SQL Injection Attack Scanner Using Boyer-Moore String Matching Algorithm

Teh Faradilla Abdul Rahman*, Alya Geogiana Buja, Kamarularifin Abd. Jalil, Fakariah Mohd Ali
Department of Computer, Technology and Network, Universiti Teknologi MARA, Malaysia.

* Corresponding author. Tel.: +60129186831; email: tehfaradilla@salam.uitm.edu.my
Manuscript submitted October 26, 2015; accepted December 26, 2015.
doi: 10.17706/jcp.12.2.183-189

Abstract: In this day and age, the proliferation of fast Internet and advanced technology, have contributed to the development of millions of web applications and the number is going to continue to increase every day. With their various purposes such as business promotions, online shopping, e-learning and social media, it has increased the possibility of privacy violation, information leakage, unauthorized access and some other security aspects. These attacks can be launched by using several methods; one of them is through a Structured Query Language (SQL) injection. Even though there are several approaches that have been introduced to detect SQL injections such as Brute Force and Knuth-Morris-Pratt, there are still some weaknesses encountered. Therefore in this paper, we studied about the SQL injection methodology and detection models for web vulnerabilities. Apart from that, we proposed a detection model to scan SQL injection on the web environment, based on the defined and identified criteria using the Boyer-Moore String Matching Algorithm. From several tests that had been done, the results showed that the proposed model is able to detect vulnerable web applications with the defined criteria of the SQL Injection. In conclusion, this proposed model can be used by web application developer and system admin to secure the application from being attacked and compromised.

Key words: Boyer-Moore, security attack, SQL injection, string matching.

1. Introduction

The beauty of web is that it is on the Internet, which connects billions of people around the world in multiple mediums. Through the web, people started to give their personal information to organizations they subscribed with and this sensitive information could be highly exposed to identity-theft, privacy violation, as well as other cyber threats [1]. These cyber-crime attacks can be launched by using several methods; one of them is through a Structured Query Language (SQL) injection [2]. SQL injection is an attack in which the attacker inserts SQL commands into forms or parameter values [3]. By using SQL injection, the attacker could gain unauthorized access to the web application that is linked with the organizations database and would be able to modify, update or steal the critically important information in the database. Unfortunately, a great number of web developers are unaware of the weaknesses of the security of their web application and this is normal as there are thousands of lines of code which makes it difficult for them to identify the loopholes. Even though an SQL injection is easy to prevent with the help of web vulnerability scanners that exist on the market, most of the scanners have the possibility to produce false negative and false positive results. A false negative is referring to a result which indicates the web application is not vulnerable to the attack when it actually is, whereas the false positive is indicating the web application is vulnerable to the attack while it actually isn't. There is a lot of significant research that has been done to study SQL injection attacks and a number of models have been introduced to prevent this type of attack. However, not a single model can ensure an adequate level of security to protect the web application, which may be because of its diversity and large scope [4]. Our research focuses on the SQL injection methodology, to develop a web vulnerability scanner using the Boyer-Moore String Matching algorithm and to evaluate the efficiency and accuracy of the scanner.

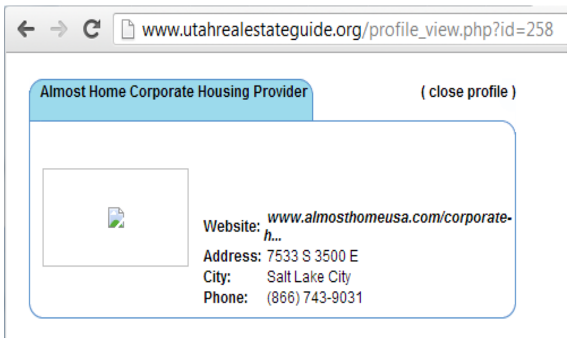
2. SQL Injection

2.1. SQL Injection Attack

An SQL injection is one of the vulnerabilities of web applications, which makes use of the user input field in a web application to create the SQL statement used to penetrate the back-end database. This attack happens when a web application uses user's input data without encryption or proper validation in the query command [4]. As an SQL injection attack can gain unauthorized access to the database, it is considered very critical as the attacker can modify, update, read and delete the database. Even though an SQL injection is easy to protect against, there is still a large numbers of web applications exposed to this type of attack. In order to prevent it, the web application developer should provide sufficient input validation for each field area in the form.

2.2. SQL Injection Methodology

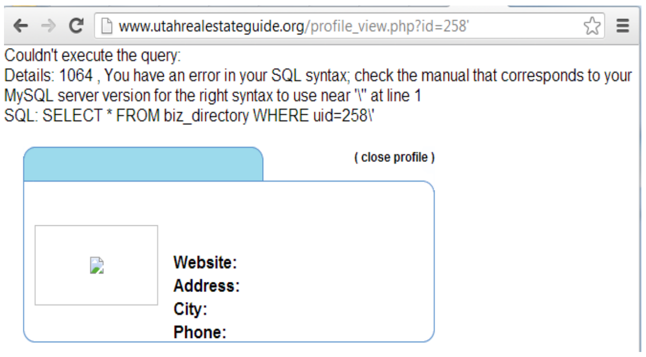
Fig. 1 shows an example of a web application that is vulnerable to an SQL injection whereas Fig. 2 shows a web application that is not vulnerable to an SQL injection. In Fig. 1(a), it shows a web page before the SQL injection attack with the user ID 258 and it looks like everything is alright. In order to determine the web page vulnerable to the SQL injection, a symbol is added after the ID 258. Fig. 1(b) shows the result returned by the application server which contains error messages. The error messages resulted from an incorrect query, with some error details telling specifically at which line the error occurs and even provide the table name of the database. Based on this detailed result, the attacker will then regenerate the query to launch the injection correctly. However, if the error details are hidden, the attacker will have to use other logical queries to get into the database.


www.utahrealestateguide.org/profile_view.php?id=258

Almost Home Corporate Housing Provider (close profile)

Website: www.almosthomeusa.com/corporate-h...
Address: 7533 S 3500 E
City: Salt Lake City
Phone: (866) 743-9031

(a) Before SQL injection attack


www.utahrealestateguide.org/profile_view.php?id=258'

```
Couldn't execute the query:
Details: 1064 , You have an error in your SQL syntax; check the manual that corresponds to your
MySQL server version for the right syntax to use near '\'' at line 1
SQL: SELECT * FROM biz_directory WHERE uid=258\'
```

(close profile)

Website:
Address:
City:
Phone:

(b) After SQL injection attack

Fig. 1. Vulnerable Web page.

3. Exact String Matching Algorithms

Exact string matching is one of the string matching techniques which conduct searching on a string by matching it exactly to the pattern specified. Some of examples of Exact Pattern Matching are Brute Force, Knuth-Morris-Pratt (KMP) and Boyer-Moore.

3.1. Brute-Force

Brute Force string matching algorithm, also known as Naïve string matching algorithm, is a very straightforward approach. This algorithm works by searching and comparing the characters in the pattern from left to right, character by character until match is found [5], [6]. The benefit of the Brute Force algorithm is that it is widely applicable and simple. In addition, it can be used to solve some problems such as searching, string matching and matrix multiplication. On the other hand, the weakness of the Brute Force algorithm is that it rarely produces efficient performance and it is intolerably slow as it compares one character with another. In other words, longer time will be taken to scan a longer string [7].

3.2. Knuth-Morris-Pratt

Knuth-Morris-Pratt is another example of a string matching algorithm that is similar to the brute force algorithm, moving from left to right, except it has a pre-processing phase where the window can be shifted more than one [5]. As a result, this algorithm can shorten the searching time. The Knuth Morris-Pratt algorithm uses a pattern to determine how far to skip, for example if the first three characters of the pattern and text are a match, the next search will make the window shift 3, as there are 3 character that already matched. The benefit of this algorithm is that it has an optimal running time; the algorithm never has to reverse back in the searched text. However, the weakness of this algorithm is that it is not efficient enough in dealing with a large number of characters as a higher possibility of mismatch could occur [7].

3.3. Boyer-Moore

The Boyer-Moore string matching algorithm is usually used for searching large amounts of data in a short period of time such as searching for virus patterns and databases [5], [6]. This algorithm is one of the fastest string searching algorithms as it searches the string for the pattern but not each character of the string. As the pattern length increases, the algorithm will run faster. Boyer-Moore arranges the text to be compared and the keyword so that the keyword can be checked from left to right along the text. The check begins with the last character of the keyword and ending with the first [8]. In order to determine the number of shifts required to slide the window on the text, a Boyer-Moore good suffix and a Boyer-Moore bad character were used.

4. The Proposed Model

In order to detect some pattern, especially in the URL, the scan process should start from right to left. In addition, the algorithm to be selected must be able scan the string using a pattern, in order to save time in searching. Therefore, the proposed model is implementing the Boyer-Moore algorithm, since it scans the string using a pattern, from right to left until the pattern is found and it is proved to be the most useful for scanning for vulnerabilities as well as for viruses. Brute Force and Knuth-Morris-Pratt were not selected because these algorithms do not fulfil the requirements needed. Moreover, these two algorithms took a longer time to complete the search.

4.1. Overview of the Proposed Model

Fig. 2 shows the flow chart of how the proposed model will work. The module will scan for the attribute or criteria of the web application vulnerabilities. Firstly, the user will enter an input string or files. The string can be the URL of the site or web application, while the file can be the stream of source code for each web application. The next step is to launch the SQL Injection Attack Detection Module and perform each of their functions.

Based on the chosen string matching algorithm, each of the string or input files will be scanned for the defined attributes of the SQL Injection attack detection. Then the input is passed to the four panel reference detection modules which are made up of crawler, parameter testing, exploit and report. The crawler will move from one page to another page of the website for parameter identification. The captured parameters are then tested to determine whether it is vulnerable to an SQL injection attack, and, if yes, the exploit panel is recommended. This panel will use the same parameters to penetrate the database and will be able to show the names of the table columns and rows affected. Finally, the report panel will generate a report which consists of a description about vulnerabilities found, tested pages and solutions to the problems. In addition, the proposed model was evaluated in terms of efficiency; the total time taken to crawl, and accuracy; the pertinence to detect vulnerable web applications.

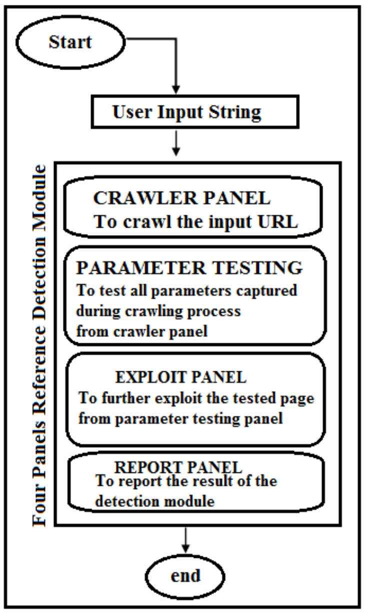

Start
User Input String
Four Panels Reference Detection Module
CRAWLER PANEL
To crawl the input URL
PARAMETER TESTING
To test all parameters captured during crawling process from crawler panel
EXPLOIT PANEL
To further exploit the tested page from parameter testing panel
REPORT PANEL
To report the result of the detection module
end

Fig. 2. SQL injection attack detection model.

5. Findings and Discussions

5.1. SDR Web Vulnerability Scanner

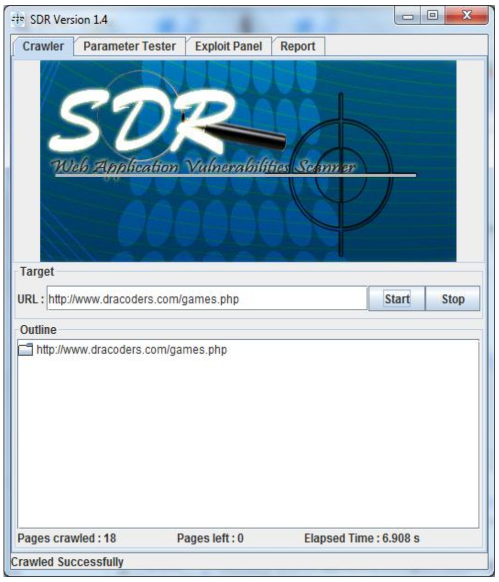
By using the proposed model, an application to scan web vulnerabilities called Scan-Detect-Report (SDR) has been developed and tested. As shown in Fig. 3, at the top of the SDR scanner window, there are four panel reference detection modules as stated earlier in the previous section. To begin the crawling process, a user needs to enter the target URL in the URL text field and click the Start button. In this example, the total of successfully crawled pages is 18 and the time taken is 6.908.

Next, the process of searching took place at the Parameter Tester Panel to test the parameters that are captured during the crawling process in the Crawler Panel.

For example in Fig. 4, the URL: http://www.dracoders.com/games.php?id=22.

There are two parameters found in the URL. To use the parameter tester, the user just needs to choose the parameter to be tested and click the Initiate Testing button or Stop SQL Injections to stop the testing mid-way.

This panel is recommended only if the page is proved to be vulnerable so that it can further exploit the tested page and to initiate an SQL injection attack on the page. The URL, vulnerable parameter and Default value will be automatically entered by SDR as we are using the Parameter Tester Panel. In the option panel, check all of the three boxes to find information about the database, to identify table or columns name (Bruteforce table/col names if necessary) that exist in the website and to get information about the database user and the database version (Retrieve user and database information). All the results are displayed in the Results panel as shown in Fig. 5.



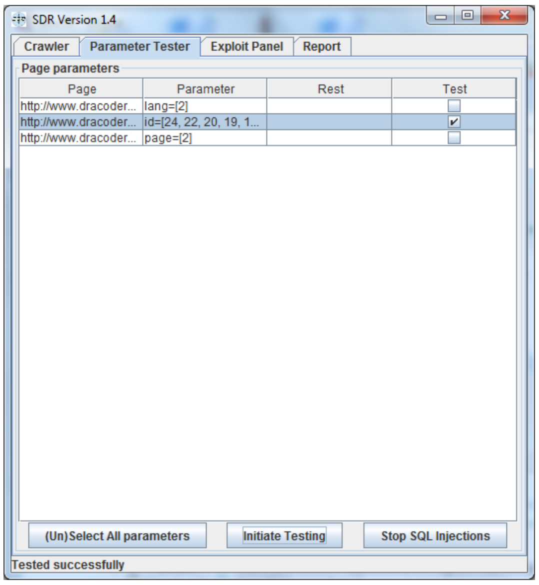
URL : http://www.dracoders.com/games.php

http://www.dracoders.com/games.php

Pages crawled : 18 Pages left : 0 Elapsed Time : 6.908 s

Crawled Successfully

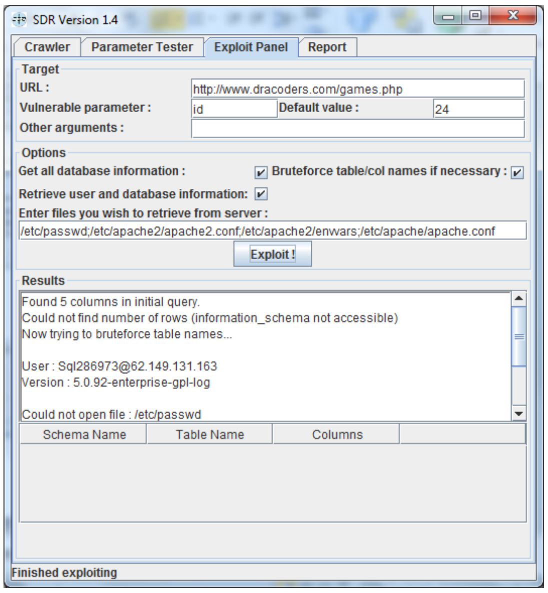
Fig. 3. Crawler panel.



Page	Parameter	Rest	Test
http://www.dracoder...	lang=[2]		☐
http://www.dracoder...	id=[24, 22, 20, 19, 1...		✓
http://www.dracoder...	page=[2]		☐

Tested successfully

Fig. 4. Parameter tester panel.



URL : http://www.dracoders.com/games.php

Vulnerable parameter : id Default value : 24

Other arguments :

Get all database information : ✓ Bruteforce table/col names if necessary : ✓

Retrieve user and database information: ✓

Enter files you wish to retrieve from server :

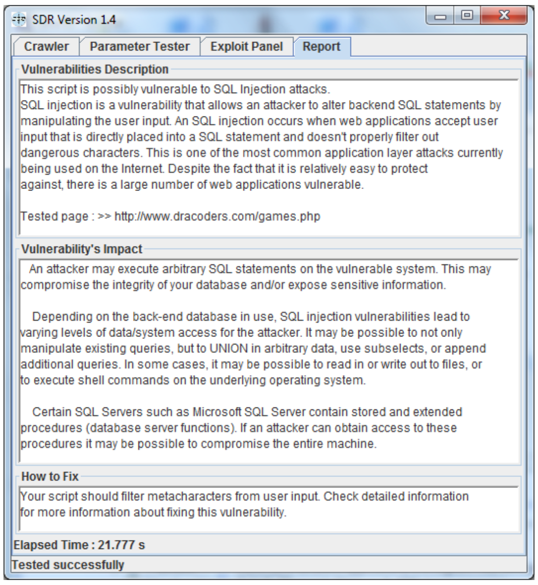
/etc/passwd;/etc/apache2/apache2.conf;/etc/apache2/envvars;/etc/apache/apache.conf

```
Found 5 columns in initial query.
Could not find number of rows (information_schema not accessible)
Now trying to bruteforce table names...

User : Sql286973@62.149.131.163
Version : 5.0.92-enterprise-gpl-log

Could not open file : /etc/passwd
```

Schema Name	Table Name	Columns

Finished exploiting

Fig. 5. Exploit panel.



Vulnerabilities Description

This script is possibly vulnerable to SQL Injection attacks.
SQL injection is a vulnerability that allows an attacker to alter backend SQL statements by manipulating the user input. An SQL injection occurs when web applications accept user input that is directly placed into a SQL statement and doesn't properly filter out dangerous characters. This is one of the most common application layer attacks currently being used on the Internet. Despite the fact that it is relatively easy to protect against, there is a large number of web applications vulnerable.

Tested page : >> http://www.dracoders.com/games.php

Vulnerability's Impact

An attacker may execute arbitrary SQL statements on the vulnerable system. This may compromise the integrity of your database and/or expose sensitive information.

Depending on the back-end database in use, SQL injection vulnerabilities lead to varying levels of data/system access for the attacker. It may be possible to not only manipulate existing queries, but to UNION in arbitrary data, use subselects, or append additional queries. In some cases, it may be possible to read in or write out to files, or to execute shell commands on the underlying operating system.

Certain SQL Servers such as Microsoft SQL Server contain stored and extended procedures (database server functions). If an attacker can obtain access to these procedures it may be possible to compromise the entire machine.

How to Fix

Your script should filter metacharacters from user input. Check detailed information for more information about fixing this vulnerability.

Elapsed Time : 21.777 s

Tested successfully

Fig. 6. Report panel.

Lastly, Fig. 6 shows the report panel which is the result from the Parameter Tester panel. This panel provides a brief description about the vulnerability found and the tested page. In addition, the Vulnerability's Impact section explains how the vulnerability can affect the website if it is left vulnerable. Some suggestions to fix the vulnerability are given in the How to Fix section. In this example, the elapsed time label is 21.777s, the time taken to finish the crawling process and the parameter tester process.

5.2. Efficiency and Accuracy

The proposed model has been evaluated in terms of efficiency and accuracy. The efficiency of the developed model is measured based on the total time taken for the process of crawling and testing the parameter while the accuracy of the proposed model is defined as the correctness of the proposed model to detect the vulnerable web application successfully based on the identified and defined criteria of the SQL

Injection Attack. Table 1 and Fig. 7 show the result of the efficiency based on the tested URL. The results obtained shows that most of the secured web application took a longer time to crawl even though it contains a small number of pages. In Fig. 7, it can be said that the time taken for the crawler to completely crawl all pages was not dependent on the number of pages a web application has, instead it was dependent on the speed of the internet connection. If the internet connection is slow, the time taken to complete the crawl process will be longer.

Through several tests, 10 URLs of web applications had been examined and the proposed model has successfully detected the first eight URLs as the vulnerable web applications while the last two URLs were detected as invulnerable to SQL Injection attacks.

Table 1. Total Time Taken for Detecting SQL Injection Pattern of Attackable Styles

	URL	No of page	Crawl (s)	Parameter Testing (s)
1	http://www.dracoders.com/games.php	18	6.99	14.78
2	http://senesco.com/newsitem.php	117	44.61	24.05
3	http://www.ath-elite.com.au/trainers.php	22	15.27	22.96
4	http://www.vivactiv.ru/trainings/trainers.php	105	32.42	27.24
5	http://www.pushingpetals.com/buy.php	4	10.19	12.90
6	http://www.suffolkconstruction.com/staffView.php	83	105.41	25.843
7	http://www.vortexgym.com/trainers.php	31	11.37	18.82
8	http://www.ureka-sg.com/trainers.php	93	27.71	18.94
9	http://www.centos.org/modules/tinycontent/index.php	37	64.40	38.19
10	http://www.rspba.org/html/newsdetail.php	141	145.11	36.06

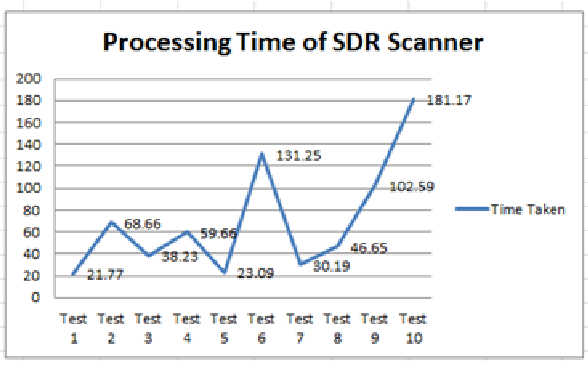


Fig. 7. The efficiency of the proposed model.

6. Conclusion

The proposed model has been able to detect the vulnerable web application with the defined criteria of the SQL Injection Attack. Moreover, the implementation of the Boyer Moore string matching algorithm has assisted the proposed model to be more efficient and accurate. In this work, the accuracy of the proposed model can be improved by identifying and adding a few parameters into the "Parameter Testing Panel". In addition, the proposed model can help the web application developer or administrator to take any further action to secure their application from being hacked or attacked by the unethical person outside the network by exploiting and compromising the vulnerabilities of the web application towards an SQL Injection attack.

Acknowledgment

The authors would like to thank Universiti Teknologi MARA for supporting this work. This work was also supported by Ministry of Higher Education Malaysia under Research Acculturation Grant Scheme 2013.

References

[1] Trend Micro. *Web Application Vulnerabilities How's Your Business ON The Web?* From http://www.trendmicro.com/cloud-content/us/pdfs/business/tlp_web_application_vulnerabilities.pdf

[2] Anjali, S. K., & Kulkarnai, R. B. (2012). Web vulnerability detection and security mechanism. *International Journal of Soft Computing and Engineering, 2(4)*, 2231-2307.

[3] Benoist, E. (2012). *Advanced Web Technology 9) OWASP Top 10 Vulnerabilities & Cross Site Scripting.*

[4] Namdev, M., Hasan, F., & Shrivastav, G. (2012). A novel approach for SQL inection prevention using hashing & encryption (SQL-EBCP). *International Journal of Computer Science and Information Technologies*, *3(5)*, 4981-4987.

[5] Charras, C., & Lecroq, T. (2004). *Handbook of Exact String-Matching Algoritms.* King's College London Publications.

[6] Singla, N., & Garg, D. (2012). String matching algorithms and their applicability in various applications. *International Journal of Soft Computing and Engineering*, 2231-2307.

[7] Wahlstrom, S. (2004). *Evaluation of String Searching Algorithm.*

[8] Coit, J. C., Staniford, S., & McAlemey, J. (2001). *Towards Faster String Matching for Intrusion Detection or Exceeding the Speed of Snor.* From http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=932231

Teh Faradilla Abdul Rahman is currently a lecturer of foundation computing at Center of Foundation Studies, Universiti Teknologi MARA, Malaysia. She received her B.Sc. in netcentric computing from Universiti Teknologi MARA, Shah Alam Malaysia, and a master in education (mathematics) from Universiti Teknologi Malaysia, Johor, Malaysia. She involved in a number of research works and presented several papers at national and international conferences. She also involved with charity work such as Mangroves Conservation Program at Bagan Sungai Kajang, Tanjung Karang, Malaysia, which was organized by Centre of Foundation Studies, Universiti Teknologi MARA, Malaysia. A Charity Program: One Foster Kid for One Family. Orphanage Teratak Nur Barakah, Shah Alam, Malaysia, which was organized by Universiti Teknologi MARA and Modenas Riders Group Malaysia. And Science Fun Day Program at Bandar Puncak Alam Secondary School, Malaysia, which was organized by Centre of Foundation Studies, Universiti Teknologi MARA, Malaysia.

Alya Geogiana Buja received her bachelor's and master's degree in Universiti Teknologi MARA in 2011. She is a Ph.D. candidate in Universiti Teknikal Malaysia Melaka since September 2014. Her current research interests include information retrieval and network and information security.

Kamarularifin Abd Jalil is a lecturer at the Faculty of Computer and Mathematical Sciences at the Universiti Teknologi MARA, Malaysia, since 1997. He has a Ph.D. degree in the Department of Electronic and Electrical Engineering from the University of Strathclyde, U.K., in 2008, an MSc degree in Information Technology for Manufacture in 1997 from University of Coventry, U.K. and a BSc degree in computer science in 1995 from Universiti Teknologi MARA. His research interests are in computer networking area, including mobile networks & protocols.

Fakariah Hani Mohd Ali obtained her PhD of security in computing from Universiti Putra Malaysia. She is a Senior Lecturer at the Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Malaysia. She is a member of the Malaysian Society Cryptology Research (MSCR). Her research interest are cryptography, network security and digital forensics.