Security Analysis of UBC Web Applications (November 2007)

Jeffrey Qian, Je-Yu George Lee, William Ha and Phoebe Hsu, *University of British Columbia*

Abstract - **UBC Web applications and online services are used by thousands of students and faculty members on a daily basis. How secure are these applications? In this report, we have analyzed security threat levels from sixteen UBC websites, and classified the threads found. A few vulnerabilities discovered are demonstrated for illustration purposes only and *no real damage has been done* to the web server and applications. However, these vulnerabilities that are found could facilitate other attackers to perform further actions and eventually compromise the system. Countermeasures and recommendations are provided in this report and will be forwarded to appropriate departments in UBC.**

Index Terms - **Web Application Security, Structure Query Language, SQL, Relational Database Management System, RDBMS, Cross Site Scripting, XSS, SQL Injection, Directory Traversal**

I. INTRODUCTION

Security of modern interactive web applications has been a serious issue since the employment of database services. The Structure Query Language (SQL) is the programming language for the retrieval and management of data in Relational Database Management System (RDBMS). SQL has been commonly used by web administrators to select, insert, update, and find out the location of data, but has also made the website vulnerable to malicious attack, such as SQL Injection. Moreover, any web applications with or without databases might still be vulnerable to attacks. For example, in Cross Site Scripting (XSS), an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user without validating or encoding the input. Statistics show 70% of websites are at immediate risk of being hacked. [1] Our project goal is to perform an evaluation on the vulnerability of UBC's web applications to different kinds of malicious attack.

Manuscript received November 19, 2007.
J. Qian (e-mail: jeffreycq@gmail.com).
J. Lee (e-mail: georgelee4@hotmail.com).
W. Ha (e-mail: haluvaly@msn.com).
P. Hsu (e-mail: phoebehsu@hotmail.com).

II. ATTACKING METHODS

A. SQL Injection

Being one of the many web attack mechanisms for application layer attack, SQL Injection is commonly used by hackers to steal and modify data from organizations. By taking advantage of improper parsing of inputs of the web forms, hackers could do the following steps: pass SQL commands through a web application to be executed by the backend database, query database directly, and gain access of the data held within the database. Dynamic script languages, such as ASP, ASP.NET, PHP, JSP and CGI, are vulnerable to this attack.

B. Cross Site Scripting (XSS)

Being another of the many web attack mechanisms for application layer attack, XSS allows attacker to embed malicious content, such as JavaScript, VBScript, ActiveX, HTML or Flash, into a vulnerable dynamic page to fool the user and gather victim's data. XSS may occur anywhere a web application uses input from a user in the output it generates without validating it. The consequences of such attack may be as trivial as image or layout change, or as severe as disclosure of user's session cookies [2].

C: Directory Traversal

Directory Traversal is an HTTP exploit which allows attackers to access or execute commands in restricted directories that are outside of the web server's root directory [3]. It is also known as directory climbing and backtracking. With a web browser, hackers can blindly find any default files and directories on the system if the system is vulnerable to Directory Traversal.

III. PROCEDURE

A. Scan for Vulnerabilities:

1) Find target websites

First of all, the attacker needs to find a website to attack. In order to get desirable results, an attack towards a website that has a database or a server is more reasonable. Also, attacking a university institution website is more challenging then attacking a random website on the network. Therefore, as we have mentioned in the proposal, we will use UBC websites with either a database or a server as our major target.

2) Scan with a web application vulnerability scanner

Many different web vulnerability scanners are available on the websites. Three different scanners, *X-Scan, Wikto, HP WebInspect,* have been tested, but *Acunetix Web Vulnerability Scanner,* was finally chosen to scan though UBC websites based on the following reasoning: 1) They generate report which contain the details of the scanning result 2) they scan more then just one type of vulnerabilities. An example of some vulnerabilities that are found by Acunetix is showed in *Fig 1*.

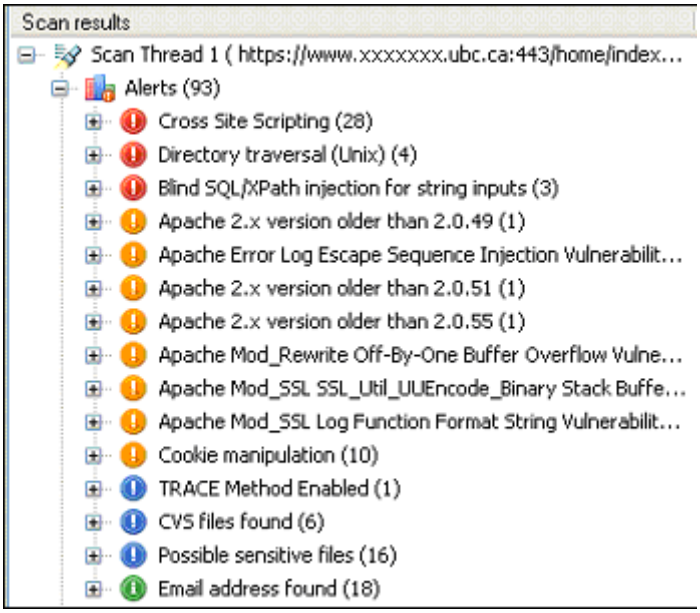

FIG. 1 Vulnerabilities found using Acunetix

.

B. Attack Based on Vulnerabilities Found:

1) SQL injection

An SQL injection attack was conducted to one of the websites found vulnerable. To request all staff information from the database, the following URL has been inputted:

http://xxx.xx.xxx.xxx/directory/directorylisting/activity.cfm?ActID=11111 or 1=1

Note: Exact hyperlink locations and graphics shown in the demonstrations throughout this report have been modified accordingly in order to give UBC IT sufficient amount of time to fix these issues before other attackers perform similar attacks.

When that URL is sent, the corresponding SQL query is executed:

```
Select *
from directory
where ActID = 1111 or 1 =1
```

Since 1 =1 is always a true statement, the database will return every single entry in the directory table that has *ActID* as its attribute.

2) Cross-Site Scripting (XSS)

In the demonstration, the XSS attack was conducted to another website found vulnerable. The attack will modify the page layout by adding a URL link into the webpage. For example, to input a prank URL into E-learning website by using XSS, the attacker edits the URL in the following way:

```
https://www.xxxxx.ubc.ca/home/index.cfm?p=%22%3E%
3Ca%20href=%22https://www.ece.ubc.ca%22%20%3ELo
gin%3C/a%3E%3Cfont%20color=%22white%22%3E
```

or in a more understandable format:

```
https://www.xxxxx.ubc.ca/home/index.cfm?p="><a
href="https://www.ece.ubc.ca" >Login</a><font
color="white">
```

If the user uses the above URL to visit the e-learning website, instead of visiting the e-learning homepage the user will be redirected to a page that has a login link. However, the login link will actually redirect the user to the EECE website instead of the login page, which is shown in *Fig. 2*.

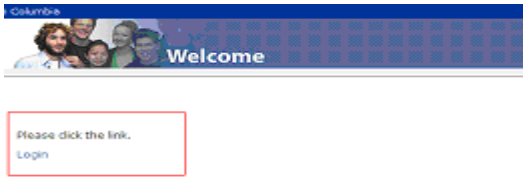

FIG. 2 Sample result using XSS on e-learning website

The reason behind this is that in the prank URL we added html code at the end of the original URL. By using the web vulnerability scanner, we know that the Get value 'p' can be set in the URL. Also, by checking the HTTP response using open source web application, we know what the p value will do in the html code [4].

For example, if we type https://www.xxxxx.ubc.ca/home/index.cfm?p=abc for the URL, in the HTTP response we will get the following html code:

```
<p><br>
    <br>
    Please click the link.</p>
    <a href="https://www.xxxxx.ubc.ca/home/
    DirCMSSiteContent/abc"target="  blank">abc</a>
```

However, since it is not possible to add or edit the existing resources in the website, simply setting the p value will be useless because it is simply trying to access the resource specified by the p value in CMS site content. Therefore we need use the following steps to perform the attack:

a) By setting the p value to "> at the end of the original URL the attacker can end the current attribute setting:

Input in web browser:

```
https://www.xxxxx.ubc.ca/home/index.cfm?p=">
```

Output in HTTP Response:

```
<a href="https://www.xxxxx.ubc.ca/home/
DirCMSSiteContent/">" target="_blank">"></a>
```

b) Since the attacker ended the first href attribute, the link will try to access the resource under the directory /home/DirCMSSiteContent. Although the resource is unavailable, the attacker can add more html code. For example the attacker can make a new href attribute by entering
<a href="https://www.ece.ubc.ca">, which will redirect users to the EECE website [5]:

Input in web browser:

```
<https://www.xxxxx.ubc.ca/home/index.cfm?p=">
<a href="https://www.ece.ubc.ca">
```

Output in HTTP Response:

```
<a href="https://www.xxxxx.ubc.ca/home/
DirCMSSiteContent/"><a href=
      "https://www.ece.ubc.ca" >"target="_blank">
      www.ece.ubc.ca"></a>
```

c) Right now the attacker has a link that will redirect users to other websites. However, because of the extra text it is obvious which website the link will redirect to (see *Fig. 3*).

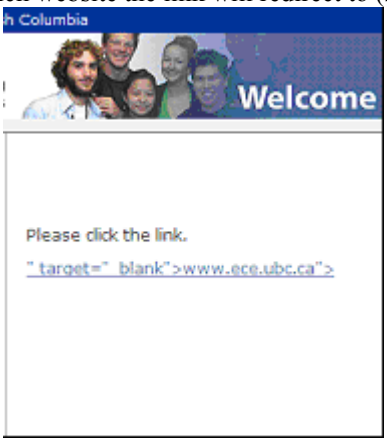

FIG.3 Redirect other user's sample

Therefore for the attacker's purpose of deceiving users we will add more codes to hide the extra text. We will first add a "Login" text to deceive user to think that the link will redirect user to the login page. Second we will add some html code to set the font color of the extra text to the same color as the background color [6]:

Input in web browser:

```
https://www.xxxxx.ubc.ca/home/index.cfm?p="><a
href="https://www.ece.ubc.ca" >Login</a><font
color="white">"
```

Output in HTTP Response:

```
<a
href="https://www.xxxxx.ubc.ca/home/DirCMSSiteCo
ntent/"><a
href="https://www.ece.ubc.ca" >Login</a><font
color="white">" target="_blank">a><font
color="white"></a>
```

And we can see the result in *Fig 2*.

If a user clicks that link, that user will be redirected to www.ece.ubc.ca. In this case, there will be no harm done. However, the destination webpage can be harmful for the user. For example, the attacker can create a similar webpage to the original homepage. The user may be tricked and enter their user/password to the fake page, therefore, the attacker can retrieve the password from the user.

3) Directory Traversal

Directory traversal is an attacking method used by attackers to access restricted Web server files residing outside of the Web server's root directory. In *Fig 4*, the /etc/passwd file on the UNIX server running one of the UBC websites' application is easily exposed to the attacker by running the following command in a web browser URL field:

```
https://www.xxxxx.ubc.ca:443/home/index.cfm?menuClic
ked=1%252F&p=../../../../../../../etc/passwd%00.html
```

where "**../**" are common Unix-like directory traversal characters.

After obtaining the password file containing a list of users, attacker may choose to use tools such as *THC Hydra*[7], a fast network authentication cracker which performs rapid dictionary or brute force attacks against more than 30 protocols, to start hacking user login accounts. Meanwhile, by manipulating the directory level on the WebCT UNIX file system, other files can also be retrieved is read access is granted on the file, which greatly facilitate the attacker to obtain more information about the system and eventually lead to a fully compromised system.

FIG. 4 Result using Directory Traversal Attack

C. Provide Counter Measure

1) SQL injection

a) Remove Culprit Character
Web developer should only allow valid character/word to be inputted. Therefore, character or word such as: --, select, =, which can be use to perform SQL should be removed before it enter to the database.

b) Limit User input's length
The side of the input length should be limit to prevent attacker entering a SQL query message in the dialogue box.

c) Server side Validation
Much current web developing language such as JavaScript can allow web developer to have 1) and 2) countermeasure. However, the attacker can modify the JavaScript code in the webpage and therefore be able to input query message. Therefore, the server side should also validate input.

d) Modify Error Message
Web developer should not allow outside users to see error message/report that is generated by the database [8]. In the example of one of the UBC websites, the webpage return an error message ("OraOLEDB error '80040e57' ORA-01401: inserted value too large for column") on the website which let the attacker know that their message has passed to the database. Therefore, they know which variable can be use for attacking the database.

2) Cross-Site Script

Encode user's Output

Cross-Site Script is based on user sending a malicious code in the script. Therefore, if the content script is encoded, when the content reach to the server, it will not in its executable format. Thus, prevent XSS from happening

3) Directory traversal attacks

a) Ensure the latest version of web server software has been installed, and all hot fixes have been applied.

b) Validate user input by removing meta characters such as (../) from the user input. Ensure only what should be entered in the field will be submitted to the server.

c) Use access control list to limit users' access to specific directories in the web server's file system. Administrator can determine whether a file or folder can be viewed or executed by users, as well as other access rights.

IV. EVALUATION

A. Statistics

According to scanning results (see *Table 1*), most of the target websites have high threat level. In addition, approximately 16% of UBC websites we tested are vulnerable to SQL injection, 35% are vulnerable to Cross Site Script, and only 5% are vulnerable to Directory Traversal. The total numbers of vulnerability for each website classified as High by the scanner are also shown in Table 1.

NOTE: Due to security reasons, the names of the websites are not shown.

Website Scanned	**High Vulnerably Found**			
UBC Websites	SQL Injection	XSS	Directory Traversal	Total No.
01	No	No	No	4
02	No	No	No	0
03	No	Yes	No	14
04	No	No	No	0
05	Yes	Yes	Yes	35
06	No	Yes	No	11
07	No	No	No	2
08	Yes	Yes	No	270
09	Yes	Yes	No	217
10	No	No	No	200
11	No	No	No	2
12	No	Yes	No	64
13	No	No	No	0
14	No	Yes	No	16
15	Yes	No	No	21
16	Yes	Yes	No	361

TABLE.1 Scanning Results

B. Analysis of website scan results

Vulnerability is classified as High when the following criteria apply [9]:

a) Nearly all users of the web application are affected.

b) The vulnerability applies to a standard configuration of the web application.

c) The impact of the vulnerability leads to root or system level privileges or other sever damages to the web application.

Based on these criteria, the following vulnerabilities were found and classified as High among the sixteen UBC websites being scanned.

- Blind SQL/XPath injection for numeric/string inputs (double quotes)
- CRLF injection/HTTP response splitting
- Cross Site Scripting

- Directory traversal (Unix)
- PHP multiple vulnerabilities (Version older than 4.4.1; Entity Encoder Heap Overflow)
- Script source code disclosure
- SQL injection
- Unfiltered Header Injection in Apache 1.3.34/2.0.57/2.2.1
- Zend_Hash_Del_Key_Or_Index Vulnerability

The number of high vulnerabilities for each website were counted and shown in the last column in *Table 1*. Generally speaking, the higher the vulnerability number, the more likely the website will be compromised within a given effort of attacking. Among these websites, UBC bookstore, EECE, and IT services seem to be the most secure. Student Service Centre and Engineering Society websites come in second place in terms of security. The least secure web websites, which are not considered as mission-critical applications, range from Dr. Konstantin Beznosov's website (containing the iBib web application), to the UBC library website, and to the UBC REC website.

C. *Information Security Principles*

Three key concepts form the core principles of information security: confidentiality, integrity and availability, have been compromised in this security analysis of web applications[10].

Confidentiality is the assurance of data privacy. However, in the Directory Traversal example, this principle has been violated since the attacker can easily obtain and read unauthorized files such as /etc/passwd file on the UNIX server via a standard web browser.

Integrity is the assurance of non-alteration of data. Source integrity is compromised when an attacker spoofs its identity and supplies incorrect information to a recipient. In the example, of cross site script, the attacker can retrieve login information from the users therefore, violate the source integrity

Availability is the assurance for authorized users to the timely and reliable data access services. If attackers receive the user login information from users, they can change the password and not allow the correct user to login to e-Learning.

V. CONCLUSION AND FUTURE WORK

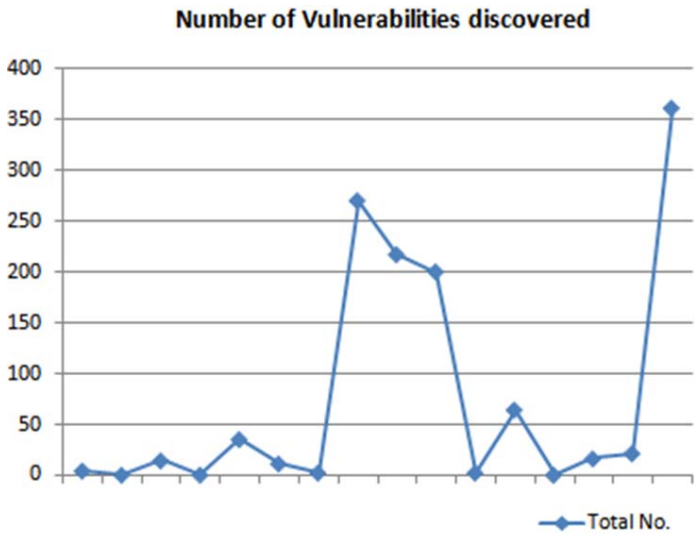

FIG. 5 Number of Vulnerabilities discoverd

As shown from the following *Fig. 4*, the number of vulnerabilities discovered for each website varies. Websites are found to be relatively more secure when one or two of the following conditions applies:

a) Website provides mission-critical online services. For example, the Student Service Centre website has been well developed and deployed since it is a central location where all UBC student information such as transcripts, tuition fee, and registration are maintained. The website must guarantee all data confidentiality, integrity and availability at the highest priority.
b) Websites developed by departments or personals who have broad knowledge in computer security related issues. The EECE, and IT Services websites are excellent examples.

A copy of this report will be forwarded to UBC IT Department and hopefully the vulnerabilities which have been discovered will be addressed by UBC web developers and system administrators in the near future. As the complexity of web applications increases and the attacking methods evolve overtime, defending techniques should also be improved constantly in order to provide worry-free services to system users.

REFERENCES

[1] Acunetix, "70% of websites at immediate risk of being hacked!", February 12, 2007, http://www.acunetix.com/news/security-audit-results.htm

[2] Cgisecurity, "The Cross Site Scripting (XSS) FAQ", August, 2003, http://www.cgisecurity.com/articles/xss-faq.shtml

[3] Imperva Inc, "Directory Traversal", http://www.imperva.com/application_defense_center/glossary/directory_traversal.html, Accessed on November 12, 2007

[4] WebSniffer.net, "View HTTP Request and Response Header", http://web-sniffer.net/, Access on November 12, 2007

[5] Web-Source.net, "HTML Tags / Codes / Web Page Design", http://www.web-source.net/html_codes_chart.htm, Access on November 10, 2007

[6] ComputerHope.com, "HTML color codes and names", http://www.computerhope.com/htmcolor.htm, Access on November 16, 2007

[7] The Hacker's Choice, "THC-Hydra", May 5, 2006, http://freeworld.thc.org/thc-hydra/

[8] P. Sharma, CERT-In, Indian Computer Emergency Response Team,Department of Information Technology, Ministry of Communications and Information Technology, Govt. of India, "SQL Injection Techniques & Countermeasures", July 22, 2005. http://www.cert.org.in/knowledgebase/whitepapers/ciwp-2005-06.pdf

[9] SecurityWarnings, "High-Level Threat", June 16, 2002, http://www.securitywarnings.com/encyclopedia/?id=26

[10] Wikepedia, "Information Security", November 13, 2007, http://en.wikipedia.org/wiki/Information_security