A Rule-based Approach to Security Test Automation on Network Layer

HYEOKCHAN KWON, JAEHOON NAH, KYOIL CHUNG
Information Security Research Division
Electronics and Telecommunications Research Institute
161 Gajeong-Dong, Yuseong-Gu, Daejeon, 305-350
SOUTH KOREA

Abstract: - Recently, Many projects have been implementing IPsec(IP Security) on the various Operating Systems for security of IPv6 network. But there is no existing tool that checks the IPsec-based systems, which provide IPsec services, work properly and provide their network security services well in the IPv6 network. In this paper, we propose a rule-based approach to security test automation on IPv6 network. So, we design STRDL(Security Test Rule Description Language) to define security test rule and rule execution engine for executing the rules defined by STRDL for testing security of the IPv6 network, and we provide implementation details. The system is divided into following part: User Interface part, Rule Execution Module Part, DBMS part and Agent that gathering information needed for security test.

Key-Words: - Security Test, IPv6, IPsec, Rule description language

1 Introduction

IP version 6(IPv6) is a new version of the Internet Protocol, designed as the successor to IP version 4(IPv4). IPv6 is finally gaining market momentum after many years of getting attention mainly in the standard forums, due to two recent market trends: the growing shortage of available IP addresses, mainly in the Far East, and the upcoming deployment of 3G mobile networks with IP address assignment per individual wireless phone or handheld device[1,2]. The changes from IPv4 to IPv6 fall primary into the following categories: Expanded Addressing Capabilities, Header Format Simplification, Improved Support for Extensions and Options and Flow Labeling Capability.[1] Currently 43 RFCs and 22 Internet drafts related to IPv6 have been made by IETF Internet Area IPv6 Working Group.[3] IPv6 is now a deployment phase. Several pieces of network equipment that support IPv6 have been shipped, and some network providers have started IPv6 commercial services. However, more implementations and experiments are necessary in some areas. One such ongoing field is Security for IPv6.[4]

For security of IPv6 network, IPsec(IP Security) is designed by IPsec Working Group of IETF. Currently nearly 18 RFCs and 30 Internet drafts related to IPsec protocols have been made[5-8]. IPsec is a method of protecting IP datagrams. This protection takes the form of data origin authentication, connectionless integrity, confidentiality, anti-replay protection, and limited traffic flow confidentiality. IPsec can protect any protocol that runs on top of IPv4 and IPv6 such as TCP, UDP, and ICMP. Recently, Several projects such as KAME, USAGI [9,10] implement IPv6 IPsec on the various Operating Systems.

In order to maintain the security of the network, we need the security test tool that can check current security status of the network. But there is no existing such a tool which checks the systems that provide IPsec services work properly and provide their network security services as well on IPv6 Network. There is several network security scanning tool such as Internet Scanner by ISS, Cisco Scanner by Cisco and LANguard network&port scanner [11-13]. But these tools only provide network(or host) scanning and they can not provide security test for IPsec-based secure platform. Moreover they can't be operated on IPv6 network. The proposed approach in this paper that evaluates security by sniffing, modifying, sending, analyzing packet by real time is a new method in the area of security test.

In this paper, we propose a rule-based approach to security test automation on IPv6 network. So, we design STRDL(Security Test Rule Description Language) to define security test rule and rule execution engine for executing the rules defined by STRDL for testing security of the IPv6 network, and we provide implementation details. As the proposed method is based on rules, it is able to automate the security test of the network and it can cope with new vulnerability easily without modifying the system. If its users get information about a new vulnerability,

they can edit security test rules by using STRDL that can handle the new vulnerability.

The paper is organized as follows. Section 2 presents about STRDL and section 3 presents architecture of the rule execution engine. Section 4 presents the examples of security test by using prototype system of rule execution engine. Finally the conclusion is given in section 5.

2 STRDL: Security Test Rule Description Language

In this section, we'll present about STRDL i.e Security Test Rule Description Language to define security test rule. Table 1 shows the list of statement and its grammar of the STRDL. The meanings of the sentences are as follows.

RULE NAME: To define Rule Name
DESCRIPTION: To describe the meaning of rule
START_EVALUATION: To inform starting of the test
END_EVALUATION: To inform end of the test
IF: IF statement
LOOP: LOOP statement. The keyword time is also defined in order that the loop can be repeated for designated time. (ex: LOOP time > 50ms : Repeat the loop for 50miliseconds)
SAVE: Process *query* from input DB(or file) and store the result in the output DB(or file) so that next test rule can use them.
CAPTURE: Capture packet by real time and store it into the output DB(or file). The processing of CAPTURE command includes sending the command to the agent that is installed in the target system that is specified in query and receiving and storing the results. The query of CAPTURE in table 1 contains the several query keywords. Currently we defined following query keywords.

- ip_packet : Capture only IP packet
- ip6 proto *protocol* : Capture the packet whose *nexthdr* value equals *protocol*
- ip6 protochain *protocol* : Capture the packet whose protocol chain contains *protocol*
- source_ip *source_address* : capture the packet whose source IP is equal to *source_address*
- dest_ip *destination_address*: : capture the packet whose destination IP is equal to *destination_address*
- icmp_type *type* : capture the icmp packet whose type is equal to *type*

SEND: Send packet that is stored in the file, specified in its filename, field to the target system
EDIT_PACKET: Edit packet that is stored in fromfile with *edit_option* and store it to the file, named tofile. Currently we defined following *edit_option* keywords.

- sa *new_sa* : replace source address of the packet with *new_sa*
- da *new_da* : replace destination address of the packet with *new_da*
- tc *new*_tc : replace traffic_class field of the packet with *new_tc*
- fl *new_fl* : replace flow_label field of the packet with *new_fl*

CHECK_PACKET: Analyze the meaning of the packet. It also checks that the packet is correctly applied to IPsec. And it is also used in extracting usable information from packet to modify the packet during test process. The detailed grammar and processing steps of CHECK_PACKET keyword are under defining.
CHECK_FILE: Check the file, named filename, whether it is NULL or not.
IPSEC_PROC: Apply IPsec to the packet, that is stored in fromfile, by referring to *ipsec_option* and store it to the file, named tofile. Currently we defined following *ipsec_option* keywords.

- RECAL_ICV *sa*: Recalculate ICV value by using predefined sa
- DECRYPT *sa*: Decrypt the packet by using predefined sa
- ENCRYPT *sa*: Encrypt the packet by using predefined *sa*

Before processing this keyword, user must set the *sa* value manually.
BREAK: Exit the loop.
PRINT: Display the message by using Graphic User Interface.
DELAY: Delay processing for time milliseconds.
COMMENT: Comment something using the symbol // or /* */

Table 1. Major keywords of STRDL

Keyword	Grammar
RULE_NAME	RULE_NAME : *strings*
DESCRIPTION	DESCRIPTION : *strings*
START_EVALUATION	START_EVALUATION
END_EVALUATION	END_EVALUATION
LOOP	LOOP *conditions* *statements* ...

	END LOOP

IF	IF conditions THEN
	statements..
	[ELSE]
	statements..
	ENDIF

conditions	*condition* {[AND | OR] *condition*}*
SAVE	SAVE (filename, *query*, DBname) SAVE (DBname, *query*, filename)
CAPTURE	CAPTURE (DBname, *query*) CAPTURE (filename, *query*)
SEND	SEND (filename, target)
EDIT_PACKET	EDIT_PACKET(fromfile, *edit_option*, tofile)
CHECK_PACKET	CHECK_PACKET(filename, *check_option*)
CHECK_FILE	CHECK_FILE(*filename*)
IPSEC_PROC	IPSEC_PROC(fromfile,*IPsec_option*,tofile)
BREAK	BREAK
PRINT	PRINT *display statement*
DELAY	DELAY *time*
COMMENT	// … or /* … */

3 Rule Execution Engine

3.1 Functional and Security Requirement

The security requirements at network layer are as follows [2].

Confidentiality: Confidentiality is the property of communicating such that the intended recipients know what was being sent, but unintended parties cannot determine it. For example, ensuring the secrecy of passwords when logging into a remote machine over the Internet. A mechanism commonly used for providing confidentiality is called encryption.

Connectionless integrity: Guarantee that the data does not get changed in transit. If you are on a line carrying invoicing data you probably want to know that the amounts and account numbers are correct and have not been modified by a third party.

Data origin authentication: Data origin authentication guarantees the claimed sender is in fact the actual sender.

Access control: Access control function controls user's access to a target system according to users' authority so that data security can be ensured in the target system.

Anti-replay: We need ways to ensure a datagram is processed only once, regardless of how many times it is received. I.e. it should not be possible for an attacker to record a transaction (such as a bank account withdrawal), and then by replaying it verbatim cause the peer to think a new message (withdrawal request) had been received.

3.2 The Architecture of Rule Execution Engine

In this section, we'll address each module in security rule execution engine. Figure 1 shows the architecture of the security rule execution engine and the test process. The system is divided into following parts - User interface Module, Rule Processing Module, DBMS (Data Base Management System), Agent module that is installed in target system. –

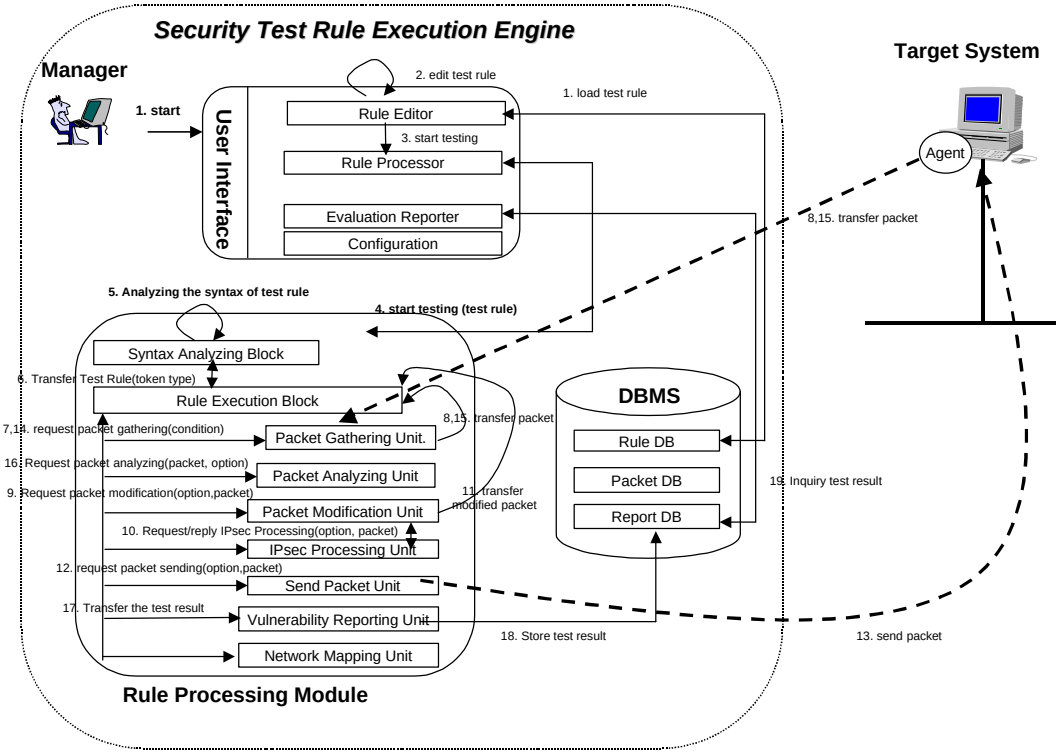

Fig. 1. Security Test Rule Execution Engine & test process

3.2.1 User Interface Module

Users may create, modify, select or delete the rule by using Rule Editor

in User Interface Module. And defined rule is stored in Rule DB in DBMS module. Rule Processor calls the Rule Processing Module for executing the defined rule. Evaluation Reporter displays the security test result that is stored in Report DB using Graphic User Interface. Configurator sets the various options of Security Test Rule Execution Engine, and it checks the status of target network and installs the agent to the target system. Figure 2 shows the installation process of agent by Configurator and packet gathering process by using installed agent.

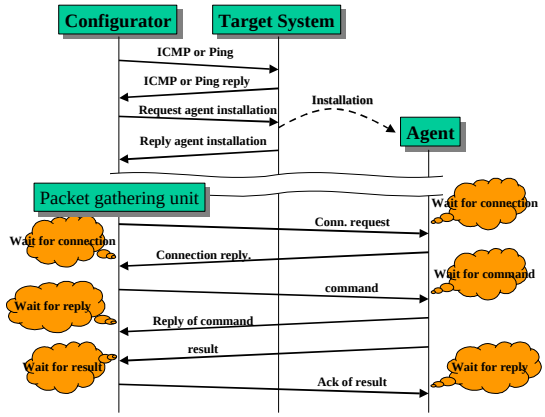

Fig 2. Installation process of agent

3.2.2 User Interface Module

Rule processing module comprised of Syntax analyzer block and Rule Execution block.

Syntax Analyzer Block: Syntax Analyzer Block in Rule Processing Module has the function of analyzing the syntax of the Rules that are selected for the security test from the Rule Evaluation DB that are selected. If it finds any syntax error during syntax analyzing process, it informs GUI module of this error message. Otherwise, Syntax Analyzer divides rule into tokens and transmits them to the Rule Execution Module.

Rule Execution Block: This Block executes the rules by using 5 execution units. The 5 execution units are as follows.

- *Packet Gathering Unit*: This unit has the function of sniffing packets, in accordance with the specified option, from the network, by cooperate with agent that is installed in target host. Currently, Packet Gathering Unit can sniff the packets generated by the protocols that are used by IPsec or IPv6 such as AH, ESP, ISAKMP, IPv6, ICMP and so on.
- *Packet Analyzing Unit*: It analyzes the packet content, and informs rule execution block of the analyzing results. The analyzing process contains the extraction of some packet information, to modify packet during test process.
- *Packet Modification Unit*: This unit has the function of modifying packets, in accordance with the specified rule. It modifies some fields of the packet. If it needs IPsec processing, it requests that to the IPsec processing unit.
- *Send Packet Unit*: This unit is used for sending a packet that is stored in its system to the target host by Raw Socket interface. Currently Send Packet Unit can send the IPv6 and IPsec packet such as AH, ESP, ISAKMP, IPv6, ICMP.
- *IPsec Processing Unit*: This unit is used for modifying packet by IPsec processing, and it is also used to examine whether packet contains valid IPsec processing data or not.

3.3 Agent

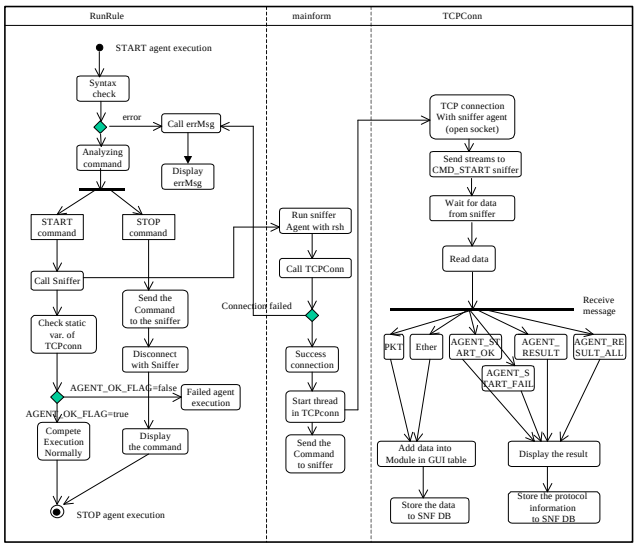

Fig 3. Processsing steps of agent execution

Agent sniffs the packets generated by the protocols that are used by IPsec and IPv6 such as AH, ESP, ISAKMP, IPv6, ICMP, TCP, and send them to the Security Rule Execution Engine in order to check whether security systems provide proper IPsec services to the network or not. When the connection is established with test engine, agent waits a command from the rule execution engine. The Rule Execution Engine can send four types of command - START, STOP, HALT, RESUME – and options to the agent. The keywords that are used by option are ip_packet, ip6 proto, ip6 protochain, source_ip, dest_ip and icmp_type. The meaning of this option was described

in section 2. Figure 3 shows the flow of agent execution. Figure 4 shows the content of sniffed ESP packet by agent by real time.

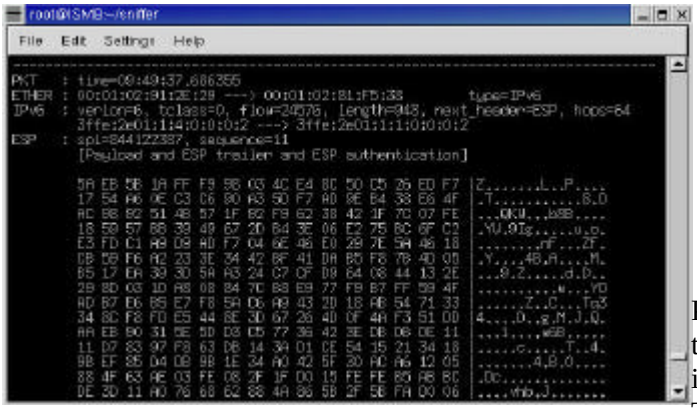

Fig 4. Sniffed ESP packet by real time

3.4 DBMS

The Databases used by Security Rule execution Engine are Packet DB, Rule DB and Result DB The Rule DB stores the rule that is used for the security test. Table 2 shows the field of Rule DB.

Table 2. Rule DB

Field Name	Type	Description
RULE_ID	INTEGER	Rule Identification Number
RULE_NAME	CHAR	Rule Name
DESCRIPTION	CHAR	Description of the Rule
RULE	MEDIUMTEXT	Defined Rule which is described in STRDL
DateTime	DATE / TIME	Date and Time

Proposed Engine provides Database Access routine by using JDBC. The algorithm to set the security test rule is as follows.

```
begin RSU module {
switch(func) {
    case SelectRule :sRules = RuleDB.select(expr);
          if (sRules != 0)  DisplayRuleTitle();
          break;
    case NewRule : if(CheckRuleSyntax(newRule) == OK)
                       RuleDB.insert(newRule);
                   else  DisplayErrMsg(badRuleSyntax);
                   break;
    case UpdateRule :if (CheckRuleSyntex(rule) == OK)
                        RuleDB.update(rule);
                   else  DisplayErrMsg(badRuleSyntax);
                   break;
    case RemoveRule :  RuleDB.delete(rule);  break;
    case DetailedInfo : DisplayDetailedRuleInfo(rule);
                        break;
   case ShowEnvVar:          DisplayEnvVar();
                             break;
    case ChangeEnvVar:       Update(envVar, newVal);
                             break;
    default:                 DisplayErrMsg(badCommand);
                             break;
    }
}
```

Table 3. Report DB

Field Name	Type	Description
HOSTID	INTEGER	Target Host ID
IP	STRING	Target Host IP address
RULE_LIST	ARRAY of INTEGER	Applied Rule ID(s)
RESULTS	ARRAY of CHAR	Each Report of rule execution
DateTime	ARRAY of DATE/TIME	Date and Time

Packet DB stores the various kinds of protocol data that the agent transmits. The protocol which is stored in the each table is the Ethernet, ARP, IPv6, AH, ESP, TCP, UDP, ICMP and ISAKMP.

The Report DB stores the result of security test. The values stored in Report DB are displayed during the test process and sometimes later it is used to verify the test result. Table 3 shows the field of Report DB.

4 Security Test

The prototype of the Security Rule Execution Engine has been implemented in Java and C language and the Database has been implemented in JDBC. And we implemented packet-sniffing component in agent with libpcap for the IPv6 network, a system independent interface for user-level network packet capturing developed by the Network Research Group at the Lawrence Berkeley Laboratory. This system operated on Windows and Linux platform. The method of evaluating the four representative security services that must be provided at ip layer can be summarized below.

Confidentiality: Sniff any ESP packet, whose source or destination IP equals target host IP, from network, and check the packet whether it is readable or not, and try to decrypt it by using arbitrary key.

Data Origin Authentication: Sniff AH or ESP packet whose destination IP equals target host IP, from network by real time, and modify source IP address in sniffed packet, and try to send it to the target host again, and see if the host responds to this trial or not.

Access Control: Generate ICMP packet with AH or ESP by using arbitrary key, and send it to the target host, and see if the host responds to this trial or not.

Connectionless Integrity: Modify arbitrary field of sniffed packet, and re-calculate ICV(Integrity Check Value) , and replace the ICV with the newly calculated

ICV, and send it to the target host, and see if the host responds to this trial or not.

Anti-replay: Monitor SN(Sequence Number) value in sniffed AH/ESP packet header, and guess and change the SN value of the packet, and send it to the target host, and see if the host responds to this trial or not.

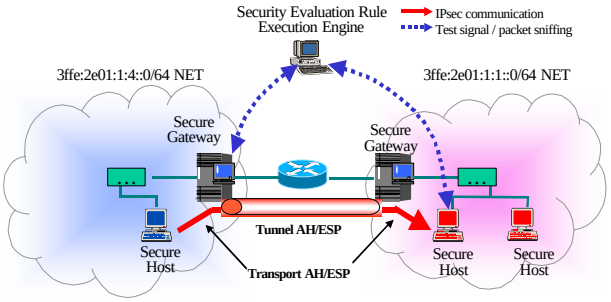

Fig 5. Testbed

By using our Security Rule Execution Engine, we tried to evaluate IPv6 IPsec Platform developed by ETRI in Korea for some test items. The test environment is shown in figure 5. Figure 6 shows the sample rule that is described using STRDL for evaluating AH Connectionless Integrity function.

The meaning of the rule displayed in Figure 6 is as follows. Sniff and store the AH+ICMP packet whose destination is target host, and modify ICV value, and send it to the target host. And see what happens to the target host. If target host responds to this trial, the Security Test System displays the warning message "It can't support AH Connectionless Integrity". Otherwise, the Security Test System displays the message "It can support Connectionless Integrity" Figure 7 shows the result of this test. As a result of this test item, IPv6 IPsec Platform support AH Connectionless Integrity.

```
RULE_NAME: AH CI
DESCRYPTION : AH Connectionless Integrity Test
START_EVALUATION
    LOOP  time < 300ms
      IF (CAPTURE (ci_test, ip6 protochain icmp and ah and
                                icmp_type 8))
        IPSEC_PROC(ci_test, RECAL_ICV sa, ci_test2);
        BREAK;
      ENDIF
    END LOOP
    IF (CHECK_FILE(ci_test2))  SEND_PACKET(ci_test2,
                                3ffe:2e01:1:4::2);
    ELSEIF    PRINT("Error: Failed to packet capturing");
    ENDIF
    LOOP time < 30ms
      IF(CAPTURE (ci_result, ip6 protochain icmp and source_ip
                  3ffe:2e01:1:4::2 and icmp_type 0))
        PRINT("It can't support AH Connectionless Integrity");
        BREAK
      ENDIF
    END LOOP
    IF NOT CHECK_FILE(ci_result)
        PRINT("It can support AH Connectionless Integrity");
    ENDIF
END_EVALUATION
```

Fig 6. The test rule of AH Connectionless Integrity

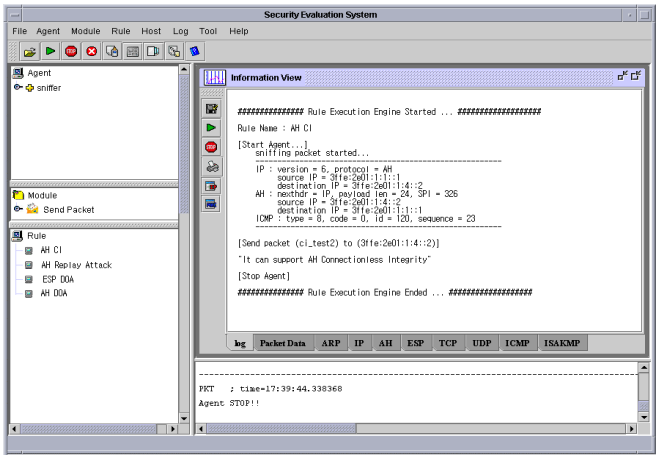

Fig 7. Test results of AH Connectionless Integrity test

4 Conclusion

In this paper, we design STRDL(Security Test Rule Description Language) to define security test rule and rule execution engine for evaluating security of the IPv6 network. The proposed method is based on rules, so it is able to automate the security test of the network and it can cope with new vulnerability easily without modifying the system. If its users get information about new vulnerability, then they can edit security test rules by STRDL that can handle the new vulnerability. Currently, we defined some security test rule by using STRDL so that user can select such a rule or edit the rule by himself.

The prototype of Security Rule Execution Engine has been implemented in Java and C language and the Database has been implemented in JDBC. And we implemented packet-sniffing component in agent with libpcap for the IPv6 network. By using our Security Rule Execution Engine, we tried to evaluate IPv6 IPsec Platform developed by ETRI for some test items. This rule execution engine can be applied to any IPv6 security systems because it exists independently and it is up to standard document such as IETF RFC 2460, 2401, 2402, 2406, 2407 and so on.

The characteristics of the STRDL and Rule Execution Engine that is presented in this paper are as follows.

- The proposed approaches in this paper that evaluates security by sniffing, modifying, sending, analyzing packet by real time is a new method in the area of security test.
- It is operated on IPv6 network
- It is able to collect and analyze various protocol packets from network
- It is rule based test system, so it is able to automate the security test of the network and it can cope with new vulnerability easily without modifying the system. If its users get information about new vulnerability, then they can add Security Test Rules that can handle the new vulnerability.
- The grammar of the security test rule is simple, so we can define test scenario easily.

In the future work, we'll extend the STRDL and test engine so that it will cover various security requirements.

References:

[1] S.Deering, R.Hinden, Internet Protocol, Version 6(IPv6) Specification, RFC2460, Dec. 1998

[2] White Paper, IPv6 to IPv4 is Not Merely 50% More, Ezchip Technologies.

[3] IETF, http://www.ietf.org

[4] N.Doraswamy and D.Harkins, IPsec : The New Security Standard for the Internet, Intranets, and Virtual Private Networks, Prentice Hall, 1999

[5] S.Kent and R.Atkinson, Security Architecture for the Internet Protocol, RFC2401, Nov. 1998

[6] S.Kent and R.Atkinson, IP Authentication Header, RFC2402, Nov. 1998

[7] S.Kent and R.Atkinson, IP Encapsulating Security Payload, RFC2406, Nov. 1998

[8] D.Harkins, D.Correl, Internet Key Exchange, RFC2409, Nov. 1998

[9] KAME, http://www.kame.net

[10]USAGI, http://www.linux-ipv6.org/

[11] ISS, ISS Internet Scanner, http://www.iss.net/

[12] Cisco Scanner, http://www.cisco.com/univercd/cc/td/doc/pcat/nssq.htm

[13] LANguard Network&Port scanner, http://www.gfi.com/languard/lanscan.htm

[14] S.Garfinkel and G.Spafford, Practical UNIX and Internet Security 2nd edition, 1996, O'Reilly

[15] M.Y.Lee, Internet Security – Cryptographic principles, algorithms and protocols, 2003, WILEY