *Products*-table is 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.

At the moment the *Orders*-table isn't used, but the plan is to implement a log of which user brought what product and when.

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

When a user is creating a new account, their credentials such as username and password are stored in the database. When created, the password is hashed with a random salt using the PHP function `password_hash` with `PASSWORD_BCRYPT` as its argument. This will prevent the users from getting their account hijacked by attacks like TMTO or Rainbow tables.

There is currently no limit for login attempts, which opens up a vulnerability for online brute-force attacks. Specially considering there is no check for password quality. This is a major security flaw that should be addressed. To fix this one could set the maximum login attempts to e.g. three, then the user has to reset their password or wait for 10minutes until they can try login again.

Currently there are no restrictions in place in regards to password quality. It is assumed that the user takes responsibility for choosing a sufficiently complex password. This is a question of balance between usability and security, and could of course be improved in future iterations.

2.3 Session management:

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

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.

2.4 Specific attacks:

2.4.1 SQL-injections

Prepared statements are used whenever the user enters information that will be stored in the database, for example username and address etc.
Users can also manually enter the amount of items to order, which will then be placed in the "orders" table in the database. This amount is casted to an INT which prevents SQL-injections this way.

2.4.2 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) which prevents content hosted from one origin (malicious.com) to make requests to another origin (vulnerable.com). However, SOP allows cross-origin writes e.g. form submissions and cross-origin embedding e.g. with the image tag. Therefore it is still possible to perform Cross Site Request Forgery (CSRF) attacks unless precautions have been taken by site owners of vulnerable.com. One way to prevent the threat of a CSRF attack through form submission would be to use CSRF tokens. These tokens are coupled server-side to the session of a user and then included with all forms that are sent to the user and checked when the form is submitted. Another origin (malicious.com) would not be able to provide this token and the form submission would not be accepted by the server.

2.4.3 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 currently 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 is 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);`.

2.4.4 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.

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