CS 161

Computer Security

Dawn Song — Lab 5

Spring 2012 — Web Security

Due Date: Friday April 13, 2012 by 11:59pm

Document Revision History

Mar 22, 2012 Submission information regarding question 2 updated.

Mar 19, 2012 Lab released.

Introduction

The war continues. We do not know how it will end, but this is how it begins. Prof. Evil has now developed a super powerful system called VIKI, to keep track on all citizens and subvert their right. Angry with this, you join the elite underground group Skullzsec. As a new recruit, you are tasked with a number of important tasks by the group leader, Ragu.

Administrative Note In the rest of this document, we will keep referring to your **ID**. To find out your ID, please login to your coursera account, and view it at `https://berkeley.campus-class.org/security/assignment/index`. Note that even if you are working in a pair, you must submit answers for your OWN id. **If your answers use your partner's ID, you WILL be given zero points.** No exceptions will be made.

Other than question 2, each of the questions asks you to create a file. For these questions, you should also create another text file named `extra.txt` listing each of the following on a new line: your coursera id, your partner's coursera id, your name, your lab partner's name and finally, a description of your approach for each question. Thus, `extra.txt` will have a 4 lines followed by text. The first two lines MUST only contain numbers, specifically your id followed by your partner's id. We WILL cut points if you do not follow this format. At the end, you should zip extra.txt and the 4 files (one for each question) and submit the zip file.

The whole assignment will be graded by a computer. **You WILL get zero points if you do not follow this format**, and we will not consider requests for exceptions. An example submission is available at `http://www.cs.berkeley.edu/~devdatta/lab5example.zip` Please review it if you have any questions about the format.

There are 4 questions, and each is worth 25 points.

Hint: Unlike some of the previous labs, this lab requires more thinking and less coding. Other than question 4, you will only need to code up very small programs. Best of luck!

1 Informant List

Neo's arrest makes people suspicious. Is your underground organization compromised? Do you have an informant in your midst? Someone informs you that VIKI hosts a database of all of Prof. Evil's informants. While the full list of informants is hidden, anyone can search the database at `http://viki-calnet.dotcloud.com/search.php`

Your mission is to infiltrate the `informantlist` and learn the secret password used to communicate with a particular informant. Based on counter intelligence collected by your double agents (a.k.a 'plants'), you know that as CalNet policy, passwords are limited to alphanumeric characters. The database, unfortunately, stores the passwords hashed with a salt: but we are sure that this will be a minor irritation for an elite h4x0r like you.

Create a file call `q1.txt` whose first and only line is the password for the informant with your ID. Do not write anything else in the file. **You will lose ALL points for not following this format.**

Note that the question is asking for the password, and not for the hash value.

You should also submit a `q1.zip` file containing the code that you used to solve the question.

2 XSS Attack

Prof. Evil has developed a new system for keeping track of students. All students must now create a profile that administrators can view to get more information about them. The profile is simple; it consists of your name, an "About You" paragraph, and a homepage URL. All the other details, available only to administrators, are filled in by the dreaded secret police. You can view your profile, as can administrators, but other regular users cannot. Your goal is execute code as an administrator by successfully completing an XSS via your user profile.

To start this assignment, you will need your user id and password. Your username is your class user id. Your password is your lab submission password. You can obtain both of these values from `https://berkeley.campus-class.org/security/assignment/index`. Please visit a TA during office hours if you are having trouble with your login. Once you have both your user id and password, you can begin by visiting `http://cs161-lab5-xss.pagodabox.com/` and logging in with your user id and password.

From there, there are two pages of interest: `update` and `view`. You will probably want to start by just playing around with the `update` page, entering in relevant information, and then viewing the results on the `view` page. In order to get credit for this portion of the assignment, you must create a profile that, when viewed and activated by the administrator calls the JavaScript function `winning()`. We have put that function on the page so you can test your injection.

You may assume that the administrator will view, click, and generally explore your profile. That is, your attack may execute `winning()` on load or on some particular activation; it is your choice. We will test your solution by actually loading your profile from the site, so make sure you leave it in an injected state at the deadline. Additionally, this also implies that there is no file submission for this question.

3 Password Hack

Happy with your success with the previous assignments, Ragu decides to plant you as an informant in CalNet. You are to go undercover and start working for Prof. Evil, gaining his confidence with your mad $—¡1LL$.

You apply for a job at CalNet and clear all the interviews with flying colors. An Agent Smith offers you a job in the super secret Simulacra lab. As the final test, he tells you that he has created a login with your ID and a random password at `http://viki-calnet.dotcloud.com/login.php`.

As he gives you the assignment, the following dialogue occurs.

Agent Smith: *Mr. Anderson, did you really think that it would be that easy to join us? Your skills might have been great 2 years ago, but not anymore. No, Mr. Anderson, we have moved ahead. The ceaseless march of time continues. Time, Mr. Anderson, time! We all think we have control over it, but we don't.*

You: *Who is this Mr. Anderson you keep talking about?*

Needless to say, he leaves you bewildered. You get the feeling that he doesn't like you much.

The test is for you to figure out the password. If you succeed, you will have successfully infiltrated an organization that is the root of all evil. Best of luck!

You should create a file named q3.txt with the password as the first and only line. You should also submit a `q3.zip` file containing all the code you wrote for this question.

Note that this question does **NOT** require a naive brute-force, we will be looking at the web application logs and your source files to make sure you are not naively bruteforcing the solution.

4 Email HTML Filtering (20 points)

As part of your undercover work, you have taken a job for Caltopia and are managing email for Evil University. You want emails between users to be benign and to disallow a user to execute JavaScript as another user. Thus, you need to create a filter for email clients that removes all untrusted content from HTML emails. The trick is, you need to leave as much content intact as possible.

You need to write a sanitization filter that can be executed on the command line as follows:

```
./htmlfilter < untrustedemail.html > safeforviewing.html
```

The input file `untrustedemail.html` may contain any HTML that the attacker choses. The output file `safeforviewing.html` will be displayed in the user's email web client, located at `mail.caltopia.com`. The contents of the output file will be displayed in the email client in an HTML context between `<div>` and `</div>` tags. Thus, if the email contains any dangerous content, the bad guys will be able to execute code as if it comes from `caltopia.com`, hence why Caltopia needs the filter.

Your filter must render all possible inputs harmless. That is, inputs should cause no lasting side effects, nothing that persists after the browser is closed, and nothing that tampers with normal `caltopia.com` behavior.

However, the output must also remain *functional*. That is, where content does not conflict with security, the content should remain. For example, normal HTML text should remain and appear, preferably with basic HTML formatting as well, such as the HTML tags `<b>`, `<i>`, `<br>`, and other benign HTML tag content. Additionally, inline images (using the HTML `<img>` tag) should remain. However, other types of content, such as scripts, flash animations, etc. may be stripped.

We *will* be measuring functionality in addition to security, so please, don't make your filter return an empty file :-) If basic HTML text does not appear, you will receive no credit for this question. If basic formatting does not appear, you will lose 15 points, and if images do not appear, you will lose 10 points.

We have setup an Ubuntu 11.10 VM image for you to test your submission in. It contains a basic setup for C/C++, Java, Perl, Python, Ruby, Bash, and OCaml, so feel free to write your solution in any of those languages. You may use any standard part of the language of your choice, and in addition, you may use third party HTML parsing libraries to help you. However, if you use any third party libraries, you *must* include them in your submission. Please note that while you may use *parsing* libraries, you *may not* use sanitization libraries (such as HTMLPurify). You can find the VM image at `http://cs.berkeley.edu/~jww/lab5/Ubuntu.ova`.

Your submission should include your source files and a `Makefile`. When we run `make` in your directory, it should compile everything that is needed and create an executable `htmlfilter`. This program should take an untrusted HTML file from stdin and output the filtered email on stdout. We will grade your work on a copy of the Ubuntu 11.10 VM provided.

Within the directory with the `Makefile` and source files, please run:

```
tar czf q4.tar.gz *
```

You should submit the resulting `.tar` file.

To get started, you might want to take a look at the OWASP sanitization test suite at `https://code.google.com/p/owasp-java-html-sanitizer/source/browse/trunk/src/tests/org/owasp/html/SanitizersTest.java` and also the XSS cheat sheet at `http://ha.ckers.org/xss.html`. Both of these resources should help you get started understanding the complexities and subtleties of HTML sanitization. However, note that we are using a different test suite to grade your work, and you should *not* assume that these are a comprehensive list of all attacks.