A Feature Extraction Method and Recognition Algorithm for Detection Unknown Worm and Variations based on Static Features

Tran Cong Hung, Dinh Xuan Lam, *Member, IEEE*

***Abstract*— There are many difference algorithms used for unknown worm detection. Some algorithms use static features, while others use dynamic features. However, no algorithm that can *perfectly* detect all unknown worms. Because, each detection method has its own drawbacks. It's difficult to detect polymorphic worms with only static features or it takes more time to execute dynamic detection algorithms. This paper describes an algorithm for detecting unknown worms and its variations based on features previously extracted from the analyzed files. This set of features is statically defined in this proposal and the method for extracting such features is also described. The proposed algorithm can detect worm and its variations with a small sample features set. This approach is not only applied well to detect worms with static features but also can be developed to detect worms based on their dynamic features and behavious. This is a first-attempt for demonstrating the effectiveness of the detection algorithm that uses both static features and dynamic features.**

***Index Terms*— computer virus, static feature, variations, worm detection.**

I. INTRODUCTION

RECENTLY, many computers and network systems in the world have been attacked by computer viruses. Anti-virus groups have done many research works and worked in several many different approaches. The antivirus software has been quite successful with signature-based virus detection technology. However, this method has drawbacks that should have a copy of the malicious code to extract recognition sample.

To solve this problem, antivirus organizations have added recognition technology into their antivirus softwares. This technology can detect behaviors and intents of the virus. However, antivirus softwares are difficult to distinguish between regular behaviors of application and destructive behaviors of virus (such as benign applications creating and deleting temporary files, while the virus copies itseft to create and delete user data) and it is difficult to detect intents of polymorphic files in the infected system.

Manuscript received April 9th, 2011. Accepted: April 26th, 2011.

Tran Cong Hung, Ph.D. is with the Posts & Telecommunications Institute of Technology, Vietnam (e-mail: conghung@ ptithcm.edu.vn).

Dinh Xuan Lam, Eng., is with the Posts & Telecommunications Institute of Technology, Vietnam (e-mail: dxlamdb@yahoo.com).

The existing malicious code detection methods are not adequate. Because "*no algorithm that can perfectly detect all possible viruses*" [1] so computer virus recognition problems are still *open problems* for the present [2, p35].

In this paper, we propose the recognition technology that can detect quite good worm and its variations. This approach can apply to detect unknown malicious code not only in executable files but also in run-time running processes. This paper is a first-attempt for demonstrating the effectiveness of such algorithm and only covers the analysis of static files and not processes at run-time stage.

Our paper is divided into the following main parts: the first part present "Introduction". The second part present "related work". The third part present "viruses detection mechanism". The fourth part present "experiment result". The fifth part present "conclusion".

II. RELATED WORK

There are many difference techniques used for malicious codes detection such as secure hash codes, neural networks, data mining, machine learning techniques, or comparisons with past copies.

Antivirus softwares combined signature-based recognition techniques with heuristic techniques, such as Bloodhount of Symantec, Heuristic scan of McAfee and Panda, Hash scan of BitComet, or using IBM's footprint technology to monitor internet transactions of BitDefender.

In our antivirus application (ATV2011), we used secure hash codes SHA-1 to detect trojans and applied this algorithm to restore a file when it is infected by one or more unknown malicious codes. The secure hash algorithm SHA-1 is one of a number of cryptographic hash functions published by the National Institute of Standards and Technology as a U.S. Federal Information Processing standard. We will present this recognition technology in another paper.

Jeffrey O. Kephart, Gregory B. Sorkin, Morton Swimmer and Steve R. White have described a immune system for computers that senses the presence of a previously unknown pathogen that within minutes, automatically derives and deploys a prescription for detecting and removing the pathogen [6]. Vesselin Bontchev summarized some ideas that are likely to be used by virus writers in the future and suggested the kind of measures that could be taken against them [7]. Wing Wong presented an effective metamorphic virus detection technique based on hidden Markov models [8]. M.G. Schultz, E. Eskin, E.Zadok, S.J. Stolfo used data mining methods for detection of new malicious executables [9]. M. Debbabi et al. discussed a dynamic monitoring mechanism called DaMon. This is capable of stopping certain malicious actions based on the combined accesses to critical resources according to rudimentary specifications [10]. George I, Davida, Yvo G. Desmedt and Brian J. Matt describes the use of cryptographic authentication for controlling computer viruses [11]. Jose Nazario presented traffic analysis technology to detect internet worms [12].

Recently, Zhang,B., Yin,J. and Hao,J. proposed fuzzy pattern recognition method [3], using support vector machine to detect unknown computer viruses [4]. Authors used API function calls as a main feature to detect unknown malicious executables. Liu Guozhu and Shang Yanjun represented amalgamation genetic algorithm into ant colony algorithm to detect unknown virus [5].

III. VIRUSES DETECTION MECHANISM

In this paper, a new approach for a worm and its variations detection is being proposed. It can be described as follows:

-- *Feature extraction*

-- *Unknown malicious executables detection* algorithm

A. Feature Extraction

In the study of unknown malicious executables code detection, Zhang,B., Yin,J. and Hao,J. used fuzzy pattern recognition method [3], It includes an extraction algorithm that use windows API function calls as a main feature to detect malicious executable or benign application. However, the algorithm achieved only when the antivirus application have to run executable files and observing its behavior. It is ineffective if malicious executables don't use api functions or using for static features analysis.

We have carried out the algorithm modified by adding variable m_i to determine the maximum number that a feature is observed. Our purpose is limit the frequency of occurrence and decrease feature analysis timing. And the algorithm does not only use for dynamic features extraction but also can be applied to extract static features. This extraction result will be used for malicious code detection algorithm that we propose in section 3.2.

Depending on the type of detected virus (malicious executables, macro virus, etc…), static features that are proposed for extraction are different. Based on the modification of Zhang,B.'s extraction method, the extraction algorithm can be described as follows:

Step 1. select the sample data set Q as

Q = V + N.

where

$V = \{V_1, V_2, \ldots, V_i\}$, 1<=i<=s, is the malicious code set.a

$N = \{N_1, N_2, \ldots, N_i\}$, 1<=i<=n, is the benign file set.

Step 2. determine all features A_i can appear in the sample data set Q.

$A = \{A_1, A_2, \ldots, A_i\}$, 1<=i<=p, is the features set that can appear in sample data set.

Step 3. count the number of occurences of the feature A_{ij}^V in every malicious code V_j and the number of occurences of the feature A_{ik}^N in every benign file N_k.

where:

j is the j-th malicious code

k is the k-th benign file

A_i is the i-th feature

Step 4. calculating average frequency $P(A_{ij}^V)$ and $P(A_{ik}^N)$ of each feature A_i in the j-th malicious code or the k-th benign application.

$$P(A_{ij}^V) = \left\{ \begin{array}{c} 0, m_i = 0 \\ \dfrac{A_{ij}^V}{m_i}, m_i > 0 \end{array} \right\}$$

$$P(A_{ik}^N) = \left\{ \begin{array}{c} 0, m_i = 0 \\ \dfrac{A_{ik}^N}{m_i}, m_i > 0 \end{array} \right\}$$

where:

m_i is the *maximum number of occurrences of i-th feature* in a specific malicious code of set V or a benign application of set N. This allows the i-th feature detection time limit in malicious code or benign application.

m_i can be determine based on *data mining result* with a sample data set, or gathered *experiences from experts*.

Step 5. The following formulas compute average frequency of each features in malicious code set V and benign files set N.

$$E(A_i^V) = \frac{1}{v}\sum_{j=1}^{v} P_{ij}^v$$

$$E(A_i^N) = \frac{1}{n}\sum_{k=1}^{n} P_{ik}^N$$

where:

v is the number of malicious codes in set Q

n is the number of benign files in set Q.

Step 6. The following formula computes total mean frequency of each feature (A_i) in set Q.

$$E(A_i) = \frac{E(A_i^v) + E(A_i^N)}{2}$$

Step 7. Compute mean square deviation $D(A_i)$ of each feature A_i as

$$D(A_i) = \sqrt{(E(A_i) - E(A_i^v))^2 + (E(A_i) - E(A_i^N))^2}$$

Step 8. We sorted features according to $D(A_i)$ sequence and choose the first t-th feature as the feature vector to recognize malicious codes and benign files.

$T = \{T_1, T_2, \ldots, T_i\}$, 1<=i<=t , $T \subset A$

B. Unknown Malicious Code Detection Algorithm

This part introduces detaily the malicious code detection algorithm that we proposed. The result of detection a file is "benign file" or "malicious code".

F defined as the file detects

C defined as the set that contains 2 subset B and M.

$C = \{B, M\}$, where B represents "detected benign files set" and M represents "detected malicious codes set".

T defined as proposed features set that is chosen from A set in feature extraction algorithm to recognize malicious codes or benign files.

$T = \{T_1, T_2, ..., T_i\}$, 1<=i<=t

Every feature T_i has four values $D(T_i)$, m_i, $E(T_i^v)$ and $E(T_i^n)$ to determine it's membership level in B subset and M subset.

Algorithm's objective is to determine any *F file belongs to B or M*.

The algorithm can be described as follows:

Input :

File F

Set of features T (every T_i has four values $D(T_i)$, m_i, $E(T_i^v)$ and $E(T_i^n)$)

Warning level W (high, medium, low)

Output :

F is "benign file" or "malicious code"

Step 1. Divide set T into two subset of features T1 and T2

$T1 = \{K_1, K_2, \ldots, K_i\}$, 1<=i<=k, $T1 \subset T$

$T2 = \{D_1, D_2, \ldots, D_j$, 1<=j=d, $T2 \subset T$

T1 is the set of static features. Every K_i has four values $D(K_i)$, $E(K_i^v)$, $E(K_i^n)$ and m_i

T2 is the set of dynamic features. Every D_j has four values $D(d_j)$, $E(D_j^v)$, $E(D_j^n)$ and m_j

Step 2. Initialize values

$f_{m =}$ 0

$f_{b =}$ 0

S = 0

f_m defined as the degree membership of the file F in M set.

f_b defined as the degree membership of the file F in B set.

S defined as the number of suspicious features.

Step 3. Sort features K_i according to $D(K_i)$ descending sequence. Sort features D_j according to $D(D_j)$ descending sequence.

Step 4.

For all $K_i \subset T1$

Count the number of occurrences of features K_i^f.

If $K_i^f = m_i$ then stop counting.

Compute percentage of occurrences of feature K_i in file f.

$$P(K_i^f) = \frac{K_i^f}{m_i}$$

In the formula, m_i is the maximum number of occurrences of i-th feature in file f.

```
If  E(K_i^v) ≥ E(K_i^n) then
    If  P(K_i^f) ≥ E(K_i^v) then
        f_m = f_m + D(K_i)
        S = S + 1
    Else
        If  P(K_i^f) > E(K_i^N) then
            S = S + 1
        End if
    End if
ELSE
    If  P(K_i^f) ≥ E(K_i^N) then
        f_b = f_b + D(K_i)
    Else
        If  P(K_i^f) > E(K_i^V) then
            S = S + 1
        End If
    End if
End if
If f_m >= 1 then
    return "f is the malicious code"
Else
    If f_b >= 1 then
        return "f is the benign file"
    End if
End if
```

Next

Step 5.

For all $D_j \subset T2$

Count the number of occurrences of features D_j^f.

If $D_j^f = m_i$ then stop counting.

Compute percentage of occurrences of feature D_j in file f.

$$P(D_j^f) = \frac{D_j^f}{m_j}$$

In the formula, m_i is the maximum number of occurrences of i-th feature in file f.

```
If  E(D_j^v) ≥ E(D_j^N) then
    If P(D_j^f) ≥ E(D_j^v) then
        f_m = f_m + D(D_j)
        S = S + 1
    Else
        If  P(D_j^f) ≥ E(D_j^N) > then
            S = S + 1
        End if
    End if
ELSE
    If  P(D_j^f) ≥ E(D_j^N) then
        f_b = f_b + D(D_j)
    Else
        If  P(D_j^f) > E(D_j^v) then
            S = S + 1
        End If
    End if
End if
If f_m >= 1 then
    return "f is the malicious code"
Else
    If f_b >= 1 then
        return "f is the benign file"
```

End if

End if

Next

Step 6. Test warning level W and value in S variable. Warning level W shows correctness of the determination a infected file or a benign file.

Select case W

Case "high":

if ($\frac{s}{k+d}$ >0.9) OR (f_m - f_b >0.75) then

return "f is the malicious code"

else

return "f is the benign file"

end if

Case "medium":

if ($\frac{s}{k+d}$ >0.75) OR (f_m - f_b >0.5) then

return "f is the malicious code"

else

return "f is the benign file"

end if

Case "low":

if ($\frac{s}{k+d}$ >0.5) OR (f_m >0.5) then

return "f is the malicious code"

else

return "f is the benign file"

end if

End Select

C. *Analysis*

When a file is recognized as a worm, it is saved in M set, and a file is recognized as a benign file, it is saved in B set. Antivirus software can use this result to detect its variations by other algorithms such as secure hash algorithm, … In the future, we use this result to calculate again $D(T_i)$ as a learning machine algorithm and we present it in another paper.

Features A_i are determined after analyze features of Q files set. It includes static and dynamic features such as api function calls, behaviors of worm and benign application, … Features T_i are determined from set A by feature extraction algorithm, after sorting features A_i according to $D(A_i)$ descending sequence. For example, if we choose api function calls as features, set A includes all api functions of windows. But "unknown malicious code detection algorithm" only examines api functions that chosen in set T.

There are some features that you always use them to detect statically. Some features always use to detect dynamically. But some features can use for both. For example, feature "The external storage device contains many files that have the same contents with file f". If our USB doesn't infect worm, we use this feature as a dynamic feature. When our USB infected worms such as w32-virut.gen (the worm is named by Avira), we can use this feature as a static feature when we plug this USB into our computer.

S value indicates the number of "suspicious" features that antivirus software found in file f. S and (f_m-f_b) values determine f is malicious or not when f_m<1 and f_b < 1.

$E(T_i^N)$ and $E(T_i^V)$ are values of feature T_i and they are previously calculated from set Q. Set Q doesn't include files that we want to detect. It is easy to realize that $D(T_i)$ max = 0.71 when $E(T_i^N)$=1 and $E(T_i^V)$=0 or $E(T_i^N)$=0 and $E(T_i^V)$=1. It means that T_i is particular feature of malicious codes or benign application. If you find such a feature in file f, you can conclude f is a malicious code (or benign application) but your appraisal is not 100% precisely because $D(T_i)$ is only calculated on a sample files set. When set Q is large enough for data mining technique, $D(T_i)$ value decreases. Because it's difficult to find a particular feature of all malicious codes or all benign files. So the algorithm examines many features to get fm >=1 or fb >=1. In this case, if we find a feature that has $D(T_i)$ = 0.71, our appraisal is more precisely. Our algorithm always previously examines features that have higher D(Ti) value. Because they are particular features of malicious codes or benign application. For example, we examine a file word, if feature "no macro" is detected, it is not necessary to continue. Our approach allows to save time for detecting worms and polymorphic worm. For example, W32.sality.y is a polymorphic worm. Although this worm can create new files that have difference size. But it still have features that we can detect it statically when it infected in our USB or in our computer. If we only have a new file and we don't know if it is worm or not? Antivirus software runs it to detect dynamic features. But when our computer infected W32.sality.y or we plug an USB that infected W32.sality.y into our computer, antivirus software can detect it that needn't run file. It's not necessary to examine all features of set T.

IV. EXPERIMENT RESULTS

We used 120 benign programs and 100 malicious executable programs that are in the Windows Portable Executable (PE) format as dataset for experiment (table 1). The clean programs were gathered from a freshly installed Windows XP machine. We installed WINXP source on the new harddisk and we also tested them by antivirus softwares, including AVIRA, Bitdefender, BKAV and Kaspersky. All of files were recognized as benign applications. Malicious codes (such as w32-virut.AR, W32-virut.gen, w32.sality.Y, trojan crypt.pepm.gen, mabezat.b, trojan spy.gen) is recognized by one of the following antivirus softwares: AVIRA, BKAV, Kaspersky.

TABLE 1
SAMPLE DATA IN EXPERIMENT

W	Sample space	Training set	Testing set
Benign file	120	50	70
Malicious file	100	30	70
Sum	220	80	140

TABLE 2
AN EXAMPLE OF SET T ARE PROPOSED FOR UNKNOWN MALICIOUS CODE DETECTION ALGORITHM

Features
1. Executable file has the same name as folder in storage disk drive.
2. File has the same content with one or many processes.
3. File has the same contents with one a many files in a disk drive.
4. The external storage device contain many files that have the same contents.
5. File change it's size but no change system time
6. Executable file has hidden attribute.
7. Many system files increase in the same size
8. Executable file has the same name as word file and word file has hidden attribute.
9. File has the same contents with one a many files in the system directory.
10. Create Autorun.inf file in many storage disk drives.
11. Executable file in autorun.inf is the same file in startup keys of windows registry
12. Process file is as the same as executable file in autorun.inf but difference path
13. Copy many times a file to the system directory
14. CallNextHookEx
15. GetFileSize
16. ExitProcess
17. VirtualAlloc
18. CloseSocket
19. GetKeyboardType
20. GetTickCount
21. GetCurrentProcessID
22. GetSystemTimeAsFileTime

TABLE 3
AN EXAMPLE OF SET T1 (*STATIC FEATURES*) ARE PROPOSED FOR UNKNOWN MALICIOUS CODE DETECTION ALGORITHM

Feature $T1_i$	m_i	$E(K_i^v)$	$E(K_i^n)$	$E(K_i)$	$D(K_i)$
Executable file has the same name as folder in storage disk drive.	1	0.67	0	0.34	0.47
File has the same content with one or many processes.	2	0.83	0	0.42	0.59
File has the same contents with one a many files in a disk drive.	3	0.78	0.11	0.45	0.47
The external storage device contain many files that have the same contents.	3	0.83	0	0.42	0.59
File change it's size but no change system time	1	0.67	0	0.34	0.47
Executable file has hidden attribute.	1	0.83	0	0.42	0.59
Many system files increase in the same size	3	0.83	0	0.42	0.59
Executable file has the same name as word file and word file has hidden attribute.	1	0.12	0	0.06	0.09
File has the same contents with one a many files in the system directory.	3	0.28	0	0.14	0.20

We used 50 benign programs (V) and 30 codes executable files (N) in the Q sample data set (training set).

The A features set is determined after analyze features of Q set (by data mining technique or our direct files analysis result, it includes static and dynamic features such as api function calls, behaviors of worm and benign programs, …).

The T features set is determined from A set by feature extraction algorithm, after sorting features T_i according to $D(T_i)$ descending sequence. An example of T features set are proposed for unknown worm and its variations detection algorithm as table 2. They include 22 typical features (t=22) of worms.

Features that we proposed in table 2 is to clarify our algorithm. That's not all. To detect worm "in general", we have to expand set Q, and the size of set A also increases. Set A can include static features and dynamic features.

Result of dividing T set into subset T1 and T2 is determined in table 3 and table 4. T1 includes the static features. T2 includes the dynamic features. Some features are particular features of T1 set or T2 set. But some features can belong to both (such as API function calls). This paper only proposes the features of T2 set that can belong to T1 set.

TABLE 4
AN EXAMPLE OF SET T2 (*DYNAMIC FEATURES*) ARE PROPOSED FOR UNKNOWN MALICIOUS CODE DETECTION ALGORITHM

Feature $T2_j$	m_i	$E(D_j^v)$	$E(D_j^n)$	$E(D_j)$	$D(D_j)$
Create Autorun.inf file in many storage disk drives.	2	0.83	0	0.42	0.59
Executable file in autorun.inf is the same file in startup keys of windows registry	1	0.06	0	0.03	0.04
Process file is as the same as executable file in autorun.inf but difference path	1	0.33	0	0.17	0.23
Copy many times a file to system directory	3	0.78	0.11	0.45	0.47
CallNextHookEx	1	0.15	0.10	0.13	0.04
GetFileSize	1	0.23	0.16	0.20	0.05
ExitProcess	1	0.59	0.34	0.47	0.18
VirtualAlloc	1	0.38	0.21	0.29	0.12
CloseSocket	1	0.16	0.02	0.09	0.10
GetKeyboardType	1	0.12	0.01	0.06	0.08
GetTickCount	1	0.12	0.64	0.38	0.37
GetCurrentProcess ID	1	0.17	0.65	0.41	0.34
GetSystemTimeAs FileTime	1	0	0.61	0.31	0.43

Experiment results of detection unknown worm on testing set are showed in table 5.

TABLE 5
EXPERIMENTAL RESULT OF DETECTION SYSTEM

W	False Negative	False Positive
High	4.29%	2.86%
Medium	7.14%	10.00%
Low	8.57%	11.43%

V. CONCLUSION

We presented a method for detecting unknown worm and its variations based on features previously extracted from the analyzed files. The proposed algorithm can detect unknown worm and its variations with a small sample features set. This approach is not only applied well to detect worms with static features but also can be developed to detect worms based on their dynamic features and behavious. It overcomes drawbacks of unknown worms detection algorithms based on only static features or dynamic features. It can protect user realtime effectively and has good effect on unknown worm detection in USB. In the future, we will continue to expand our algorithm for both dynamic features of malicious code and static features of benign executables to gain higher accuracy and detection rates. We also would like to test this method on a larger set of malicious and benign executables.

REFERENCES

[1] David M. Chess, Steve R.White, "An Undetectable Computer Virus", *Virus Bulletin Conference*, September 2000.

[2] Essam Al Daoud, Iqbal H. Jebril, Belal Zaqaibeh, "Computer Virus Strategies and Detection Methods", *Int. J. Open Problems Compt. Math.*, Vol. 1, No. 2, September 2008.

[3] Zhang,B., Yin,J., Hao,J., "Using Fuzzy Pattern Recognition to Detect Unknown Malicious Executables Code". In: Wang,L.,Jin,Y.(eds.):Fuzzy Systems and Knowledge Discovery. LNAI,Vol.3613. Springer-Verlag, Berlin Heidelberg New York(2005) 629-634.

[4] Zhang,B., Yin,J., Hao,J., Zhang.D., Wang.S., "Using Support Vector Machine to Detect Unknown Computer Viruses", In: International Journal of Computational Intelligence Research, ISSN 0973-1873 Vol.2, No. 1 (2006), pp. 100-104.

[5] Liu Guozhu, Shang Yanjun, "Unknown Virus Detection Method Amalgamation Genetic Algorithm into Ant Colony Algorithm", *Journal of Computers*, Vol. 5, No. 6, June 2010, pp. 879-884.

[6] Jeffrey O. Kephart, Gregory B. Sorkin, Morton Swimmer and Steve R. White, "Blueprint for a Computer Immune System", *Virus Bulletin International Conference San Francisco, California,* October 1-3, 1997.

[7] Vesselin Bontchev, "Future Trends in Virus Writing*", 4th International Virus Bulletin Conference*, 1994, pp.65-82.

[8] Wing Wong, "Analysis and Detection of Metamorphic Computer Viruses", A writing project presented to the faculty of the department of computer science San Jose State University, May 2006, pp. 25-60.

[9] M.G. Schultz, E. Eskin, E.Zadok, S.J. Stolfo, "Data Mining Methods for Detection of New Malicious Executables", sp, pp. 0038, IEEE Symposium on Security and Privacy, 2001.

[10] M. Debbabi et al., "Dynamic Monitoring of Malicious Activity in Software Systems", *Symposium on Requirements Engineering for Information Security*, Indianapolis, Indiana, USA, March 5-6, 2001.

[11] George I, Davida, Yvo G. Desmedt and Brian J. Matt, "Defending Systems Against Viruses through Cryptographic Authentication", *Proceedings of the 1989 IEEE Symposium on Computer Security and Privacy,* 1989, pp. 312-318.

[12] Jose Nazario, "Defense and Detection Strategies against Internet Worms", *Artech House Inc.,* 2004, pp. 137-158.

. **TRAN CONG HUNG was born in VietNam in 1961**
He received the B.E in electronic and Telecommunication engineering with first class honors from HOCHIMINH university of technology in VietNam, 1987.
He received the B.E in informatics and computer engineering from HOCHIMINH university of technology in VietNam, 1995.
He received the master of engineering degree in telecommunications engineering course from postgraduate department HaNoi university of technology in VietNam, 1998.
He received Ph.D at HaNoi university of technology in VietNam, 2004.
His main research areas are B – ISDN performance parameters and measuring methods, QoS in high speed networks, MPLS.
Currently, he is a lecturer, Deputy Head of Training & Science Technology Department in Posts and Telecoms Institute of Technology (PTIT), in HOCHIMINH City, VietNam.

DINH XUAN LAM was born in Vietnam in 1971.
He received the B.E in Physical from DALAT university in VietNam, 1993.

He received the B.E in Information Technology with first class honors from HANOI university of polytechnic in VietNam, 2001. Will receive Master in Post and Telecommunication Institute of Technology (PTIT), 2011, major in Networking and Data Transmission.
His main research fields are Application programming and Security.
Senior of IT group of Card Department at Vietnam Import & Export Bank, 2002-2004
Deputy Head of Software Consulting and Implementing Department at Kha Thi Corp., 2005 - 2006
Team Leader of Security Group at Aplis VietNam Corp., 2007.
Currently, he is a lecturer at CaoThang Technical College, in HOCHIMINH City, VietNam.