CS 294 Report:
Cross-site Scripting and Prevention

Donglin Wei
Donglin.Wei.GR@dartmouth.edu

1. Introduction

Code injection is the exploition of a computer bug that causes applications to process malicious code[4]. This kind of attack is allowed because of poor handling of input. For example, lack of proper input and output data validation.
Cross-site scripting, known as XSS , and HTML injections are two common types of code injection. Both of them inject malicious code to benign and trusted websites. They are different not because of the severity of vulnerability, but in the type of attack that leverages the vulnerability[1]. While XSS focus on injecting malicious Javascript code, HTML injection focus on changing the HTML page for malicious reasons.
The report will first discuss how a successful XSS attack is achieved. Then I will focus on how to replicate a vulnerable environment and the demonstration of a successful attack. At last, I will show how to avoid and prevent this kind of attack.

2. Cross-site scripting (XSS)

Cross-site scripting occurs when dynamic generated web pages display something that is maliciously injected and not properly validated. The attack can let attacker embed malicious JavaScript code into the generated page[2] and users of that website will see the ouput when they visit. As a result, the hacker is able to bypass access controls, gain higher-level rights, deface web pages and get sensitive data(usernames, passwords or even credit card numbers).
This vulnerability can be utilized in many fileds. Below are some of the common fields:

- Injecting throul (HTML) form inputs
- URL injection
- Search engines that print out the search keyword that was entered
- Injection through other kinds of inputs(email, texts, etc.)

XSS attacks are very frequent in web applications. This is bcause XSS is not considered as a serious flaw by web developers and the consequences are largely under-estimated.
To understand what the attack does, the following section describes a concise example.

3. An example: Steal password from users

In order to produce more friendly user experience, many web applications will alow users to enter data and as a result, generates a new web page based on the user's input. This will leave attackers an oppurtunity to attack web applications. For example, in a search engine where a key word is required, instead of searching for a normal record, the attacker enters

"<script> alert("Type anything you want") </script>"

and the resulted web page will possbily execute this JavaScript code if data validation is not properly done to protect the system.
To simulate XSS, I ceate a login system for an airline company. The system receives username and password from users without doing any validation.

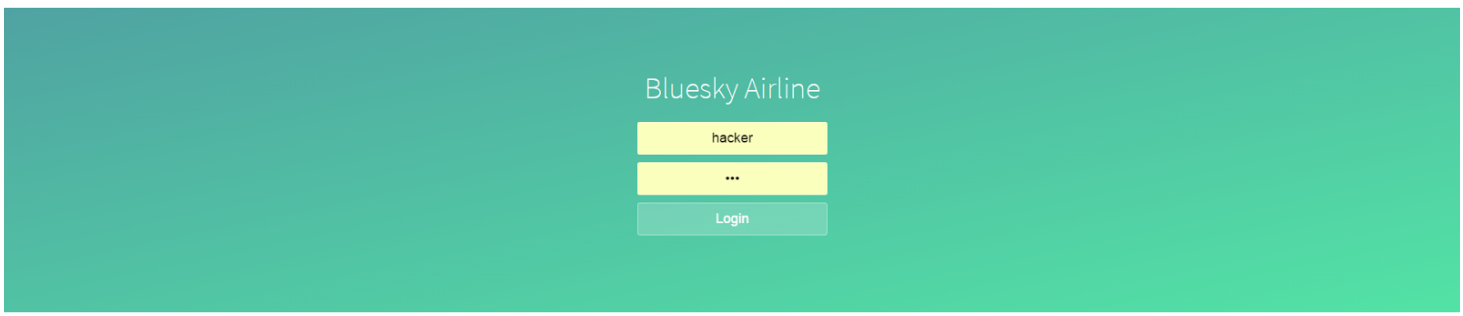

Figure 1: Login page with wrong username.

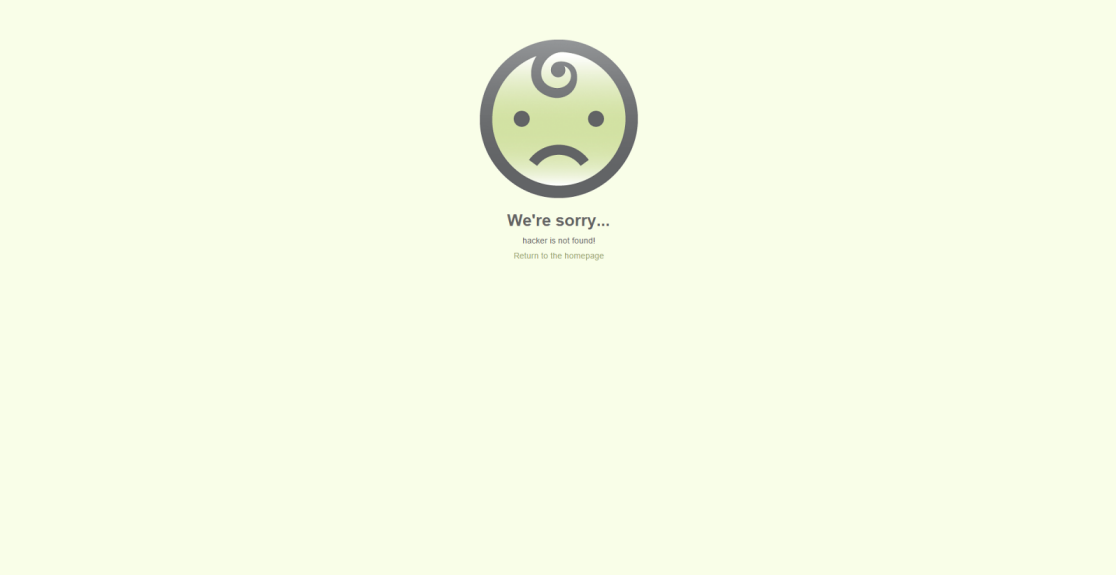

Figure 2: The failed login response

By entering a wrong username, the attacker knows that the system will “hand back” information in the error page, where the wrong username is printed.
Now, the attacker will try to inject HTML and JavaScript code into the login page. This can be easily achieved by entering the code into the username field.

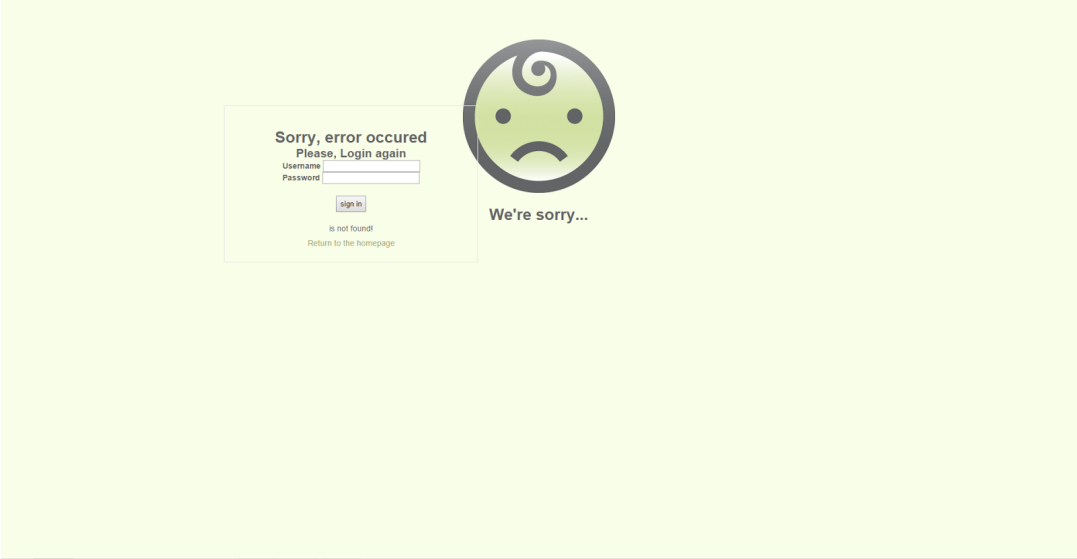

Figure 3: Error page after being hacked

In this example, the injected code create a new input form which ask users for his username and password. And when sign in is pushed, it will redirect the user to a malicious web page which captures the user information. In this case, all the private information of the user is exposed.
Although this is just a simple example, it shows how XSS is able to hack web applications.

4. XSS Prevention

Just as the example shown, XSS can be a real threat to web applications[5].
(1) One way to reduce this problem is b applying proper filtration on user-supplied data. For example, all client-supplied data should be converted to HTML character entities before showing to clients and filter out HTML tags if the data is not “rich text”[3].
(2) Another way to alleviate the problem is by using a firewall to drom frames containing known XSS based on a signature.
(3) For web users, they should be selective about how they initially visit a website. For example, don’t click links on untrusted web pages.

Reference

[1] Wikipedia.org, “Cross-site scripting”
https://www.owasp.org/index.php/HTML_Injection
[2] Spett, Kevin. "Cross-site scripting." *SPI Labs* 1 (2005): 1-20.
[3] Kirda, Engin, et al. "Noxes: a client-side solution for mitigating crosssite scripting attacks." *Proceedings of the 2006 ACM symposium on Applied computing*. ACM, 2006.
[4] Kieyzun, Adam, et al. "Automatic creation of SQL injection and cross-site scripting attacks." *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on*. IEEE, 2009.
[5] Fogie, Seth, et al. *XSS Attacks: Cross Site Scripting Exploits and Defense*. Syngress, 2011.