Project 3 Description

CS161 Computer Security, Fall 2008
Assigned : 11/19/2008
Due Date : 12/14/2008

1 Web Application Security

StockBank is a stock management web application, hosted at `http://sphere.cs.berkeley.edu:8080`, which allows registered users to post profiles, buy "stocks" and transfer them to each other. Each registered user starts with a balance of 10000 "dollars" to buy stocks with. This project has two parts :

- *Part 1*: Construct various attacks against StockBank website.
- *Part 2*: Fix the web application to prevent the attacks you found in part 1.

Although many real-world attackers do not have the source code for the web sites they are attacking, you are one of the privileged ones: you have access to the source code. You would not actually need to look at the site's source code until Part 2, but it's there if you get stuck. The code is available at: `/home/ff/cs161/proj3-fa08/proj3-code.tgz`

1.1 Part 1 : Attacks

You have to perform 4 kinds of attacks on StockBank.

- *A : Cookie Theft* [**5 points**].
  - Your solution is a URL starting with `http://sphere.cs.berkeley.edu:8080/users.php?`.
  - The grader will already be logged in to StockBank before loading your URL in the browser
  - Your goal is to steal the document cookie and email it to the grader, using an *email script* that we provide. The function of the email script is described later.
  - Except for the browser address bar (which can be different), the grader should see a page that looks exactly as it normally does when the grader visits `users.php`. No changes to the site appearance or extraneous text should be visible. Avoiding the red warning text is an important part of this attack. (It is ok if the page looks weird briefly before correcting itself.)
  - *Hint:* Here is an example attack as a starting point. Place this link in a HTML file, open it in a browser and click it : `<a href='http://sphere.cs.berkeley.edu:8080/users.php?user=%22%3E %3Cscript%3Ealert%28document.cookie%29%3B%3C/script%3E'> Example </a>`
- *B : Cross-site Request Forgery* [**5 points**]
  - Your solution is a short HTML document that the grader will open using the web browser.
  - The grader will already be logged in to StockBank before loading your page.
  - Transfer 10 dollars from the grader's account to the account of a user named "attacker". The browser should be redirected to `http://sphere.cs.berkeley.edu:8080` as soon as the transfer is complete (so fast the user might not notice).
  - The location bar of the browser should not contain `sphere.cs.berkeley.edu:8080` at any point.
- *C : Password Theft* [**5 points**]

  - Your solution is a short HTML document that the grader will open using the web browser.
  - The grader will **not** be logged in to StockBank before loading your page.
  - Upon loading your document, the browser should immediately be redirected to the StockBank web site at `http://sphere.cs.berkeley.edu:8080/`. The grader will then enter a username and password and press the "Log in" button.
  - When the "Log in" button is pressed, send the username and password (separated by a comma) using the email script to the grader.
  - The login form should appear perfectly normal to the user. No extraneous text (e.g. warnings) should be visible, and assuming the username and password are correct the login should proceed the same way it always does.
  - *Hint*: The site uses the PHP `htmlspecialchars()` function to sanitize the reflected username, but something is not quite right.

- *D : Profile Worm* [**5 points**]
  - Your solution is a profile that, when viewed, transfers 1 dollar from the current user to a user called "attacker" (that's an actual username) and replaces the profile of the current user with itself.
  - Your malicious profile may include a witty message to the grader (optional, but it helps us see that it replicated).
  - To grade your attack, we will cut and paste the submitted profile file into the profile of the "attacker" user and view that profile using the grader's account. We will then view the copied profile with more accounts, checking for the transfer and replication.
  - The transfer and replication should be reasonably fast (under 15 seconds). During that time, the grader will not click anywhere.
  - During the transfer and replication process, the browser's location bar should remain showing `http://sphere.cs.berkeley.edu:8080/users.php?user=username`, where `username` is the user whose profile is being viewed. The visitor should not see any extra graphical user interface elements (e.g. frames), and the user whose profile is being viewed should appear to have 10 dollars.
  - You will not be graded on the corner case where the user viewing the profile has no dollars to send.
  - *Hint:* The site allows a sanitized subset of HTML in profiles, but you can get around it. This MySpace vulnerability may provide some inspiration for this attack. You can find an online document about this at : `http://www.securityfocus.com/archive/82/430263/30/120/threaded`.

Email script. For Attacks A and C, you will need a server-side script to automatically email information captured by your client-side JavaScript code to the reader for grading. We have provided this script for you at `http://sphere.cs.berkeley.edu:8080/sendmail.php`. It explains how you can use it to automatically send email using a hyperlink. The code for the `sendmail.php` is also given to you. While developing your exploits, you can mail the stolen data to your email account to verify if your attacks worked. Be sure to have the stolen data sent to the grader when submitting the final exploits.

Mozilla Firefox. We will grade your project with default settings using the latest official release of the Mozilla Firefox browser at the time the project is due. We have verified that Firefox 3.03 is a safe choice. We chose this browser for grading because it is widely available and can run on a variety of operating systems. There are subtle quirks in the way HTML and JavaScript are handled by different browsers, and some attacks that work in Internet Explorer (for example) may not work in Firefox. In particular, you should use the Mozilla way of adding listeners to events (`https://developer.mozilla.org/En/DOM:element.addEventListener`) . We recommend that you test your code on Firefox before you submit, to ensure that you will receive credit for your work.

1.1.1 Deliverables and Grading

Create files named `a.txt` containing the only the attack URL for part (a), `b.html` for part (b), `c.html` for part (c), and `d.txt` for part (d), ans submit them electronically as `proj3-part1`. The web site has lots of other interesting vulnerabilities, but you will not receive credit for finding vulnerabilities other than those described above.

We will run your attacks after wiping clean the database of registered users, so any data you submitted to StockBank website while working on the assignment will not be present during grading. We will create a user named "attacker" right before we grade each of your attacks. So, please do not rely on persistent data in the web server. We reserve the right to delete users from the database at any time during the course of the project if it gets too large, so please keep local copies of any important data you submit there. Also, your attacks will run in a restricted environment, so you can only connect to StockBank website during attacks.

1.2 Part 2 : Defenses

Now that you've figured out the attacks, it is time to do a thorough fix for the above vulnerabilities. For this part, you will need the web server code. You will make changes to the existing files in the code, to prevent that attacks that you have detected in part 1.

1.2.1 Setup

Web Server. You will need a PHP enabled web server setup to test your code. For compiling on the instructional machines, you can download and install Apache (the web site runs on version 2.2.10) from `http://httpd.apache.org/download.cgi` and PHP from `http://www.php.net` (web site uses version 5.2.6). Several reasonable tutorials are available online – we used `http://dan.drydog.com/apache2php.html` for our setup.

If your server runs as a normal user, you should host it on a high-numbered port (above 1024) so that it can work. You can configure this in the file `httpd.conf` (the line with `Listen` statement) in your server installation directory.

Alternatively, you can download a server VM image and use VMware player to have a fully functional server to start with. VMware Player is available at : `http://www.vmware.com/products/player/`. You can obtain a vanilla Ubuntu 6.06 server image online from this URL :
`sourceforge.net/project/showfiles.php?group_id=243108&package_id=296281&release_id=636680`.

Project Website Code. Code for the website is given to you. You can run : `gtar xzvf proj3-code.tgz` in the space where you want your server hosted. By default, if you use Apache 2, your web site code will reside in the `htdocs/` directory in your server installation folder. The code will untar into two subdirectories, `sitecode` and `db`. The former is the code and the latter is the database folder. You will make modifications to the files in `sitecode`.

Database. The website uses a flat file SQL database called `txt-db-api` to manage persistent user state. The database engine needs to be able to write to the `db` folder, so you should give the correct write permissions to that folder. You may place your database folder anywhere, but you should specify the fully-qualified path of the database directory to the database engine code in 3 places :

1. `includes/common.php` at Line 4, requires the full-path upto and including the PHP file `db/php-txt-db/txt-db-api.php`.

2. `db/php-txt-db/txt-db-api.php` at Line 39, requires the full-path upto and including the directory `db/php-txt-db/`.

3. `db/php-txt-db/txt-db-api.php` at Line 49,, requires the full-path to upto and including the `db/` directory.

1.2.2 Goals

- **[12 points]**. Prevent attacks A, B, C and D outlined in part 1. 3 points are awarded for preventing each class of attacks.

- **[4 points]**. Do not change the site behavior or appearance on normal inputs.

- [**4 points**]. Submit a `README` as part of your code submission for part 2, explaining briefly each change you made (in no more than 3 sentences for each change), and what types of user input (if any) will cause your site to refuse requests or show error messages.
- ***Extra Credit*** [**4 points**]. Find other cross-site scripting (XSS) attacks in this web site, at other locations in the code than the ones you have identified in part 1. Give a short exploit test case demonstrating each new XSS bug you find. Secure the web site against this bug. You receive *2 points* for finding and fixing each new XSS bug you find in StockBank web site. You may get a maximum of 6 points as extra credit.

1.2.3 Restrictions

- Do not add new files. Do not edit the files in the `includes/` directory or change the `sendmail.php` file. Do not add new database tables or columns.
- You will not receive credit for fixing any of the following issues: SQL injection vulnerabilities, attacks on other sites, database race conditions, buffer overflows, attacks that only work when `register_globals` is on, or lack of HTTPS.
- There are no specific requirements for error messages on bad input. You can sanitize the input or simply `die()`, as long as you note your decision in the `README` file. Sanitizing is probably the more user-friendly option.

1.2.4 Deliverables and Grading

Submit the following files as `proj3-part2` :

- 6 files of the root code directory of your web site : `customizer.php`, `index.php`, `login.php`, `stock.php`, `transfer.php`, and `users.php`.
- The `README` file, containing information as described.
- An optional `proj3-part2EC.tar.gz` file, which contains any additional files required for evaluating the extra credit in part 2.

For grading, we will follow the following steps :

- We will copy your submission files into the grader's web space. Grader's web space will be the same as the StockBank web space, i.e. on `sphere.cs.berkeley.edu`, with the configuration of packages described above.
- We will also copy over a fresh, unchanged version of the files in the `includes/` and `db/` directories.
- We will run the grader's solutions to Attack A, C, and D against your site and check to make sure they don't work. We will check your source code to make sure you've identified these and placed appropriate checks.
- We will look over your source code to ensure that all cross-site request forgery attacks are prevented.
- For all attacks, we will try to identify abnormal behavior to normal inputs caused by the addition of your code. You will be penalized for these, depending on how serious these are.
- We will check your `README` and files in your extra credit submission for additional cross-site scripting vulnerabilities.

It is important to check the mailing-list and newsgroup for announcements and information on the project – it is your responsibility to monitor these at least once in 24 hours for announcements. If you have any difficulties for the required setup, please set up a meeting with your TA as soon as possible. **It is prudent to not wait until the last week to test your attacks on the target web site. You should expect that `sphere.cs.berkeley.edu` will be overloaded close to the submission time.**