Computer Security
Coursework Exercise CW2

Exploitation

School of Informatics
University of Edinburgh
http://www.inf.ed.ac.uk/teaching/courses/cs

This is an individual assessed coursework exercise. It will be awarded a mark out of 25. It is one of two assessed exercises in the Computer Security course. Each exercise is worth 12.5% of the final result for the course. The deadline for completing this coursework is 16:00, 20th April 2015. This exercise explores some topics in exploiting and preventing vulnerabilities. This answers should be submitted on paper, by hand at the ITO. To answer the questions you will need to consult the lecture slides and additional resources such as the referenced papers that are cited in the lecture slides.

Question 1 The following C code has a vulnerability:

```
void vuln(uint64_t in[], size_t len)
{ char a[20];
  size_t i;
  for (i=0; (i<len) && (i<20); i++)
    memcpy(&(a[i]), &(in[i]), sizeof(uint64_t));
}
```

(a) Depict the stack frame of a call to `vuln()` before the execution of the `for` loop. You can assume the code is executed on a 32-bit machine, with the size of the `uint64_t` data-type being 8 bytes and the size of `size_t` being 4 bytes.

(b) What is the *maximum* number of bytes from the start of `a` that `vuln()` can write in terms of `len`?

(c) Assume that `in` and `len` are under the control of the attacker, but the calling function checks that `vuln()` is only called with `len` equal to the number of elements in `in`. The `vuln()` function is still vulnerable to a buffer overflow attack. Explain why, and give values of `len` that would enable such an attack.

(d) The C compiler can insert stack canaries to help prevent buffer overflow attacks. Describe what a stack canary is and how they prevent buffer overflow attacks. Would the canary prevent this vulnerability being exploited and why?

(e) Is it possible to exploit this vulnerability to gain the ability to execute any code and why? What is the maximum damage an attacker can inflict? Give your answer in the cases where there are, and then where there are not stack canaries.

Question 2 Alice is browsing a website that contains a reflected XSS vulnerability. Whenever she attempts to browse a page that doesn't exist the site's `404` page displays the URL she tried to access along with a *"does not exist"* message. The URL is displayed back without any attempt to escape or sanetize its content.

(a) Describe briefly the difference between a reflected and a persistent XSS vulnerability. Why is it important to sanitize all user inputs?

(b) Lets suppose the website is `vulnerable.net`. Describe how to exploit this vulnerability to run the following JavaScript snippet on the victim's browser:

```
alert("XSS!");
```

(c) Alice has been given a persistent cookie on `vulnerable.net` called `auth`. The cookie was given to her after she logged in and allows her to remain logged in without having to reauthenticate. Mallory wishes to steal it. Describe how Mallory could perform her attack using the XSS vulnerability from part *(b)*. You can assume Mallory has a server of her own at `mallory.net`.

After detecting the vulnerability the website developers implement a *Content Security Policy* (CSP)[1]. This involves sending an additional header along with the web page that describes where JavaScript is allowed to run.

(d) Assume that all the resources are loaded form `https://vulnerable.net`, and no plugins or frames are required to view the `404` page. Images and CSS are loaded locally, fonts from Google's Web fonts. Give a CSP header that would mitigate this vulnerability.

(e) Is this sufficient to fix the vulnerability? Justify your answer and suggest any additional fixes as you see fit.

[1] CSP is a policy language to describe where JavaScript is allowed to run on a website. It works using white-listing, and is designed to prevent XSS attacks. If you are unfamiliar with it you might like read up on it at:

- `https://developer.mozilla.org/en-US/docs/Web/Security/CSP`
- `http://www.html5rocks.com/en/tutorials/security/content-security-policy/`