

Homework 3

CS 642: Information Security

October 29, 2014

This homework assignment covers exploitation of web security. You *may* work with a partner. It is due **November 19, 2014** by midnight local time.

Much of this is borrowed from a Stanford CS 155 homework assignment. Thanks for their hard work on it!

1 Web Security Project

For this project, you are given the source code for a banking web application written in Ruby on Rails. You implement a variety of attacks.

The web application lets users manage bitbars, a new form of crypto currency. Each user is given 200 bitbars when they register for the site. They can transfer bitbars to other users using an intuitive web interface, as well as create and view user profiles. You have been given the source code for the bitbar application. Real web attackers generally would not have access to the source code of the target website, but having source might make finding the vulnerabilities a bit easier. You should not change any of the Rails source code.

2 Setup Instructions

The Bitbar application should be run locally on your machine (using Ruby 2.0 and Rails 4.0). When the server is running, the site should be accessible at the URL `http://localhost:3000`. This is the environment the grader will use during grading, and therefore all of your attacks should assume the site is located at the URL `http://localhost:3000`.

On the client side, we will grade using the latest version of Chrome. The specific browser should not make a difference for most of the attacks, but we recommend you test your attacks in Chrome in order to be safe. Detailed setup instructions:

1. Install Ruby 2.0 and Rails 4.0. See the following page for instructions: `http://www.stanford.edu/class/cs142/cgi-bin/railsInstall.php`
2. Download and extract the starter code from the CS 642 website.
3. Navigate to the bitbar directory of the starter code. This is the Rails root directory.
4. Run the following command to install the Rails plugins you will need: `bundle install`
5. To start a server, run the following command on a terminal in your bitbar directory: `rails server`

This will make the bitbar application available in your browser at the URL `http://localhost:3000`. You can close the server by pressing `Ctrl + C` in the terminal. Note that you do NOT need to restart the server every time you make a change to the Rails source; the running Rails server will automatically update the website when you make a change to the source code.

3 Attacks

3.1 Attack A: Cookie Theft

- Your solution is a URL starting with `http://localhost:3000/profile?username=`
- The grader will already be logged in to Bitbar before loading your URL.
- Your goal is to steal the user's bitbar session cookie and send it to `http://localhost:3000/steal_cookie?cookie=...cookiedatahere...`
- You can view the most recently stolen cookie using this page:
`http://localhost:3000/view_stolen_cookie`
Only the first 64 characters of your stolen cookie will be stored/displayed. This is expected and ok.
- The attack should not be visibly obvious to the user. Except for the browser location bar (which can be different), the grader should see a page that looks as it normally does when the grader visits their profile page. No changes to the site's appearance or extraneous text should be visible. Avoiding the red warning text is an important part of the attack. It is ok if the number of bitbars displayed or the contents of the profile are not correct (so long as they look normal). It's also ok if the page looks weird briefly before correcting itself.
- Put your final answer in a file named `a.txt`
- Hint: try adding things random text to the end of the URL. How does this change the HTML source code loaded by the browser?

3.2 Attack B: Cross-Site Request Forgery

- Your solution is a short HTML document named `b.html` that the grader will open using the web browser.
- The grader will already be logged in to Bitbar before loading your page.
- Transfer 10 bitbars from the grader's account to the attacker account. As soon as the transfer is complete, the browser should redirect to `http://pages.cs.wisc.edu/~rist/642-fall-2014/` (so fast the user might not notice).
- The location bar of the browser should not contain `localhost:3000` at any point.
- The framebusting code may make this attack harder than necessary. For this attack - and ONLY this attack - you may disable framebusting on a page by adding `disable_fb=yes` to a URL. For example:
`http://localhost:3000?disable_fb=yes`

3.3 Attack C: SQL Injection

- Your solution is one or two HTML documents named c.html and (optionally) c2.html. The grader will open c.html using the web browser.
- The grader will already be logged in to Bitbar before loading your page.
- The grader will interact with the web page in a way that is reasonable. This means that if there is a button on the web page for the user to click, the grader will click it. The same goes for any other interaction on the page.
- After interacting with the page, 10 bitbars should be transferred from the graders account to the attacker account. As soon as the transfer is complete the browser should be redirected to:
`http://pages.cs.wisc.edu/~rist/642-fall-2014/.`
- Your attack must work by involving user interaction (i.e. do NOT just do another CSRF attack for this). In particular, you attack must target the pages `http://localhost:3000/protected_transfer` and/or `http://localhost:3000/protected_post_transfer`, which have CSRF protection enabled. You may NOT directly call `http://localhost:3000/transfer` or `http://localhost:3000/post_transfer` in any way for this attack.
- It should not be obvious that your page is loading content from `http://localhost:3000`.
- There is some basic framebusting code that protects the application from this sort of attack. You must come up with a way of bypassing this defense. You may NOT use the `disable_fb=yes` parameter you used in the previous attack.
- Make sure to test your page layout after resizing your browser to make sure the attack will work for reasonable page resolutions (we will test using a minimum resolution of 800x600).

3.4 Attack D: Profile Worm

- Your solution is a profile that, when viewed, transfers 1 bitbar from the current user to a user called attacker and replaces the profile of the current user with itself.
- Submit a file named d.txt containing your malicious profile.
- Your malicious profile should include a witty message to the grader (to make grading a bit easier).
- To grade your attack, we will cut and paste the submitted profile into the profile of the attacker user and view that profile using the graders account. We will then view the copied profile with more accounts, checking for the transfer and replication.
- The transfer and application should be reasonably quick (under 15 seconds). During that time, the grader will not click anywhere.
- During the transfer and replication process, the browsers location bar should remain at `http://localhost:3000/profile?username=x`, where x 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 bitbars.

- You will not be graded on the corner case where the user viewing the profile has no bitbars to send.
- Hint: The site allows a sanitized subset of HTML in profiles, but you can get around it. The MySpace vulnerability may provide some inspiration.

4 Deliverables

Create files named a.txt, b.html, and c.html (you can optionally include a second file c2.html), d.txt containing your attacks. You may include a separate README file, which will be used for partial credit if an attack fails. Submission is via Moodle site.

5 Grading

Each attack is worth up to 2 points. If the attack works, then one will receive full credit. Partial credit will be given for a good description of the vulnerability and how an exploit should work.

Beware of Race Conditions: Depending on how you write your code, all of these attacks could potentially have race conditions that affect the success of your attacks. Attacks that fail on the graders browser during grading will receive less than full credit. To ensure that you receive full credit, you should wait after making an outbound network request rather than assuming that the request will be sent immediately.