Network Penetration Testing Scheme Description Language

Zhiyong Dai, Liangshuang Lv, Xiaoyan Liang, Yang Bo
State Key Laboratory of Virtual Reality Technology and System
Beijing Key Laboratory of Network Technology
School of Computer Science and Engineering, Beihang University
Beijing, China
{dzy, liangxiaoyan, boyang}@cse.buaa.edu.cn, lls@buaa.edu.cn

Abstract—Penetration testing is widely used to help ensure the security of the network. Traditional penetration testings were manually performed by tester according to scheme, the process is usually complex resulting in that it is labor-intensive and requires tester to be familiar with all kind of tools. So it is very desirable to use a unified method to describe the scheme which can be identified by computer, then the computer can be used to substitute for tester to perform penetration testing. A new method is presented to describe penetration testing scheme. We propose a language for tester to describe the penetration testing scheme. Based on the conceptual model of penetration testing scheme we design the penetration testing scheme description language (PTSDL) and outline the important EBNF. PTSDL is declarative abstracting the process of penetration testing which makes this language flexible, extensible and adaptable to new penetration testing method. Use of this language is illustrated by an experimental study, which shows that the language can effectively describe the penetration testing scheme.

Keywords-penetration testing; scheme; language

I. INTRODUCTION

Penetration testing is a new security risk evaluation technology. Karen defined penetration testing as security testing in which assessors mimic real-world attacks to identify methods for circumventing the security features of an application, system, or network[1]. The idea is to find out how to gain unauthorized access to its information and information systems[2]. Penetration testing has become a widely used and integral part of quality assurance techniques for network. In fact, many government agencies and trade groups, such as the Communications and Electronic Security Group in the U.K., OWASP, and OSSTMM, accredit penetration testers and sanction penetration testing "best practices."[3]

Penetration testing is normally completed with finite resources, focused on a particular area, over a finite period of time[2]. Penetration testing scheme is used to describe how to perform a penetration testing, including testing goals, testing scope, testing time and concrete steps.

Penetration testing can be invaluable, but it is labor-intensive and requires great expertise to minimize the risk to targeted systems[1]. Penetration testing should be performed only after careful consideration, notification, and planning. After that, tester will manually perform penetration testing. It is not easy to perform penetration testing correctly and in a short time because it is a complex process, In order to solve those problems, in this paper, we focus on a new method to describe penetration testing scheme. We propose a conceptual model of penetration testing scheme. This model consisted by relations of penetration testing scheme's conceptions. Then we design penetration testing description language (PTSDL) and outline its EBNF. With the language, tester can easy to describe a penetration testing scheme.

The rest of this paper is organized as follows. In section II, related work on penetration testing model, penetration testing scheme and penetration testing tool are introduced. The conceptual model of network penetration testing scheme is discussed in section III. We outline PTSDL's EBNF in section IV. Section V validates the model and the language with an experiment. Finally, conclusion is presented.

II. RELATED WORK

In the field of penetration testing, there are three important theoretical models. National Institute of Standards and Technology(NIST)published Technical Guide to Information Security Testing and Assessment, SP800-115[1],the guide proposed a penetration testing model which is composed by four phases, including planning, discovery, attack, and reporting. Executing an attack is at the heart of penetration testing, which is a represented process including gaining access, escalating privileges, system browsing and install additional tools steps. System browsing can return collected information to discovery phrase that make penetration testing to a dynamic feedback process. It is very useful in actual penetration testing. ISECOM published Open-Source Security Testing Methodology Manual (OSSTMM)[4]. The manual proposed one method for performing a thorough security test, and solved the problems of information security, process security, internet technology security, communications security, wireless security and physical security. It is provides reference to tester when designing penetrating testing. Federal Office for Information Security published a penetration testing model which is composed by five phases[5]. Those phases are preparation, reconnaissance, analyzing information and risks, active intrusion attempts and final analysis. More important is that every phase's output is standard, so tester very clears what needs to be done at each step and provide a good reference to standardize penetration testing.

According to penetration testing models, some researchers have developed penetration testing tools that aimed at special requirement. Paper[6] designed and implemented an automated penetration testing system based on SNMP, multi-source vulnerability database and a plug-in mechanism which was based on the NASL. Paper[7] provided a network information security penetration test platform based on network leak ranks. Literature[3] proposed an approach for penetration testing based on improved input vector identification and automated response. Literature[8] analyzed a model-driven penetration test framework for Web applications which provides a repeatable, systematic and cost-efficient approach fully integrated into a security-oriented software development life cycle. Literature[9] discussed an extensible exploit framework for automation of penetration testing without loss of safety and describe possible methods for sanitizing unreliable code in each part of the framework. Some companies also developed penetration testing tools, such as IMPACT[10], CANVAS [11] and Metasploit[12].

We can see that the penetration testing tools can not set a concrete goal, take gain a host's privilege for example, only verify identified potential vulnerabilities by attempting to exploit them without considering the relation of vulnerabilities, The tester can not grasp the penetration testing in macroscopic image. Tester must have prolific knowledge with tools to perform penetration testing. In order to solve those problems, before perform penetration testing, we use a language describes the penetration testing scheme which is composed by penetration testing cases that illustrate how to perform penetration testing. It is efficient and accurate to perform penetration testing according to scheme, resolve the problem that tester must pay much attention to the process.

III. Analysis of Conceptual Model

In this section, we present the analysis model of network penetration testing scheme. At First, we introduce the basic concepts of network penetration testing scheme, then we description the conceptual model of the system.

A. The definition of basic concepts

- Definition 1: Penetration testing target: a network which is composed by hosts and the relation of hosts.

$$pentest\_target = (\text{HOSTS}, \xi_{\text{HOSTS}}).$$

HOSTS denotes the set of hosts that to perform penetration testing, ξ_{HOSTS} denotes the set of relations of hosts.

- Definition 2: Penetration testing time: the time that tester perform penetration testing, which is a ordered tuple composed by begin time and end time.

$$\begin{cases} pentest\_time = \langle begin, end \rangle, \\ begin, end \in \text{TIME}. \end{cases}$$

TIME denotes the set time that to perform penetration testing.

- Definition 3: Penetration testing goal: gain access right of a host that belong to penetration testing target.
- Definition 4: access right: the role when tester access the penetration testing target.

$$\begin{cases} \text{ACCESS} ::= \{(t,h,r) \mid t \in \text{TESTER}, h \in \text{HOSTS}, p \in \text{PRIVILEGE}\}, \\ \text{PRIVILEGE} ::= \{\varepsilon, guest, user, adminstrator, root\}. \end{cases}$$

TESTER denotes the set of tester, PRIVILEGE denotes the set of privilege.

- Definition 5: penetration testing case: four tuple which is composed by test goal, test environment, test method and expectation.

$$\begin{cases} test\_case ::= (goal, environment, method, expectation), \\ goal \in \text{GOAL}, environment \in \text{ENVIROMENT}, \\ method \in \text{METHOD}, \text{expectation} \in \text{EXPECTATION}. \end{cases}$$

GOAL denotes the set of testing case's goal, ENVIRONMENT denotes the set of testing case's environment, METHOD denotes the set of testing method, EXPECTATION denotes the set of expectation.

- Definition 6: goal of penetration testing case: the expectation of case, including access right and relation of hosts.
- Definition 7: penetration testing environment: composed by what's vulnerability to be exploit and the precondition, precondition includes access right and the relation of hosts.

$$\begin{cases} environment ::= (vulnerability, precondition), \\ vulnerability \in \text{VULNERABILITY}, \\ precondition ::= (access, \xi_{relation}), \\ access \in \text{ACCESS}, \ \xi_{relation} \in \xi_{\text{HOSTS}}. \end{cases}$$

VULNERABILITY denotes the set of vulnerability.

- Definition 8: vulnerability: triple tuple which is composed by identify, exploit result, exploit location.

$$\begin{cases} vulnerability ::= (id, result, locality), \\ id \in \text{CVE}, result \in (\text{ACCESS} \cup \text{R}_{\text{TRUST}}), \\ locality \in \text{LOCALITY}, \text{LOCALITY} ::= \{remote, local, lan\}. \end{cases}$$

id denotes the vulnerability identification, identified by cve number, CVE denotes the set of cve identifiers, $result$ denotes the result after exploit, $locality$ denotes the location where exploit, such as remote, local.

- Definition 9: penetration testing method: the methods to achieve the penetration testing goal.
- Definition 10: expectation: the result which expect for to get, including access privilege and the trust relation of hosts.

$$\text{EXPECTATION}::=\{\text{ACCESS},\text{R}_{\text{TRUST}}\}.$$

- Definition 11: relation of hosts: including connection, trust and service.

$$\xi_{\text{HOSTS}}::=\{\text{R}_{\text{CONNECT}},\text{R}_{\text{TRUST}},\text{R}_{\text{SERVICE}}\}.$$

$\text{R}_{\text{CONNECT}}$ denotes the set of connection relation, R_{TRUST} denotes the set of trust relation, $\text{R}_{\text{SREVICE}}$ denotes the set of service relation

- Definition 15: penetration testing scheme: the detail information of how to perform penetration testing, which is a four tuple that is composed by penetration testing time, penetration testing targets, penetration testing case and their' relation.

$$\begin{cases} scheme ::= (time, target, goal, \text{TESTCASE}), \\ time \in \text{TIME}, target \in \text{TARGET}, goal \in \text{GOAL}. \end{cases}$$

TARGET denotes the set of penetration testing targets, TESTCASE denotes the set of penetration testing cases.

B. The model of basic concepts

According to the definitions of the conceptions, Fig.2 gives the conceptual model of network penetration testing scheme (CMNPTS).

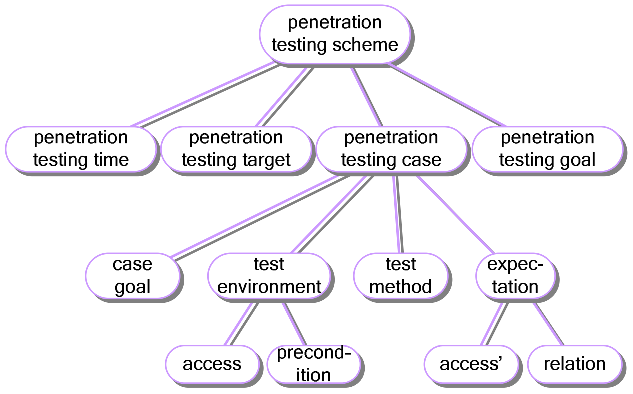


Figure 1 Conceptual model of network penetration testing scheme(CMNPTS)

IV. PTSDL

According to CMNPTS mentioned above, we design and implement a penetration testing scheme description language, describe the contents which the penetration testing scheme involved, and then outline the EBNF, finally we prove validity of PTSDL through an experiment.

According to the analysis, what's PTSDL have to describe including: penetration testing targets, penetration testing goal, penetration testing case, constraints. For design request, the design goals of PTSDL are:

- It is easy to use for penetration testers with simple grammar
- Has some scalability, can extend on demand through add key words

Extended Backus-Naur Form of PTSDL's constitution is outlined as follows.

Penetration testing scheme is composed by scheme declaration, global variable definition and scheme description.

<ptsdl>::=<scheme_declaration><global_variable_definition><scheme_description>

A. Scheme declaration

Scheme declaration used to declare the condition and target of penetration testing scheme, including the time, goal and target of penetration testing.

<scheme_declaration>::=<time_declaration> | <goal_declaration> | <target_declaration>

<time_declaration>::=**define time**'{'<time_value>',' <time_value>'}'

<goal_declaration>::=**define goal**'{'<final_goal >'}'

<target_declaration>::=**define target**'{'{<target_variable>}'}'

B. Global variable definition

Global variable definition used to define global variable, including variable statement and variable assignment. The variable has time style, goal style and expectation style.

- variable statement

<global_variable_definition>::=<variable_statement> | <variable_assignment>

<variable_statement>::=<variable_type> <variable_name>

<variable_type>::= **time | goal | expectation**

<variable_name>::=<time_variable> | <goal_variable> | <expectation_variable>

<time_variable>::=<string>

<goal_variable>::=<string>

<expectation_variable>::=<string>

- variable assignment

<variable_assignment>::=<variable_name>=<variable_value>

<variable_value>::=<time_value> | <goal_value> | <expectation_value>

<time_value>::=<function> | <time>

<function>::=**now()**

<goal_value >::=<final_goal> | <case_goal>

<final_goal>::=gain_privilege'(' <target_variable>',' <privilege_variable>')'

<case_goal>::=<goal_trustrelation> | <goal_privilege>

<goal_trustrelation>::=**goal_trust**'(' <target_variable> ','<target_variable> ')'

<goal_privilege>::=**goal_privilege**'(' <target_variable> ','hyper-<privilege_variable>')'

<privilege_variable>::=**guest** | **user** | **administrator** | **root**

<expectation_value>::=<expectation_trustrelation>|<expectation_privilege>

<expectation_trustrelation>::=**expecte_trust**'(' <target_variable>','< target_variable >')'

<expectation_privilege>::=**expecte_privilege**'('<target_variable>','<privilege_variable> ')'

C. *scheme description*

Scheme description used to describe the concrete details of the scheme, including penetration testing cases and constraints. It is organized by penetration path which can achieve the penetration testing goal.

<scheme_description>::= {**scheme** <scheme_name>‘{’ {<pentest_scheme_statement>} ‘}’}

<scheme_name >::=<string>

<pentest_scheme_statement>::=<pentest_case> | <constraints>

- Penetration testing case

Penetration testing case is composed by the goal, environment, method and expectation of penetration testing case. The environment of penetration testing includes vulnerability’s information and preconditions that required when exploit vulnerability.

<pentest_case>::=**if**‘(’<precondition>‘)’ **then**
use <method_variable> **exploit** <vulnerability_variable>
penetrate <target_variable> **expect** <expectation_variable> **out** <status_variable>
endif

<precondition>::=<atomic_precondition> | (<precondition>) && (<atomic_precondition>) | (<precondition>) ||(<atomic_precondition>) | !(<atomic_precondition>)

<atomic_precondition>::=<privilegecondition> | <relationcondition>

<privilegecondition>::=**own**‘(’ <target_variable>‘,’ <privilege_variable> ‘)’

<relationcondition>::=<connectcondition> | <servicecondition> | <trustcondition>

<connectcondition>::=**connect**‘(’ <target_variable>‘,’ <target_variable>‘,’ <port> ‘)’

<servicecondition>::=**service**‘(’ <target_variable>‘,’ <target_variable>‘,’ <service_variable> ‘,’ <port> ‘)’

<service_variable>::=**ftp | telnet | web**

<trustcondition>::=**trust**‘(’ <target_variable>‘,’ <target_variable> ‘)’

<target_variable>::=<network> | <computer>

<network>::=<domain> | <subnet>

<domain>::=**domain**<domain_name>

<subnet>::= **ip** <ip_constant> [**mask**<mask>]

<domain_name>::=<string>

<computer>::=<node> | <ip_constant>

<node>::=<string>

<method_variable>::=**overflow** | **racecondition** | **passwordcrack** | **login**

<vulnerability_variable>::=**cve**‘**-**’<string>‘**-**’<string>

<status_variable>::=<string>

- Constraints

Constraints used to judge whether the conditions are met to execute the next penetration testing case.

<constraints>::=**if**‘(’<conditon>‘)’ **then** <pentest_case> **endif** | **if**‘(’<conditon>‘)’ **then** <pentest_case> **else** <pentest_case >**endif**

<conditon>::=<atomic_condition> | (<condition>) && (<atomic_condition>) | (<condition>)||(<atomic_condition>) | !(<atomic_condition>)

<atomic_condition>::=<status_variable_condition> | <time_variable_condition>

<status_variable_condition>::=<status_variable><operator_variable>< status_variable>

<operator_variable>::= != | <= | >= | < | > | ==

<status_variable>::=**success | fail**

<time_variable_condition>::=<time_variable><operator_variable><time_variable>

<string>::=<letter> | <string><letter> | <string><digit> | <string><symbol> | <time>

<digit>::=<digit>::=[0-9]

<letter>::=<letter>::=[a-zA-Z]

<symbol>::=-|_

<ip_constant>::=<ip_dotted_decimal_numer>.<ip_dotted_decimal_numer>.<ip_dotted_decimal_numer>.<ip_dotted_decimal_numer>

V. EXPERIMENT

In this section, we conduct a penetration testing instance to illustrate the usage of PTSDL.

A. *Experimental environment*

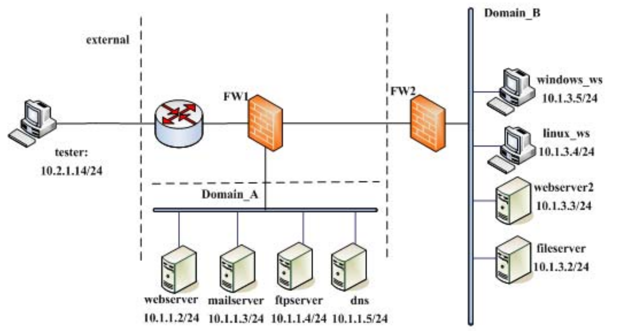


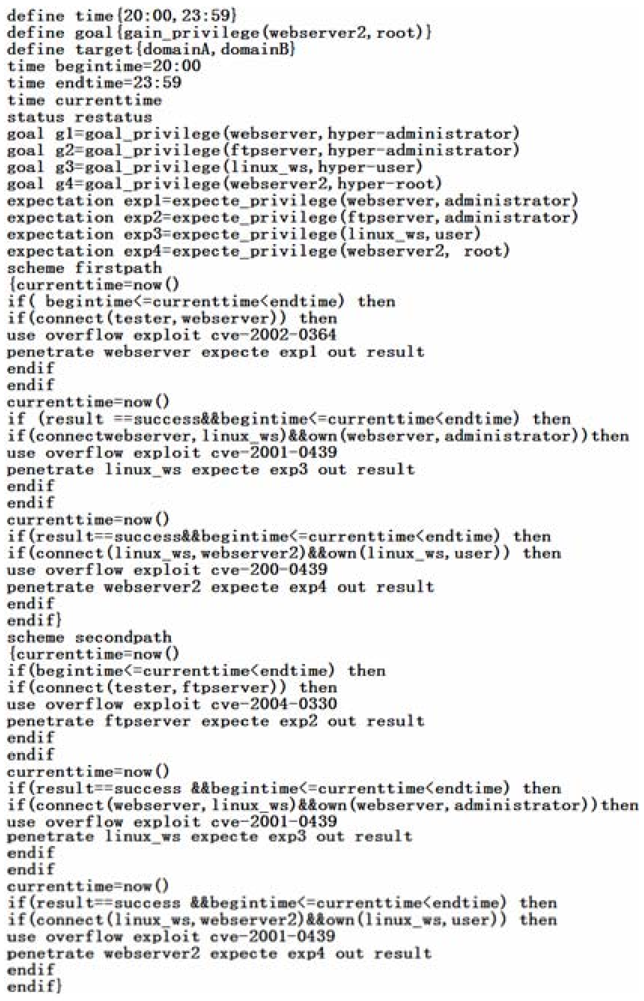
Figure 2 Topology of the experiment

A organization is connected to Internet through network which is composed by domain A and domain B. Domain A is DMZ area used to provide services, such as Web, Mail, Ftp and DNS. The webserver is running Microsoft IIS on port 80. The ftpserver is running Serv-U on port 21.Domain B is composed by workstations and internal servers. The linux_ws is running LICQ on port 5190. The webserver2 is running Apache to provide http service.

The configuration of network environment as follow: there is an overflow vulnerability in IIS of webserver identified by ‘CVE-2002-0364’, tester can gain administrator privilege by use of the vulnerability; there is an overflow vulnerability in Serv-U of ftpserver identified by ‘CVE-2004-0330’, tester can gain administration privilege by use of it; there is an overflow vulnerability in LICQ of linux_ws identified by ‘CVE-2001-0439’, tester can gain user privilege by use of it; there is an overflow vulnerability in Apache of webserver2 identified by ‘CVE-2003-0542’, tester can gain root privilege by use of it.

B. *PTSDL description*

In this experiment, the tester must perform penetration testing from 8pm to 12pm, the goal is to gain the root privilege of webserver2. The PTSDL description as follow:



```
define time{20:00,23:59}
define goal{gain_privilege(webserver2,root)}
define target{domainA,domainB}
time begintime=20:00
time endtime=23:59
time currenttime
status restatus
goal g1=goal_privilege(webserver,hyper-administrator)
goal g2=goal_privilege(ftpserver,hyper-administrator)
goal g3=goal_privilege(linux_ws,hyper-user)
goal g4=goal_privilege(webserver2,hyper-root)
expectation exp1=expecte_privilege(webserver,administrator)
expectation exp2=expecte_privilege(ftpserver,administrator)
expectation exp3=expecte_privilege(linux_ws,user)
expectation exp4=expecte_privilege(webserver2, root)
scheme firstpath
{currenttime=now()
if( begintime<=currenttime<endtime) then
if(connect(tester,webserver)) then
use overflow exploit cve-2002-0364
penetrate webserver expecte exp1 out result
endif
endif
currenttime=now()
if (result ==success&&begintime<=currenttime<endtime) then
if(connectwebserver, linux_ws)&&own(webserver,administrator))then
use overflow exploit cve-2001-0439
penetrate linux_ws expecte exp3 out result
endif
endif
currenttime=now()
if(result==success&&begintime<=currenttime<endtime) then
if(connect(linux_ws,webserver2)&&own(linux_ws,user)) then
use overflow exploit cve-200-0439
penetrate webserver2 expecte exp4 out result
endif
endif}
scheme secondpath
{currenttime=now()
if(begintime<=currenttime<endtime) then
if(connect(tester,ftpserver)) then
use overflow exploit cve-2004-0330
penetrate ftpserver expecte exp2 out result
endif
endif
currenttime=now()
if(result==success &&begintime<=currenttime<endtime) then
if(connect(webserver, linux_ws)&&own(webserver,administrator))then
use overflow exploit cve-2001-0439
penetrate linux_ws expecte exp3 out result
endif
endif
currenttime=now()
if(result==success &&begintime<=currenttime<endtime) then
if(connect(linux_ws,webserver2)&&own(linux_ws,user)) then
use overflow exploit cve-2001-0439
penetrate webserver2 expecte exp4 out result
endif
endif}
```

Figure 3 Description of the scheme using PTSDL

C. *Result analyze*

The above penetration testing scheme described by PTDSL interpreted through Interpreter, throughout the penetration testing process, tester needn't to maintain and control the process, the system will automatically perform penetration testing. The following is the final report. We can see that we achieve the goal which is to gain the root access right of webserver2 successfully.

```
gain root access right of webserver2
scheme_1{
step1:exploit cve-2004-0330 of ftpserver,gain administrator right
step2:exploit cve-2001-0439 of linux_ws,gain user right
step3:exploit cve-2003-0542 of webserver2,gain root right}
scheme_2{
step1:exploit cve-2002-0364 of webserver,gain administrator right
step2:exploit cve-2001-0439 of linux_ws,gain user right
step3:exploit cve-2003-0542 of webserver2,gain root right}
```

Figure 4 Reports of the penetration testing

We can draw a conclusion from experiment results that the PTDSL which is independent to CMNPTS, can describe penetration testing scheme.

VI. CONCLUSIONS

In this paper, we systematically analyzed the components of penetration testing scheme in details and present a new model CMNPTS. Based on these work, we designed description language PTDSL and outlined the main EBNF. The usage of PTDSL is illustrated by an experiment instance. We can see that system can use PTDSL to describe penetration testing scheme then automatically perform penetration testing, finally give the penetration testing report. In future, we will develop a friendly interface to show the penetration testing report for tester who can see the report conveniently.

ACKNOWLEDGMENT

This work is supported by the national research project.

We would like to express gratitude to Ms. Beibei Fan and Mr. Lu Shen for many helpful discussions. Substantial suggestion by all of them greatly improves the paper.

REFERENCES

[1] K. Scarfone, M. Souppaya, A. Cody and A.Orebaugh, "Technical Guide to Information Security Testing and Assessment", National Institute of Standards and Technology, Sep. 2008, pp. 36-39.

[2] J. P. Mcdermott, "Using an Ethical Hacking Technique to Assess Information Security Risk", Information Technology Advisory Committee, Jun. 2003, pp. 4-10.

[3] G. J. William, Halfond, S. R. Choudhary and A. Orso, "Penetration Testing with Improved Input Vector Identification", International Conference on Software Testing Verification and Validation, 2009, pp. 346-355, doi:10.1109/ICST.2009.26.

[4] P. Herzog, "Open-Source Security Testing Methodology Manual", Institute for Security and Opne Methodologies, Dec. 2006, pp. 11-50.

[5] German: BSI, "Study A Penetration Test Model", Federal Office for Information Security, 2005, pp. 20-36.

[6] B. XING, L. GAO, J. SUN and W. YANG, "Design and implementation of automated penetration testing system", Application Research of Computers, Jul, 2010.

[7] Z.Q. XU, "Study on the Penetration Platforin of Network Information Security", Guangdong University of Technology, May. 2008.

[8] P. L. XIONG, P. Liam, "A model-driven penetration test framework for Web applications", Eighth Annual International Conference on Privacy Security and Trust, 2010, pp. 173-180.

[9] H. Kwon, S. M. Lee, H. Lee, J. Kim and S. C. Kim, "HackSim: An Automation of Penetration Testing for Remote Buffer Overflow Vulnerabilities" C. Kim (Ed.): ICOIN 2005, LNCS 3391, 2005, pp. 652–661.

[10] T. P. Layton, "Penetration Studies-A Technical Overview", SANS Institute InfoSec Reading Room, May, 2002.

[11] H. Allen, H. Shon, N. Jonathan and E. Chris, "Gray Hat Hacking:the Ethical Hacker's Handbook", McGraw-Hill Osborne Media, 2007, pp. 139-160.

[12] G. Dennis, S. Fedor, T. Edward, "Racewalk: fast instruction frequency analysis and classification for shellcode detection in network flow", 2009, pp. 4-12, doi:10.1109/EC2ND.2009.9.