IEIE Transactions on Smart Processing and Computing, vol. 3, no. 5, October 2014
http://dx.doi.org/10.5573/IEIESPC.2014.3.5.319

Improving Malicious Web Code Classification with Sequence by Machine Learning

Incheon Paik

School of Computer Science and Engineering, University of Aizu / Aizu-Wakamatsu City, Fukushima, Japan
paikic@u-aizu.ac.jp

***** Corresponding Author: Incheon Paik

Received January 15, 2014; Revised March 17, 2014; Accepted July 28, 2014; Published October 31, 2014

** Extended from a Conference: Preliminary results of this paper were presented at the IEEE ISCE, 2014. This present paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

***Abstract*:** Web applications make life more convenient. Many web applications have several kinds of user input (e.g. personal information, a user's comment of commercial goods, etc.) for the activities. On the other hand, there are a range of vulnerabilities in the input functions of Web applications. Malicious actions can be attempted using the free accessibility of many web applications. Attacks by the exploitation of these input vulnerabilities can be achieved by injecting malicious web code; it enables one to perform a variety of illegal actions, such as SQL Injection Attacks (SQLIAs) and Cross Site Scripting (XSS). These actions come down to theft, replacing personal information, or phishing. The existing solutions use a parser for the code, are limited to fixed and very small patterns, and are difficult to adapt to variations. A machine learning method can give leverage to cover a far broader range of malicious web code and is easy to adapt to variations and changes. Therefore, this paper suggests the adaptable classification of malicious web code by machine learning approaches for detecting the exploitation user inputs. The approach usually identifies the "looks-like malicious" code for real malicious code. More detailed classification using sequence information is also introduced. The precision for the "looks-like malicious code" is 99% and for the precise classification with sequence is 90%.

***Keywords*:** Malicious web code, Machine learning, Classification, Sequence information, Web security

1. Introduction

Currently, many web applications have security problems in the application layer, which can be exploited by crackers hackers. Hackers typically attack web applications by injecting or changing a query of the HTTP request or response.

They normally pay attention to two types of malicious code; SQLIAs and XSS [1-3]. The purpose of SQLIAs is to control databases in a web application illegally. SQLIAs enable the theft or change of important data (e.g. addresses, credit card numbers, etc.) in databases. The purpose of XSS is to inject malicious scripts in a web application. When the users access a web application with an injected XSS, the injected scripts perform illegal actions on the user's web browser (e.g. phishing, and stealing cookies, etc.).

The existing approaches can detect registered patterns like existing malicious web code with high accuracy. The models used to detect malicious code can be divided into two types, positive security model and negative security model. The positive security model registers many patterns of non-malicious web code, and detects a query as a malicious web code when the query does not match up with registered patterns. In contrast, the negative security model registers many patterns of malicious web code, and it detects a query as malicious web code when the query matches up with the registered patterns.

The current methods to detect malicious code analyze the user input using a parser, and confirm a match limited to fixed and very small patterns that are modeled by reference to existing malicious web code. On the other hand, there are new malicious web codes that can be created deliberately over the registered patterns in existing approaches. Therefore, these are difficult to adapt to a changed environment. A machine Learning method can

overcome this problem because it can provide leverage for a broader range of malicious web code and it is easy to adapt to variations and changes.

In this study, a novel classifier of malicious web code for an evaluation of the approach for code classification. Several types of machine learning algorithms have been created. Therefore, it is important to investigate the aptitude of each machine learning algorithms. As a result, each classifier uses different machine learning algorithms to find the appropriate method for classifying of malicious code. On the other hand, the algorithm can classify "just looks-like malicious" code because the feature vector does not include sequence information. Therefore, another study improves the previous classification system so that it can classify looks-like malicious code as normal code exactly. The improved classification system detects malicious web codes by learning the sequence of the tokens included in codes to recognize the syntax of the codes.

This paper explains a background of malicious web code regarding SQLIAs in Section 2. Section 3 explains the related work. Section 4 provides details of the proposed approach. The experiment and evaluation of this approach are explained in Section 5, and Section 6 concludes the study.

2. Background of Malicious Web Code (SQLIAs)

Many web applications use the Relational Data Base (RDB), which is a popular method for managing large volumes of data. The RDBs are generally controlled by statements written in Structured Query Language (SQL). When web applications require data from the user, it requests the data to RDB by input SQL, which is the associated hardcode and user input. Generally, web applications suppose that the user input is constructed by only just text data (e.g. queries for search engines, password, comments on articles, etc.). On the other hand, it also enables the input of malicious SQL statements for rewriting SQL statements against the intention of the managers and developers of a web application. These attacks are called SQLIAs, which can be divided into several types.

The first example of SQLIAs is an attack for an unauthorized login by converting the WHERE statement to TRUE. The WHERE statement is a part of the SQL statements, which are used to describe the conditions for extracting data as follows:

SELECT * FROM *user_table* WHERE *id*='*user_id*' AND *password*='*pass*'; (1)

This SQL statement (1) requests that data be extracted from *user_table* whose *id* and *password* agree with *user_id* and *pass* for an authorizing login. If a condition of the WHERE statement is always TRUE, it extracts all the data in the table. Therefore, it allows access to all data in a table through the use of malicious SQL statements that can rewrite the result of the WHERE statements as follows:

'OR 1=1; -- (2)

For example, this malicious SQL statement (2) is injected into the SQL statement (1) as follows:

SELECT * FROM *user_table* WHERE *id*='' OR 1=1; -- ' AND *password*='*pass*'; (3)

The WHERE statement in this injected SQL statement (3) is always TRUE, because 1=1 is always TRUE. Therefore, this injected SQL statement (3) extracts all the data from *user_table* despite the incorrect *id* and *password*, so it becomes an unauthorized login. Several unnecessary symbols (' AND *password*='*pass*';) are ignored by the line comment. The words after the symbol of the line comment (--) are treated as comments.

Another example of SQLIAs is the attack for stealing data illegally by the UNION statement. The UNION statement is a part of the SQL statements, which are used to combine the results of two SELECT statements as follows:

SELECT * FROM *products_table* WHERE *name*='*product_name*'; (4)

This SQL statement (4) requests to extract data from *products_table*, whose *name* agrees with *product_name* for searching the products. In this case, SQLIAs come into effect by injecting the following code:

'UNION SELECT * FROM *user_table*; -- (5)

This malicious SQL statement (5) is injected to the SQL statement (4) as follows:

SELECT * FROM *products_table* WHERE *name*=' 'UNION SELECT * FROM *user_table*; -- '; (6)

This SQL query (6) combines the results of two SELECT statements; "SELECT * FROM *products_table* WHERE *name*=' '" and "SELECT * FROM *user_table*;" by the UNION statement. If the web applications respond to the search results using the SQL statement like (6), it becomes available to display data in *user_table* with the search results by the injecting a malicious SQL statement (5).

3. Related Work

The most popular method for detecting malicious web code is Pattern Match. This method classifies malicious web code when the user input matches up with registered patterns. AMNESIA is a model-based technique like Pattern Match for classifying SQLIAs. AMNESIA analyzes a web application statistically for extracting several positions, which sends SQL queries to the Databases and generates models of a normal SQL query. AMNESIA detects queries as SQLIAs if the sent SQL queries do not match up with the generated models.

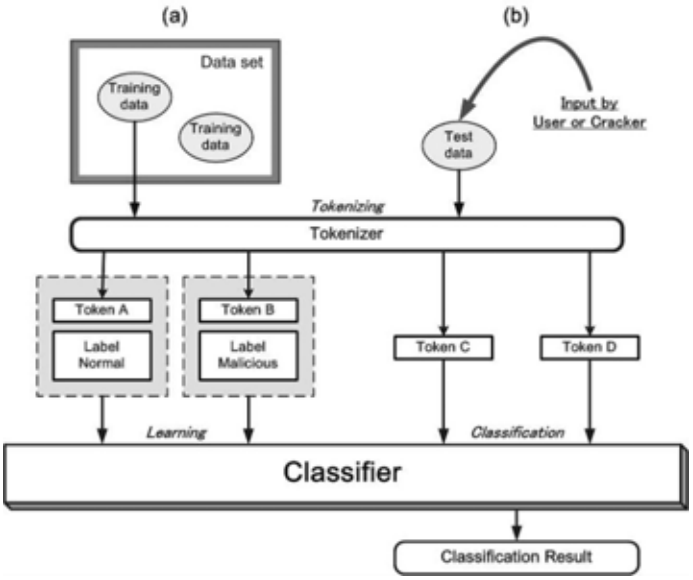

Fig. 1. Flows of two classification processes (a) the learning process, (b) the classification process.

Dynamic Tainting [7] is also a similar Pattern Match. Dynamic Tainting treats all user inputs or processed data of the user input as tainted data. When tainted data is used for unsafe processes (e.g. sending SQL queries, file access), Dynamic Tainting detects tainted data as malicious web code if the tainted data does not match up with the registered patterns.

Malicious web code are also detected by a Parsing Approach. This method parses the user input along a particular grammar. For example, SQL queries are parsed along the grammar of SQL. SQL Check is one Parsing Approach. SQL Check analyzes the syntax of SQL queries by the SQL parser and each user input is then taken as an argument query. If the argument queries do not match up with the registered data type, SQL Check detects it as malicious web code. SQL Guard [8] is also one of the Parsing Approach, which is similar to SQL Check. SQL Guard extracts some positions that send SQL queries statically like AMNESIA, and it parses the normal SQL queries of each point. When SQL queries are sent, SQL Guards it dynamically. If a dynamically analyzed syntax does not match the corresponding statically analyzed syntax, SQL Guard detects it as malicious code.

Another method uses random numerals or strings of characters for detection. Random numerals or strings of characters are used to mark the keywords written in hardcode. SQL rand [9] is included in this method. When SQL queries are sent, SQL rand appends the random numerals to a SQL standard keyword written in hardcode dynamically. The SQL standard keywords in the malicious input are not appended by it. When SQL queries, which are appended random numerals, reach a database server, SQL rand removes its random numerals and executes queries. If there are SQL standard keywords that are not appended random numerals in SQL queries, SQL rand detects it as malicious web code.

4. Proposed Solution

This novel approach enables the detection of new malicious web code flexibly by utilizing machine learning algorithms. Machine learning algorithms have a learning function that is close to human beings. New malicious web code set is created over the patterns of existing malicious web code, so it is difficult to define a border between malicious web code and non-malicious web code. On the other hand, machine learning algorithms can classify new malicious web code either a malicious web code class or non-malicious web code class with high accuracy. Because machine learning algorithms learn some features of malicious web code from training datasets, which include many malicious web codes and non-malicious web codes it derives a flexible criterion between malicious web code and non-malicious web code. In addition, the accuracy of machine learning algorithms is affected by its defined features. Generally, machine learning algorithms for the classification of texts uses terms separated by blanks as features. The present approach, however, realizes higher accuracy than existing machine learning approaches by extracting some tokens that consist of specific terms of malicious web code as features. In this study, classifiers are created to classify malicious web code using machine learning algorithms.

4.1 Classifier

The proposed classifiers classify a user input character string as either malicious code or non-malicious code. This approach requires two processes for classifying input; learning process and classification process. Fig. 1 illustrates the flow of the processes. In the learning process, it must determine the criterion for classifying input. The classifier extracts the features of malicious web code or non-malicious web code from each training data as a feature vector, and determines the criterion from feature vectors and its label using a machine learning algorithm. The labels describe the class belonging to the training data

after tokenizing it. In the classification process, the classifier classifies the user input based on the classification criterion that is determined in the learning process by the machine learning algorithm.

4.2 Generation of Feature vector

A feature vector is the abstraction format of the features of data as documents, which include both training data and input. Many machine learning algorithms for classification to several classes need to extract the features from documents as a format for learning and classification. A feature vector is described as follows:

$$(w_1,\ w_2,\ w_3,\ \ldots). \tag{7}$$

where w_x is a weight of each term. Each term is extracted from the documents by any extraction method, and the weights of each term are calculated using any evaluation method, such as TF-IDF method [10]. This paper proposes two extraction methods, Blank Separation Method and Tokenizing Method.

Blank Separation Method: Documents generally consist of a combination of many terms that are separated by blank space. The Blank Separation Method extracts terms with respect to each blank space. The number of each term is used to calculate the weight, e.g., when the separate the following document is separated:

$$\text{`OR 3=3; --} \tag{8}$$

The Blank Separation Method separates this document as follows:

Table 1. Example of Blank Separation Method.

Name	Term Frequency
‘	1
OR	1
1	2
=	1
;	1
--	1

Tokenizing Method: Malicious web code has some tokens that describe the feature of malicious web code. The Tokenizing Method extracts the terms with respect to each token that correspond to each type of malicious code by an evaluation repeatedly to determine a better token as follows:

Table 2. Tokens of SQLIAs.

Name	Corresponding Terms and Symbols
Line Comment	-- #
Comment	/* */
Operator	<> <=> >= <= == = != << >> < > | & - + % ^ ?
Logical Operator	NOT AND OR XOR ! && ||
Punctuation	[ ] ( ) , ; . ‘ “
Literal	“*any string*” ‘*any string*’ `*any string*`
Table Handling	SELECT INSERT UPDATE DELETE CREATE DROP ALTER RENAME
Numeral	*Numeral included in* +, -, e, *and* R

4.2 Generation of Feature vector with Sequence

The system generates the sequence vectors from the annotated token. The system can deal with the sequence of tokens included in the code by quantifying them as sequence vector. The expression below is to generate the sequence vectors. The ordered set, U, is a set of annotated tokens (T_1, T_2, ..., T_m). The system generates an ordered subset S from the ordered set U as the expression where size of the window (WS) is the number of annotated tokens that are included in S. Next, the system generates a sequence vector from all ordered subsets (S_1, S_2, ..S_j), where the element of the vector is the frequency of the same ordered subsets.

$$U = (T_1, T_2, ..., T_m) \tag{9}$$

$$Sj = \{T_k | \ j <= k <= j + WS,\ k \in N,\ T_k \subset U\} \tag{10}$$

$$(j = 1, 2, 3, ..., size(U) - WS + 1) \tag{11}$$

SQL injection has features on the first and last token. The system was made to learn the first and last tokens independently to improve performance. This means that the system adds a mark element when generating a code sequence.

5. Evaluation

A classifier was implemented and evaluated for SQLIAs, which can use the TF-IDF method for weight calculations, and a three machine learning approach among the Support Vector Machine (SVM), Naïve-Bayes, and k-Nearest Neighbor Algorithm. In addition, some types of kernel functions were evaluated in SVM, linear kernel, polynomial kernel, and Gaussian kernel. In addition, the evaluation application was implemented for an evaluation of two classifiers. In the learning process, this evaluation application reads a training dataset from text files and puts each data into the learning methods of each classifier. The classifiers generate the feature vectors from the received data using the TF-IDF method and learns it according to each machine learning method. In the classification process, this evaluation application also reads the test dataset from text files and puts each data into the classification methods of each classifier. The classifiers generate the feature vector from the received data and classifies. The classification results of each machine learning approach are analyzed according to the precision, and recall [12] in the evaluation application.

Several dataset were sued for the evaluation as follows:

Table 3. Data Sets for SQLIAs Evaluation.

	Training Data	Test Data
Normal Code	347	137
Malicious Code	348	221

These dataset were collected with the cooperation of experts in attacks on web applications.

Table 4. Evaluation Result of SQLIAs.

	Accuracy	Precision	Recall
SVM (Linear Kernel)	98.60%	0.977	1.00
SVM (Polynomial Kernel)	98.60%	0.977	1.00
SVM (Gaussian Kernel)	99.16%	0.986	1.00
Naïve-Bayes	98.88%	0.986	0.995
k-Nearest Neighbor Algorithm	98.60%	0.991	0.986

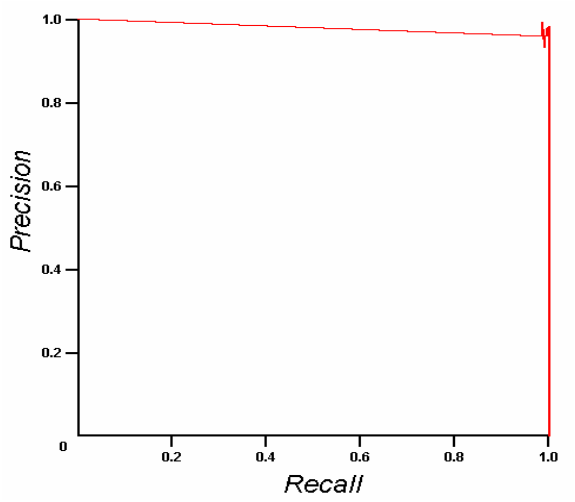

Fig. 2. Recall-Precision graph of SVM with the Gaussian Kernel in SQLIAs.

Fig. 2 shows the evaluation results described by Recall-Precision graphs. In addition, Table 4 lists the evaluation results described by precision and recall. The maximum accuracy of SVM is the highest for SQLIAs detection. In addition, both precision and recall are high. The performance by Naïve-Bayes and k-Nearest Neighbor algorithm is lower than that by SVM. The maximum accuracy of the Gaussian kernel is the highest of the three kernel functions. These results suggest that the classifier by SVM using Gaussian kernel is the most appropriate classifier in this evaluation.

The code set was also used to evaluate the performance of classifying malicious and "looks-like malicious" code. This code set is called the improved code set. In addition, original codes set of previous research was evaluated for comparison.

In the evaluation, two types of improved classification system was dealt with: with code sequence only, and code sequence with an added mark element. Furthermore, each type of system uses five different window sizes or five different combined window sizes. To validate the result, a two folded cross validation was used for each type of system [6]. Figs. 3 and 4 shows the results of the improved code set without a mark element and with a mark element, respectively. The feature with the mark element showed better performance than the case without the mark element. The classification with the improved code set and mark

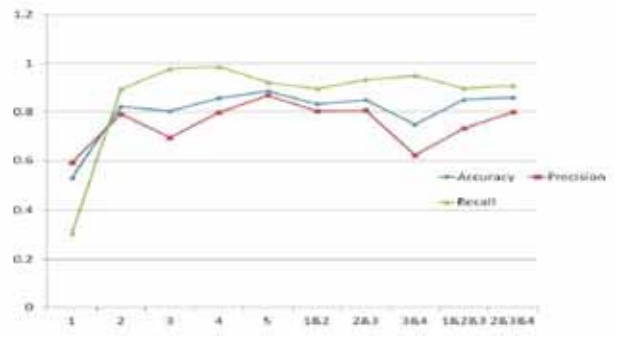

Fig. 3. Evaluation Using the Improved Code Set.

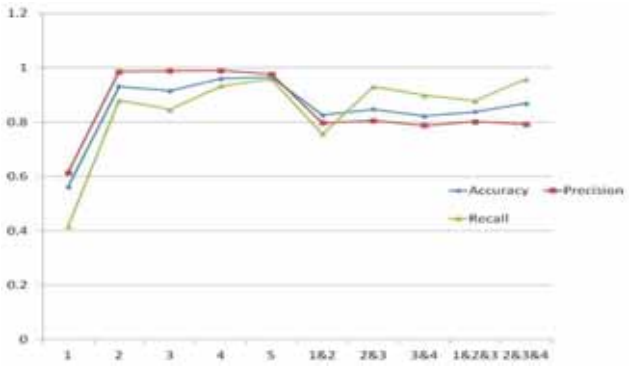

Fig. 4. Evaluation Using the Improved Code Set with a Mark Element.

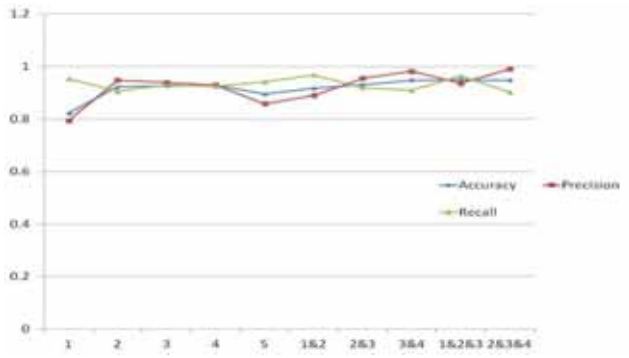

Fig. 5. Evaluation Using the Original Code Set with a Mark Element.

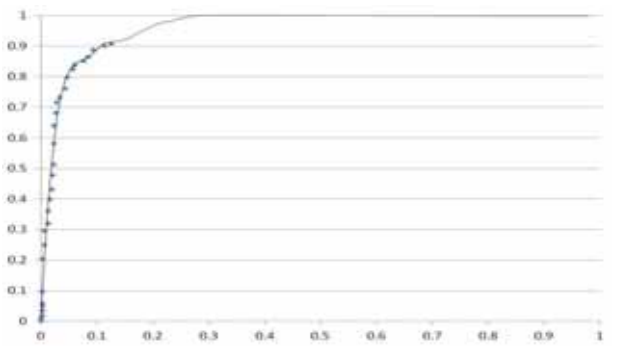

Fig. 6. ROC Curve of the Classification System Using Window Size 2 and 3 with Original Code Set.

element showed better performance than the existing approach. The Receiver Operator Characteristic (ROC) curve was used to determine the optimal cut-off point with the SVM's output value [5]. The area under curve of the Fig. 6 was 0.959, which shows high accuracy.

5. Conclusion

Many web applications have vulnerability to malicious code, such as SQLIAs and XSS. These vulnerabilities allow the theft or defacing of important data by malicious web code. The existing provision cannot deal with new malicious web code flexibly. The proposed approach solves the problem by using the ability of characterizing and training by machine learning. An evaluation of the proposed approach indicated high precision for SQLIA (99.16%) using SVM with a Gaussian kernel.

The performance of the improved classification system was also evaluated using code sequence by SVM. As a result, for an improved codes set, the system with the added mark element and window size 3 showed high-performance in all experimental combinations. On the other hand, for the original codes set, the system using window size 2 and 3 showed the highest performance.

In several data sets for training and testing of the experiment, the accuracy varied from 0.8 to 0.95, and it can be considered as satisfying the goal. The system however, is unstable because the window size to have the highest accuracy was changed according to the code set. In future studies, a method will be developed to find the stable window size to optimize the performance.

References

[1] W.G.Halfond and A.Orso, "AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks," Proc. IEEE and ACM International Conference on Automatic Software Engineering (ASE 2005), Long Beach, CA, USA, Nov 2005. Article (CrossRef Link)

[2] The Open Web Application Security Project (OWASP). The Ten Most Critical Web Application Security Risks 2010. https://www.owasp.org/index.php/Top_10_2010-Main

[3] Z.Su and G.Wassermann, "The Essence of Command Injection Attacks in Web Applications," The 33rd Annual Symposium on Principles of Programming Language (POPL 2006), Jan 2006. Article (CrossRef Link)

[4] V.Haldar, D.Chandra, and M.Franz, "Dynamic Taint Propagation for Java," Proc. 21st Annual Computer Security Applications Conference, Dec 2005. Article (CrossRef Link)

[5] G.T.Buehrer, B.W.Weide, and P.A.G.Sivilotti, "Using Parse Tree Validation to Prevent SQL Injection Attacks," International Workshop on Software Engineering and Middleware (SEM), 2005. Article (CrossRef Link)

[6] S.W.Boyd and A.D.keromytis, "SQLrand: Preventing SQL Injection Attacks," Proc. the 2nd Applied Cryptography and Network Security (ACNS) Conference, pp. 292-302, Jun 2004. Article (CrossRef Link)

[7] C.Cortes and V.Vapnik, "Support vector networks," Machine Learning, 20, pp. 273-297, 1995. Article (CrossRef Link)

[8] D.Pedro and M.Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," Machine Learning, 29, pp. 103-137, 1997. Article (CrossRef Link)

[9] Belur V.Dasarathy, "Nearest neighbor (NN) norms: Nn pattern classification techniques," IEEE Computer Society Press, 1991. Article (CrossRef Link)

[10] N.Cristianini and J.Shawe-Taylor, "An Introduction to Support Vector Machine and Other Kernel-based Learning Methods," Cambridge University Press, 2000. Article (CrossRef Link)

[11] David L.Olson and D.Delen, "Advanced Data Mining Techniques," Springer; 1 edition, pp. 138, Feb 2008. Article (CrossRef Link)

[12] R. Komiya, Incheon Paik, Masayuki Hisada, Classification of Malicious Web Code by Machine Learning, Proceedings of IEEE 3rd International Conference on Awareness Science and Technology, Dalian, China, September 27-30, 2011. Article (CrossRef Link)

Incheon Paik received his ME and PhD degrees in electronics engineering from Korea University in 1987 and 1992, respectively. During 1993-2000, he worked as an associate professor at Soonchunhyang University, Korea. From 1996 to 1998, he was a visiting researcher at the State Key Laboratory, Beihang University, Beijing, China. He is currently an associate professor at the University of Aizu, Japan. His research interests include Big Data Science, Semantic Web, web services and their composition, web data mining, awareness computing, security for e-business, and agents on Semantic Web. He served at several conferences as a program chair and program committee member for numerous international conferences. He is a member of the ACM, IEEE and IPSJ.