Security Analysis of Cross-Site Scripting Vulnerabilities in Reunion.com (Dec 2008)

Samir Gupta, Edwin Jaury, Onn Tai Yong, and Yan Chiu, *EECE412 Group 7*

***Abstract*—This report studies the cross-site scripting (XSS) vulnerabilities in a very popular social networking site, Reunion.com. Introductions to cross-site scripting and Reunion.com are provided, as well as the vulnerabilities found in the website and the methods used to discover and exploit these vulnerabilities. The report concludes with some possible solutions to the vulnerabilities found.**

I. Introduction

SINCE their introduction, social networking sites have attracted millions of users looking to express their individuality online. Over the last few years, this class of websites has diversified in the type of information they allow users to express [7]. Despite the growing popularity of social network sites, little research has been made on the security vulnerabilities of allowing users to upload and share information over the web. The purpose of this report is to analyze the cross-site scripting vulnerabilities of one particular social-networking website, Reunion.com.

Reunion.com is currently ranked as one of the top 10 North American social networking websites, and has over 40 million members worldwide. The website allows its members to share their personal information, such as their names, hobbies, interests, address. Reunion.com also allows members to create and share photo albums with other members.

Cross-site scripting, or XSS, is a very important type of security vulnerability that allows code injection by malicious web users into the web pages viewed by other uses. XSS attacks are written in a markup language such as HTML, combined with a client-side scripting language such as Javascript [3, pp. 130-150].

Reunion.com was found to have many XSS vulnerabilities that allow potential attackers to perform very damaging actions. The next sections describe the specific vulnerabilities found in Reunion.com, the methods taken to discover and exploit these vulnerabilities, and potential solutions to eliminate and reduce the impact of these vulnerabilities.

II. Reunion.com Vulnerabilities and Design Flaws

Three major vulnerabilities were found in Reunion.com through the course of this analysis: XSS vulnerabilities, cross-site request forgery vulnerabilities, and authentication vulnerabilities.

A. XSS Vulnerabilities

In the Photos page of Reunion.com, users can add albums and set their names and descriptions. The critical security risk that let us accomplish our attack was that Reunion.com allows script tags as input for the album name and album description.

This vulnerability is the result of violating the *Complete Mediation* principle of designing secure systems. This principle requires that every access to every object in a system must be checked for authority. Filter should have been employed by Reunion.com on all input forms, including the Album description input to ensure that no malicious inputs were entered.

B. Cross-Site Request Forgery Vulnerabilities

Cross-site request forgery, or CSRF, is a type of attack similar to XSS, except that it allows attackers to make requests to other websites on behalf of the victim.

Once a script is inserted into the Photos page, any user that views this page will run the script in their browser. This gives the script significant control over the victim's browser, and can be used to navigate the victim to other pages, and steal cookies from the victim's browser. For example, the script could cause the victim's browser to navigate to the victim's email site and steal the victim's contact list or emails. This attack is CSRF because it exploits the trust that the email site has that the requests are coming from the user and not a script.

This vulnerability is a result of violating the *Questioning Assumptions* principle of designing secure systems. Reunion.com assumes that all requests made to and from its website are by trustworthy members. Instead, the website should check for suspicious requests that might indicate they are coming from a script.

C. Authentication Vulnerabilities

Reunion.com allows users to change their passwords or delete their accounts. However, the website does not require any more authentication to perform these actions. For example, the user is not required to enter their old password when changing their password or deleting their account. Combined with Reunion.com's XSS and CSRF vulnerabilities, an attacker could very easily change a victim's password or delete a victim's account.

The *Defense in Depth* principle of designing secure systems was violated in this case. This principle requires a system to layer its defenses. Having a user re-authenticate himself when performing important actions will reduce the likelihood that the action is being made by a malicious attacker.

The next section describes the XSS and authentication vulnerabilities in detail, how they were discovered, and how they were exploited.

III. Methods of Attack

A. *Finding the XSS Vulnerabilities in Reunion.com*

The first step to attacking Reunion.com was finding XSS vulnerabilities in the site. This required inserting a basic alert script into every input field of Reunion.com until the script was actually executed and an alert box opened up in our browser. An example is shown in *figure 1* where the script is inserted into our Profile information.

Fig. 1. Inserting a basic alert script into all the input fields of the Edit Profile page of Reunion.com.

We found that the scripts inserted into our profile information were not recognized as HTML scripts because they were converted to UML-Encoded ASCII characters when stored on the Reunion.com server. We attempted to insert scripts in every page of Reunion.com - include Blogs, Comments, Announcements, and Profiles – and discovered the same problem. We finally found that scripts inserted into the Photos page of our profile were successfully executed.

Reunion.com allows members to create albums and create an album description in the "Photos" section of their profile. Both the Album Name and the Album Description fields allow the user to inject a script. The Album Name field has a length constraint of 10 characters, which is too short to inject a script. For this reason our scripts were inserted into the Album Description field, which has a longer length constraint of 90 characters. *Figure 2* shows the script being inserted and executed in the Album Description field of the Photos page.

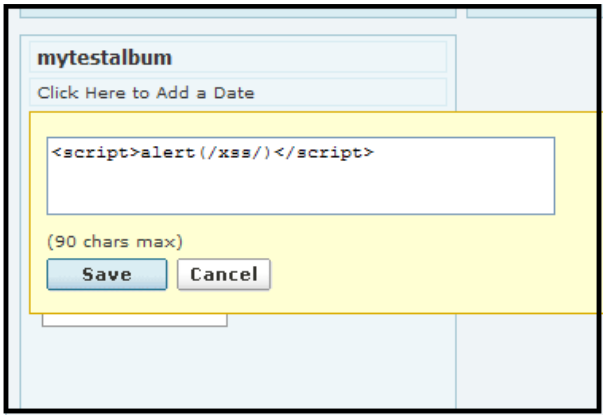

Fig. 2. Inserting a basic alert script into album description field and being successfully executed

Once we found this injection point, the next step was to host a script on an FTP server.

B. *Hosting an XSS Script on a Server*

The scripts were stored on the UBC EECE FTP servers provided to us. *Figure 3* shows the location of the script on one of our EECE accounts.

Fig. 3. Hosting a JavaScript file on our UBC EECE server.

The script inserted into the Album Description field was then changed to point to the location of *myscript.js* on the EECE server. The new script is shown in *figure 4*.

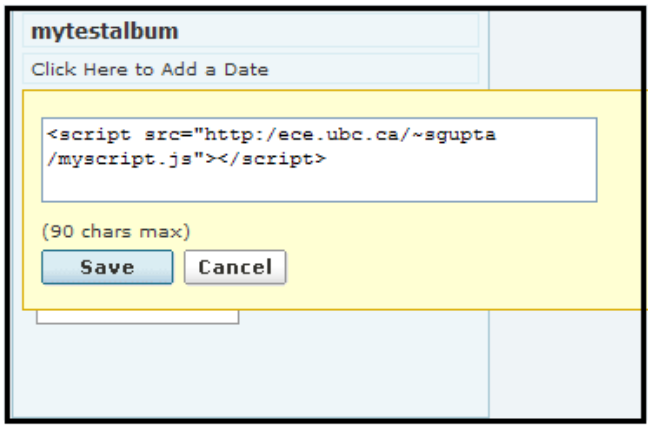

Fig. 4. Creating a script in the Album description field to point to the JavaScript file stored on the FTP Server.

Once the script was successfully hosted on our FTP server, the next step was to modify the script to actually perform an attack.

C. *Crafting the Reunion.com Attacks*

We were able to successfully create and execute six attacks: editing the victim's profile, changing the victim's login password, inviting a friend on behalf of the victim, creating a photo album in the victim's profile, and deleting the victim's Reunion.com account.

In all attacks, an iframe is first created that navigates to the desired page. For example, to create an iframe and direct it to the victim's Edit Profile page the following javascript is used:

```
var iframe = document.createElement("IFRAME");
iframe.setAttribute(src,
                    "http://www.reunion.com/showEditProfile.do");
document.body.appendChild(iframe);
```

Once the iframe is created, the script has full access to all of the directed page's elements, including its input fields and buttons. The next sections provide detailed descriptions of each attack.

1) Editing the Victim's Profile

First, an iframe is created and directed to the Edit Profile page, *reunion.com/showEditProfile.do*. To change the victim's First Name, the text field containing the first name is retrieved and changed.

```
var firstname =
        iframe.contentDocument.getElementsbyName('firstName');
firstname.value = "Stinky";
```

To submit the new First Name, the script can "click" the Save button at the end of the Edit Profile page using the javascript click() method.

```
var saveButton =
        targetFrame.contentDocument.getElementsbyName('save');
saveButton.click();
```

Once the button is clicked by the script, the victim's web browser will send the victim's changed profile information to the Reunion.com server. The next time the victim views their Profile, they will see their first name changed to "Stinky".

An attacker could potentially use this attack to modify or retrieve any of the victim's personal information – including hobbies, interests, home address, phone number, and email - by simply changing the name of the variable retrieved from the iframe. This exploit shows that the integrity property of information security has been compromised.

2) Changing the Victim's Login Password

This attack is very similar to editing the victim's profile information, with the exception that the iframe is directed to the Change Password page, *reunion.com/showLoginInfo.do*.

Reunion.com does not require a user to re-enter their old password when changing a password. For this reason, the script can retrieve the password field and set its value to another password and click the submit button using the same method as editing the victim's profile.

The following code shows how to change the victim's password to "attackerspassword":

```
var password =
    targetFrame.contentDocument.getElementsbyName('password');
password.value = "attackerspassword";

var saveButton =
        targetFrame.contentDocument.getElementsbyName('save');
saveButton.click();
```

An attacker could potentially use this attack for identity theft by changing a victim's password to one that the attacker knows. The attacker would then be able to log in as the victim and completely take-over the victim's profile. This means that the victim's confidentiality has been breached. The attacker could also use this vulnerability to perform a denial-of-service attack by not allowing the victim to login.

3) Inviting a Friend on Behalf of the Victim

This attack is also similar to the first editing a victim's profile and password, except the iframe is directed to the Friend Invite page, *reunion.com/invite.do*.

Reunion.com allows members to block their profiles from being viewed by certain members. For example, a member may set their profile viewable only by their friends. Members can also add friends by sending them "Friend Invites". If a member receives a Friend Invite, they will become friends with the inviter and gain viewing privileges to the inviter's profile.

For this attack, the script sets the "Invitee" text field of the Friend Invite page to our email addresses. The script then clicks the Invite button to send the invite on behalf of the victim. We then receive an email from Reunion.com indicating that the victim has added us as a friend and we can now view the victim's profile. The following code is used to send a friend invite on behalf of the victim:

```
var invitee =
    targetFrame.contentDocument.getElementsbyName('invitee');
invitee.value = "attackersemail@hotmail.com";

var sendButton =
  targetFrame.contentDocument.getElementsbyName('sendInvite');
sendButton.click();
```

An attacker could use this attack to view profiles they are not allowed to view by setting the Invitee field to his own email address. Whenever a victim that has restricted their profile is attacked, the attacker will become the victim's "Friend" on Reunion.com and gain viewing rights to the victim's profile.

4) Creating an Album on the Victim's Profile

This attack takes advantage of Reunion.com's existing AJAX scripts to add albums and set their descriptions. The script first opens an iframe and directs it to the victim's Photo's page, *reunion.com/photo/album/allalbums.do*.

Once the iframe has loaded, the script calls the AJAX method PhotoDwr.createAlbum() to create an album on the victim's profile. The createAlbum() method is an existing method embedded in the Photo page returned from Reunion.com. Many other AJAX methods that attack script

can use are embedded in this page, including updatePhotoAlbum(), which modifies the album information. The following code creates an album called “test album” on the victim’s page and sets the new album’s description to “test description”:

```
iframe.PhotoDwr.createAlbum(“test album”, “”, null);

var createdAlbumID = getElementsbyName(“test album”).id;
iframe.PhotoDwr.updatePhotoAlbum(createdAlbumID, 0,
                        “test album”, “test description”, “”, null);
```

When the victim views his Photos page, he will see a new album called “test album” with description “test description”.

An attacker could use this attack to create a virus. Whenever a victim is attacked, a new album can be created on the victim’s profile with the description set to the script on the attacker’s server. Now, whenever one of the victim’s friends views the victim’s photos, the same script will be executed to create an album on the victim’s friend’s profile. This virus could spread very quickly if enough people viewed the malicious album.

5) Deleting a Victim’s Reunion.com Account

Reunion.com allows members to delete their own account with the click of one button, without requiring any additional authentication. To do this, the script simply directs the iframe to the Delete Account page, *reunion.com/deleteAccount.do*. Once the iframe is loaded, the victim’s account is automatically deleted, and the victim is logged out and can not log back in.

An attacker could use this attack to delete any account he wishes to because Reunion.com does not require the users to re-authenticate themselves when deleting an account. With a deleted account, availability to the victim’s account is taken, decreasing the value of another information security property.

D. Luring Victims to Our Photos Page

Once the attacks have been made, the next step is to lure victims to the photos page of our profile. This can be done in many ways, including uploading interesting pictures to entice victims to view our photos.

The next section describes some possible solutions to the XSS vulnerabilities in Reunion.com that will prevent or reduce the impact of these attacks.

IV. COUNTERMEASURES

As a social networking website, Reunion.com has two possible ways of dealing with its XSS vulnerabilities: eliminating them, or reducing their impact.

A. Eliminating XSS Vulnerabilities

There are many solutions that other social networking websites have employed to reduce or eliminate XSS vulnerabilities. The most effective solutions to XSS are input filtering, input encoding, and subdomains.

1) Input Filtering

The first layer of defence to XSS attack is input filtering. Before being stored on a server, all inputs from a website are first filtered for common hazardous content, such as JavaScript. Filters can be used to remove special characters such as <, >, &, as well as some HTML tags such as “script” [3, pp. 396-400].

This simple solution greatly increases the security of a web application. However, the website developer must ensure that every user input entered in a form is passed to the filter before being stored at the server or sent back to a client’s browser. A single vulnerable injection point can lead to a disastrous consequence to the web application if exploited by an attacker.

If Reunion.com filters the inputs fields from the Album Description, a script tag such as the one described in our attack will be detected and therefore will not be executed.

2) Input Encoding

To couple input filtering, it is advised to also encode user input. If an attacker knows that a website uses input filtering, he can encode special characters to their ASCII-encoded values. For example, encoding the character ‘<’ to ‘&#60’ will allow the attacker’s script to bypass any input filter, since the filter will be searching for ‘<’ and not ‘&#60’ [6, pp. 220].

Input encoding can be used to convert a user input into URL-encoded values. URL encoding replaces unsafe characters with a ‘%’ followed by two hexadecimal digits corresponding to the ASCII character set [4]. For example, ‘<’ or ‘&#60’ will be replaced with ‘%3C’. The URL-encoded value will not be recognized as the browser as HTML tags and will therefore not be executed on the client [8].

Input filtering combined with input encoding will eliminate virtually all possible XSS attacks on Reunion.com. Un-encoded scripts will be detected by the input filter, and any encoded scripts that bypass the filter won’t even be recognized by the victim’s browser since they will be URL encoded.

3) *Using Subdomains*

A major problem with input filtering and encoding is that these techniques limit the type of information website users can publish on their profiles. Many social networking sites, including Reunion.com, want their users to be able to create their own HTML content and share it by publishing it on their profiles. Input filtering and encoding filters will eliminate most HTML content.

A solution to this problem is for web developers to utilize subdomains. That is, to make Website pages that allow HTML input as sub-domains of pages that do not allow HTML input. The *Same Origin Policy* implemented in all latest web browsers does not allow scripts being executed in one domain to access elements of a page in another domain [9].

As an example, consider Reunion.com, which allows HTML script tags to be placed in the Photos page of the website. Currently, the domain of the Photos page is www.reunion.com, which is the same domain as the Edit Profile, Change Password, and Delete Account pages. If the domain of the

Photos page was changed to a subdomain of www.reunion.com, such as photos.reunion.com, scripts inserted in the Photos page would not be able to access any elements of the other pages since the domain of the script is different.

The *Same Origin Policy* works because different domains have different cookies, and all the cookies of the same domain have the same cookies [1, pp.22-23]. In the above example, the Photos page will have all cookies in the photos.reunion.com domain, but will not have the cookies of the www.reunion.com domain and will therefore be denied access by the victim's web browser.

B. *Reducing the Impact of XSS Vulnerabilities*

One of the major vulnerabilities in Reunion.com is that the user is not re-authenticated when performing serious actions, such as changing a password or deleting an account. Even if Reunion.com does not employ any techniques to eliminate its XSS vulnerabilities, it should at least re-authenticate its users when they request to change their passwords or delete their account.

For further readings on other possible solutions to prevent XSS attacks, see [2] and [5]. A paper written by Philipp Vogt *et al*. suggests an interesting approach to XSS security [6].

The next section describes a similar attack to ours done on another popular social networking site.

V. Another XSS Attack

The Samy Worm, which attacked MySpace.com in October 2005, was a similar attack that exploited XSS vulnerability in MySpace.com. The worm's payload added the comment "but most of all, Samy is my hero" to the victim's profile and sent a Friend Invite to Samy, the author of the worm. It infected one million users within 20 hours before bringing down MySpace and was one of the fastest spreading worms [2].

The Samy Worm was similar to our attack in that it required MySpace users to visit Samy's profile to be executed. Victim would be injected by the worm when he viewed Samy's profile. It differs from our attack in its implementation. Samy's attack utilized the AJAX XMLHTTPRequest POST method to edit the victim's profiles and add friends [3, pp. 387-388]. Our attack utilized JavaScript to manipulate iframes and DOM-based objects.

The Samy Worm showed the social networking community how popular websites are prone to XSS attacks. Unfortunately, many websites, including Reunion.com, have still not eliminated the threat of XSS.

VI. Conclusion

Our analysis on Reunion.com has showed that an XSS-vulnerable web application, specifically a social networking site, could put its users' privacy and personal information at risk. Not only can an attacker modify a victim's profile, but he can also highjack a victim's account by changing his password or deleting the victim's account.

The design of Reunion.com does not follow several principles of designing secure systems. Our attacks also show that the confidentiality, integrity, and availability of information on Reunion.com can be compromised.

To aid Reunion.com in patching its XSS security holes, we will report our findings to Reunion.com so that these vulnerabilities can be addressed in the near future.

References

[1] R. Cannings, H. Dwivedi, and Z. Lackey, *Hacking Exposed: Web 2.0*. NY: McGraw-Hill, 2008.

[2] J. Grossman. Cross Site Scripting Worms and Viruses: the Impending Threat and the Best Defense. presented at the 2006 Whitehats Conference [Online]. Available: http://net-security.org/dl/articles/WHXSSThreats.pdf

[3] F. Seth, J. Grossman, R. Hansen, A. Rager, P. Petkov, *XSS Exploits: Cross Site Scripting Attacks and Defense*. Burlington, MA: Syngress Publishing, Inc., 2007.

[4] J. Scambray and M. Shema, *Hacking Exposed: Web Applications*. pp. 200-230. Barkeley, CA: McGraw-Hill, 2002.

[5] A. Klein, "DOM Based Cross Site Scripting or XSS of the Third Kind," July 2005.

[6] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, G. Vigna, "Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis," Secure Systems Lab, Technical Univ. Vienna and Univ. California, 2007.

[7] D. Stuttard, M. Pinto, *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*, pp. 40-50. NY: McGraw-Hill, 2006.

[8] J. Erickson, *Hacking: The Art of Exploitation*, pp. 80-85. NY: McGraw-Hill, 2007.

[9] S. Cook. A Web Developer's Guide to Cross-Site Scripting [Online]. Available: http://www.grc.com/sn/files/CrossSiteScripting.pdf