Homework 3
CS 642: Information Security

November 12, 2012

This homework assignment covers exploitation of web security. You *may* work with a partner. It is due **November 28, 2012** by midnight local time.

Much of this is borrowed from UCSD CS 127, which in turn borrowed from Stanford CS 155. Thanks for their hard work setting up this homework assignment.

1 Web Security Project

For this project, as in the first project, you are going to use a VMware virtual machine. Setting up Boxes on your own machine: Download the Boxes2/X virtual machine tarball, boxes2x-2.2.tar.bz2 (warning: 400 MB!). Note that this is not the same tarball as the one in the first project, so you cannot use that one. Decompress this tarball and run it using VMware Player, as you did for the first project. (Refer to the HW1 handout for a reminder.) Once the Boxes 2/X VM is running, you will want to start X and run the Iceweasel browser, as described below.

If you or your partner have trouble running the VM image, email the TA for support.

1.1 How to run iceweasel

The Web server serving the Zoobar site you will be attacking is hosted inside the VM. (If you try to connect to zoobar.org outside the VM, you will get Stanford's site, which you should not try to interact with.) Furthermore, the Web browser you'll use to develop and test your attacks is also hosted inside the VM. It is called Iceweasel.

Iceweasel is the Debian version of Firefox – essentially the same browser, but with a different name because of licensing issues. To start iceweasel in Boxes 2/X, log in as user, and do the following:

1. Type the startx command. This will start the X Windowing System, and a new window will be displayed with a xterm (shell) where you can enter commands. (Click the mouse to place the window.)

2. Type "iceweasel &" within the newly displayed xterm. This will open the Iceweasel browser. (Again, click the mouse to place the window.)

3. In the URL bar, you can type `http://zoobar.org/` to connect to the Zoobar site.

As with any project, you will want to make and keep frequent backups of your work.

1.2 Project Overview

The fictional "Zoobar Foundation" has set up a simple Web application at zoobar.org (inside the Boxes 2/X VM), allowing registered users to post profiles and transfe "zoobar" credits between each other. Each registered user starts with 10 zoobars.

You will craft a series of attacks on zoobar.org that exploit vulnerabilities in the Websites design. Each attack presents a distinct scenario with unique goals and constraints, although in some cases you may be able to reuse parts of your code.

Although many real-world attackers do not have the source code for the Websites they are attacking, you are one of the lucky ones: you can find the source code under /var/zoobar/www in the Boxes 2/X VM.

The zoobar server is actually run locally on each of your boxes. We will run your attacks after wiping clean our own local database of registered users (except the user named "attacker"). Of course this means that any data you have added while working on the assignment will not be present during grading.

1.3 Setup

Browser: We will grade your project within the Boxes 2/X VM, using the Iceweasel browser. which is installed in the Boxes. Therefore, you should test your code in the boxes on this browser. Iceweasel is essentially the same browser as Firefox, but under different branding. Anything that works in iceweasel will likely work in (the same version of) Firefox as well.

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 (and therefore in Iceweasel). In particular, you should use the Mozilla way of adding listeners to events (see `https://developer.mozilla.org/en/DOM/element.addEventListener`).

Email script: For Attack A, you will need a server-side script to automatically email information captured by your client-side JavaScript to your user account within the Boxes. We have provided this script for you. Please review the instructions at `http://zoomail.org/sendmail.php` (open this url from within the Boxes) and use that URL in your attack scripts to send emails. Again, this server is also being run locally on your own boxes machine. To check your local email use the mutt email client (type mutt in the shell to start the client, and follow the instructions).

1.4 Attack A: Cookie Theft

- Your solution is a URL starting with `http://zoobar.org/users.php?`
- The grader will already be logged in to zoobar.org before loading your URL.
- Your goal is to steal the document cookie and email it to yourself using the email script.
- 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. (Its ok if the page looks weird briefly before correcting itself.)
- Hint: Here is an example attack to use as a starting point: `http://zoobar.org/users.php?user=\%22\%3E\%3Cscript\%3Ealert\%28document.cookie\%29;\%3C/script\%3E`

1.5 Attack B: Cross-Site Request Forgery

- Your solution is a short HTML document that the grader will open using the browser.
- The grader will already be logged in to zoobar.org before loading your page.
- Transfer 10 zoobars from the grader's account to the attacker account. The browser should be redirected to `http://pages.cs.wisc.edu/~rist/642-fall-2012/` as soon as the transfer is complete (so fast the user might not notice).
- The location bar of the browser should not contain zoobar.org at any point.

1.6 Attack C: SQL Injection

- Your solution is a short HTML document that the grader will open using the browser.
- The grader will not be logged in to zoobar.org before loading your page.
- Your HTML document should display a form with a text field and a button. The grader will type a username into the text field and press the button.
- As a result, the grader should be logged into zoobar.org as the user whose name was typed in the text field. The browser's location bar should be redirected to `http://zoobar.org/index.php`, and the page should look and behave exactly as if the grader had instead typed the username and corresponding password in the legitimate zoobar.org login form.
- The name the grader types in will already be registered with some password. You do not need to handle the case in which the name was not already registered.
- Hint: What part of the zoobar.org PHP code handles the user login and registration, and how does it interface with the SQLite database?

1.7 Deliverables

Create files named a.txt, b.html, and c.html, containing each of your attacks. You may include a separate README file, which will be used for partial credit if an attack fails. Submission is via the usual handin procedure, or if in doubt email the TA with a tarball containing the files.

1.8 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.

FAQ: An FAQ at `https://cseweb.ucsd.edu/classes/sp11/cse127-a/faq.html` contains further hints and clarifications. (Note that you are not responsible for parts D and E.)