Summer University, Web Security Lab 4, July 2014 Name: Anushree, Denis, Israel, Raphael

Group: Calvin Hobbes

1. (10 points) This exercise will walk through the Caja playground. Go to http://caja.appspot.com/. Select ES5/3 mode from the dropdown menu.

   (a) Under "Attacks", click on 'sniffing history'. Click on 'Cajole'. Because the guest code on the page is prevented from accessing your history, the link is blue. (Note that this example is a little outdated, as newer browsers do not always permit access to the computed color of links. However, the code has been rewritten so that the guest code could not determine the user's history even without browser restrictions.) Now click on the 'Cajoled Source' tab. Note that the HTML has been rewritten as well as the JavaScript code. Give a brief description of the changes.

   Characters are replaced with their HTML entities. All the code is wrapped in a <caja-v-html><caja-v-head> wrapper. The Javascript is expanded, because code is loaded from a specified location.

   (b) Experiment with the other attack examples. What would happen in each case without Caja? What happens with Caja?

   Using the redirect attack, the button generated does not work. And the same happens when attempting to access a cookie. But the Auto Submit form ends up working.

   Without Caja all Javascript that is attempting to run something cross site will end up executing.

   With Caja, code is replaced not allowing changes to be made to the current page.

   (c) How recognizable is the translated code? Why is it so complex?

   The code is not very recognizable because Caja is making the HTML, CSS, and JS into a more sanitized form in order for it to reference everything, making sure that all code ran is from the direct source.

2. (10 points) Extend eliza.rb from the course website. Note that if you call `ruby eliza.rb -test`, you will get some cases to consider.

   If the patient says "always", "never", or something similar, Eliza should respond CAN YOU BE MORE SPECIFIC?

   If the patient asks a question beginning with "Are you" or something similar, ELIZA should respond with IS IT IMPORTANT IF I AM ⟨original statement⟩.

   Filter out words like "Well" or "Perhaps" from the beginning of the sentence.

   Add an additional memory for Eliza to support "they", following the pattern for "he" and "she".

   Finally, add one change of your own to make Eliza seem more human.

3. (10 points) This question explores Ruby's taint mode. Download taint.rb from the course website.

   (a) Update the Record class so that updates with either a tainted name or a tainted value are ignored. Do this first by explicitly checking the taint on a field.

   (b) Would this be sufficient if an attacker could control part of the code? If not, how could the different taint modes be useful?

   You can untaint variables so that you can bypass the security.

   When running in a safe mode, potentially dangerous methods will raise a Security Error if passed a tainted object.