International Journal of Knowledge-Based Intelligent Engineering Systems, Vol. 5, No. 4, October 2001

Pakemon – A Rule-based Network Intrusion Detection System

Keiji Takeda † and Yoshiyasu Takefuji ‡
† Graduate School of Media and Governance
‡ Faculty of Environmental Information
Keio University
Endo 5322
Fujisawa Kanagawa 252-8520
Japan

Abstract

This paper describes design and implementation of "Pakemon", a system that detects network intrusions and their attempts by stealthy monitoring network traffic. Through the design of the system, knowledge about intrusive network traffic was formulated as suspicious patterns. An operation model for network security by knowledge sharing was proposed. Artifacts derived from this work have been distributed through the Internet and widely used for real-world practices of intrusion detection. Rules to detect intrusions with this system have been released and shared by third parties. By using this system, users can implement security mechanisms that enable administrator to respond to network intrusions. The system works as a supplement of the traditional preventive security mechanisms such as identification and authentication, access control and encryption.

1 Introduction

Use of computer network, especially the Internet, is increasingly important in the modern society. Today, globally interconnected networks are critical elements of social infrastructure. In it's definition, computer security is based on the realization of confidentiality, integrity, and availability in computer systems [1]. To achieve the objectives of computer security, various preventive mechanisms, such as identification and authentication, access control, and encryption have been utilized [2].

Network intrusions are any actions aiming to compromise the goal of computer security from both internal and external sources of networks. Such intrusions may cause information leakage, deletion or modification of resident data or software, denial of service, unauthorized use of systems and so on [3].

It has been understood in the computer security community that virtually all computer and communication systems are not immune to intrusions [4]. Computer systems sometimes have flaws. Finding and fixing all of them are not feasible technically and economically. Furthermore, even with a system that has strong security mechanisms, internal abuses of the system are still difficult to prevent [5].

In response to these difficulties, surveillance mechanisms of computer activity have been proposed [6]. Since audit-trail records can be used to identify system troubles, misuse of systems and their attempts, automated audit trail analysis has been utilized to detect anomaly in the audit data [7][8]. Most early works to detect network intrusion were anomaly detection approaches with some statistical algorithms [9] [10]. Heberlein et.al. [11] proposed an approach that applies the anomaly detection mechanism to network traffic monitoring. The problem of detecting suspicious activities on networks has been termed as network intrusion detection [12].

Data sources for most intrusion detection system are local audit trails or network traffic. Methodology of analysis can be divided into two categories: anomaly detection and misuse detection. Anomaly detection refers to as detecting intrusions based on anomalous behavior of systems or users. In contrast, misuse detection, alternatively called rule-based detection refers to as detecting intrusions with pre-defined patterns of the behavior [13][14].

Pakemon, the system that we describe in this paper, is implemented as a network-based and rule-based intrusion detection system. A number of similar systems have existed so far. Thus the objective of this paper is not to propose novel methodology for intrusion detection but to present a detailed description of techniques to improve accuracy and performance of the system.

2 Network-based Intrusion Detection Systems

In this paper, we focus on network and rule-based intrusion detection approach. The reason is that, although there are many issues to be solved on the approach it is widely accepted as a practical measure to detect network intrusions in real world environment. Advantages of network-based intrusion detection approach include detection in real-time manner and independency from operating systems to protect [11].

A fault of the current network and rule-based intrusion detection approach has been published as "insertion and evasion" techniques [15]. Many intrusion detection systems fail to detect under such evasion techniques. The evasion is done by making differences between the traffic that an intrusion detection system observes and the traffic that the final destination (target host) receives. This technique applies fragmentation, disorder, duplication, overlap, insertion, evasion, and de-synchronization at IP (Internet Protocol) or TCP (Transfer Control Protocol) of the traffic. For example, IP has fragmentation and de-fragmentation functions in the protocol. A single IP packet can be broken up into multiple fragmented packets. These fragmented packets will be re-constructed at the destination host. If an intrusion detection system perform only simple pattern matching without de-fragmentation, it fails to match the traffic to an attack signature for complete single IP packet. The same technique can also be applied to the TCP layer. Fragmented IP packets and TCP segments have their own id number in it. Scramble, disorder, duplicate or overlap of them makes intrusion detection systems difficult to complete pattern matching. Each operating system has different interpretation of network protocol. Especially, when operating systems receive irregular network packets, they handle the packets in different ways. Thus, it is difficult for an intrusion detection system to handle such traffic exactly in the same way without having network protocol stacks of all types and versions of operating systems.

Another problem on the current intrusion detection approach are acquisition and management of detection rule-set. Since new exploits are found almost every day, rule-set must be updated frequently to catch up with the new types of attack. In case of commercial intrusion detection systems, this update is done by reinstallation of entire programs with distribution media such as CD-ROMs.

3. System Design and Implementation

3.1 Overview of the system

In response to the problem on network-based intrusion detection, an intrusion detection system named "pakemon," has been developed. The system monitors Ethernet in promiscuous mode, and detects predefined attack signatures on the wire with reasonable computational resource consumption. At the time of detection the system generates an alert and/or logs the detail information of observed traffic. The system is implemented and run on various Unix variants.

The developed system is compliant to the "insertion and evasion" technique against network-based intrusion detection system. Features of the system are as follows. First, it does full TCP/IP protocol analysis in the same way as Linux kernel 2.0.37 does. This interpretation is done by utilizing a library named "libnids[16]." The library was derived from TCP/IP protocol stack of Linux's kernel source code. Thus, packets received by this intrusion detection system are processed like as a Linux system does. This makes the system immune to the "insertion and evasion" techniques against Linux platform.

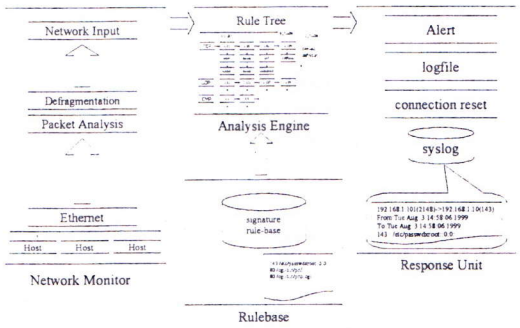

Figure 1. Overview of Pakemon

Another feature of the system is about its development and operation model. The source code of the program has been publicly available on the Internet so that users are able to access its source code. Users can write their own detection rule-sets, can tune the rule-sets, and distribute their original rule-sets to the entire community. As a result, many attack signatures have been designed, provided and shared by users. This process will solve the problem of supply of network intrusion detection. The system has been widely used in real operational environment through the world and supported by several security research groups so that the community supplies rule-sets.

3.2 System Architecture

The system is constructed from four modules, "network monitor," "analysis engine," "rule-base," and "response unit." The "network monitor" captures network traffic and reassembles fragmented IP and TCP traffic. The processed traffic data is sent to "analysis engine." The analysis engine generates search trees from a rule-base at initialization of the system. The engine examines if observed traffic matches any nodes of the search trees. When one or more signatures are matched, a response unit processes a predefined action for the signature. The responses include alerting, logging to a file or to "syslog" service, and shutting down correspond TCP connections.

3.3 Network Monitor

A key function of network monitor is to provide network traffic data captured from connected wire for the analysis engine. It captures packets on a network; reassembles fragmented IP packets; and reconstructs broken TCP streams. For the function

of the component, two libraries "libpcap[17]" and "libnids" are used in pakemon (Figure 2). These libraries are system-independent and run on various Unix variants.

"Libpcap" provides a system-independent interface for user-level packet capturing. The library can be used with "Berkeley Packet Filter" commands that filter out unnecessary traffic into the kernel. It can reduce traffic road to process.

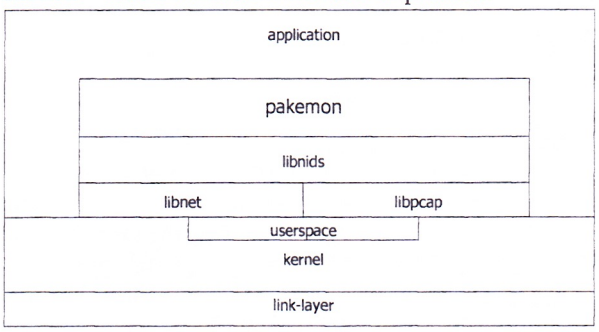

Figure 2. Runtime Structure

Libnids reassembles fragmented IP packets and TCP segments in the same way as Linux 2.0.36 kernel does. This reassembly function of the library makes the system immune to the "insertion and evasion attack" against intrusion detection systems.

By using these two libraries "pakemon" receives all network traffic on a network in promiscuous mode without interfering communication under the monitor (Figure 3). Since the monitor does not transmit any packets to the interface attackers cannot aware the existence of the monitor.

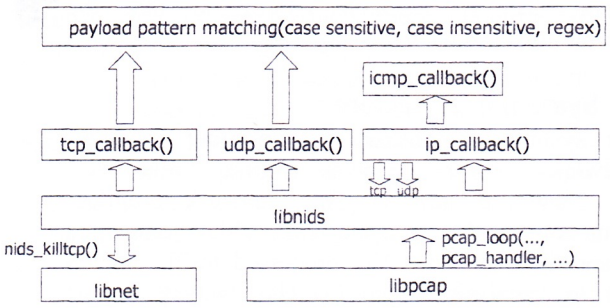

Figure 3. Raw Packet Processing in Pakemon

3.4 Analysis Engine

Traffic data acquired through the network monitor is sent to the analysis engine. The function of analysis engine is to examine if observed packets match predefined pattern of interest.

At the initialization time of the system, analysis engine reads predefined attack rule-sets stored in a rule-base. Then it organizes search trees for detection rules in two-dimensional linked lists. The tree structure is depicted in Figure 4. The tree has two types of nodes, service node and content node. Service nodes have information on service types, port numbers in TCP and UDP and type of service in ICMP protocol. The content node includes binary data patterns to search. The service nodes are lined horizontally and content nodes are lined vertically in the figure. Every time the analysis engine receives newly arrived traffic, it searches for matched service node from correspond tree for the protocol. Then the traffic data is examined if it contains patterns stored in contents nodes. For example, when a TCP segment to port 80 arrived, firstly it is merged with stored buffer traffic. Then the data is examined with the TCP tree of detection rule. Then the port number is compared with the service nodes. When a service node contains number 80 as a service port then linked content nodes are examined. Each patterns in contents node is examined if the traffic data contains the pattern in it. The pattern match is done by Boyer-Moore algorithm [18].

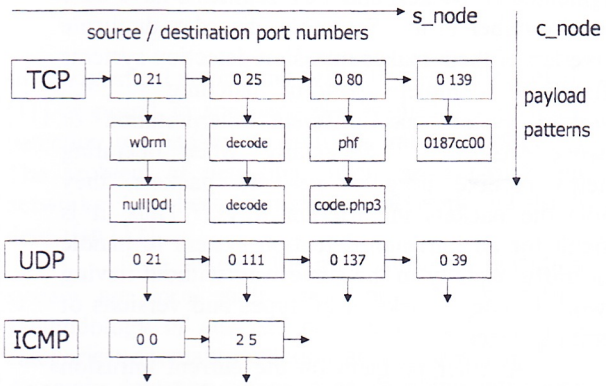

Figure 4. Search Tree in Analysis Engine

3.5 Rule-base

Rule-base is implemented as a single file that contains rules to detect attacks in the current version of pakemon. A rule is a description of patterns contained in network traffic. The format of rule is shown in Figure 5. and example of a rule-set is presented in Figure 6.

Rule format

name	protocol	src	dest	pattern

name : name of the signature

protocol : protocol name (tcp, ip, icmp)

src, dest: source/destination port numbers for TCP or UDP, type/code for ICMP

pattern : data pattern to detect

"..." case sensitive

'...' case insensitive

|... | binary expression

<...> regular expression

Figure 5. Rule-set Format

FTP-exploit1	tcp " 21 " 50 57 44 0A 2F 69 "
SMTP-exploit1	tcp " 25 'Croot 09090909090909 Mprog.P=/bin'
SMTP-exploit2	tcp " 25 'rcpt to 3a207c sed '1,/^$/d' 7c'
SMTP-exploit3	tcp " 25 'mail from 3a20227c'
SMTP-exploit4	tcp " 25 'Croot 0d0a Mprog, P=/bin/'
SMTP-exploit6	tcp " 25 'rcpt to 3a `decode`'
SMTP-exploit7(CVE-1999-0204)	tcp " 25 ' 0a C 3a daemon 0a R'
SMTP-exploit11(CVE-1999-0204)	tcp 113 25 ' 0a D"
DNS-named-exploit	tcp " 53 " CD 80 E8 D7 FF FF FF "
DNS-zone-transfer	tcp " 53 " 01 00 00 01 00 00 00 00 00 00 "
IIS-administrator.pwd	tcp " 80 '/_vti_pvt/administrators.pwd'
IIS-ISM.DLL-Exploit	tcp " 80 "%20%20%20%20%20.htr'
CGI-phf(CVE-1999-0067)	tcp " 80 '/cgi-bin/phf'
HTTP-ApacheDOS	tcp " 80 " 2f2f2f2f2f2f2f2f2f "
HTTP-PiranhaPasswd.php3	tcp " 80 "passwd.php3"
RPC-portmap-request-rusers	udp " 111 " 01 86 A2 00 00 "
IMAP-exploit1	tcp " 143 " bf ff "
IMAP-exploit2	tcp " 143 " E8 C0 FF FF FF "
BD-BackOrifice	udp " 31337 ""

Figure 6. Example of a Rule-set

A rule-base contains arbitrary number of attack signatures. The signatures are read when the system is initialized. Then the system organizes a detection rule tree for each layer 3 protocol (TCP, UDP, ICMP). Payload patterns can be defined both in text form, and in binary form. Matching categories for each payload can be chosen from text case sensitive match, text case insensitive match and regular expression match.

3.6 Response-Unit

Response-unit has four functionalities. The unit can issue an alert, log record to a file, send a signal to syslog, and reset corresponding TCP connections. These functionalities are activated by command options. When a signature is detected an alert is issued and the record is stored in the log. An example of out put log is shown in Figure 7.

Alert Log Example

```
TCP 194.168.10.12(1032)->192.168.10.5(80) Thu Jan 11 19:40:51 2001
CGI-handler(CVE-1999-0148)

TCP 194.168.10.12(1033)->192.168.10.5(80) Thu Jan 11 19:40:53 2001
CGI-test-cgi

TCP 194.168.10.12(1034)->192.168.10.5(80) Thu Jan 11 19:40:55 2001
CGI-nph-test-cgi(CVE-1999-0045)

TCP 194.168.10.12(1035)->192.168.10.5(80) Thu Jan 11 19:40:57 2001
CGI-phf(CVE-1999-0067)

Thu Jan 11 22:59:20 2001
Scan from 194.168.10.12 scan type: SYN

Fri Jan 12 07:25:15 2001
Invalid TCP header,from 194.168.10.12 to 192.168.10.5

Fri Jan 12 09:02:00 2001
Too much data in TCP receive queue,from 194.168.10.12:80 to
192.168.10.5:3983
```

Dump Log Example

```
TCP 194.168.10.12(1032)->192.168.10.5(80) Thu Jan 11 19:40:51 2001
CGI-handler(CVE-1999-0148) (TCP 4294967295:80)
47 45 54 20 2f 63 67 69 2d 62 69 6e 2f 68 61 6e   GET /cgi-bin/han
64 6c 65 72 20 48 54 54 50 2f 31 2e 30 0d 0a 41   dler HTTP/1.0..A
63 63 65 70 74 3a 20 69 6d 61 67 65 2f 67 69 66   ccept: image/gif
2c 20 69 6d 61 67 65 2f 78 2d 78 62 69 74 6d 61    image/x-xbitma
70 2c 20 69 6d 61 67 65 2f 6a 70 65 67 2c 20 69   p, image/jpeg, i
6d 61 67 65 2f 70 6a 70 65 67 2c 20 2a 2f 2a 0d   mage/pjpeg, */*.
0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 56 6f 69   .User-Agent: Voi
64 45 59 45 20 43 47 49 20 73 65 63 75 72 69 74   dEYE CGI securit
79 20 73 63 61 6e 6e 65 72 0d 0a 48 6f 73 74 3a   y scanner..Host:
20 31 39 32 2e 31 36 38 2e 31 30 2e 35 0d 0a 50    192.168.10.5..P
72 61 67 6d 61 3a 20 6e 6f 2d 63 61 63 68 65 0d   ragma: no-cache.
0a 0d 0a

TCP 194.168.10.12(1033)->192.168.10.5(80) Thu Jan 11 19:40:53 2001
CGI-test-cgi (TCP 4294967295:80)
47 45 54 20 2f 63 67 69 2d 62 69 6e 2f 74 65 73   GET /cgi-bin/tes
74 2d 63 67 69 20 48 54 54 50 2f 31 2e 30 0d 0a   t-cgi HTTP/1.0..
41 63 63 65 70 74 3a 20 69 6d 61 67 65 2f 67 69   Accept: image/gi
66 2c 20 69 6d 61 67 65 2f 78 2d 78 62 69 74 6d   f, image/x-xbitm
61 70 2c 20 69 6d 61 67 65 2f 6a 70 65 67 2c 20   ap, image/jpeg,
69 6d 61 67 65 2f 70 6a 70 65 67 2c 20 2a 2f 2a   image/pjpeg, */*
0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 56 6f   ..User-Agent: Vo
69 64 45 59 45 20 43 47 49 20 73 65 63 75 72 69   idEYE CGI securi
74 79 20 73 63 61 6e 6e 65 72 0d 0a 48 6f 73 74   ty scanner..Host
3a 20 31 39 32 2e 31 36 38 2e 31 30 2e 35 0d 0a   : 192.168.10.5..
50 72 61 67 6d 61 3a 20 6e 6f 2d 63 61 63 68 65   Pragma: no-cache
0d 0a 0d 0a
```

Figure 7. Output Log Example

4 Evaluation

4.1 Evaluation Methodology

To evaluate performance of the system, experiments were conducted both in simulated environment and in real-world environment. Evaluation of intrusion detection system is not straightforward since the expected behavior of intrusion detection system varies from environment to environment. The evaluation methodology of intrusion detection system itself can be a research subject. Even though research projects for such evaluation has conducted, no common criteria to evaluate intrusion detection system has not been established yet [19]-[21].

In this evaluation, we limited the goal of network-based intrusion detection system only to identify pre-defined event. This means that, a network intrusion detection system is expected to have ability to detect all known attacks. To examine our system an experiment was conducted in following steps.

A benchmark performance test for rule-based network intrusion detection systems has been conducted. A tool "nidsbench" [22] were used in the experiment. The purpose of this benchmark is to present load performance and compliance to various evasion techniques. In this experiment, the system was compared to another popular intrusion detection system named "snort." "Snort" is currently a leading open source intrusion detection system available on the Internet [23].

The topology of the experimental network is constructed as shown in Figure 8. The network is isolated from outside. To show the compliance of the system against to evasion techniques an experiment, in which a simulated attacker send crafted HTTP requests to a HTTP server, was conducted. In this experiment, a special router "Fragrouter" was used to split, disorder, and de-synchronize the traffic. As the attack-side, a vulnerability scanner that looks for vulnerable services on a HTTP server was used. In this experiment, 119 well-known vulnerabilities on a HTTP server were scanned with 5 evasion techniques. Other evasion techniques are to change normal HTTP request for attacks into encoded ASCII code that destination HTTP server might be able to decode but some intrusion detection system cannot.

The HTTP server was under the monitor of the intrusion detection system and the number of detections was counted. Both intrusion detection systems were run on Pentium III processor with 512MB memory with Free-BSD Operating Systems. Since the rule-set to detect the scan was given to the systems in advance, all attempts in the scan must be detected.

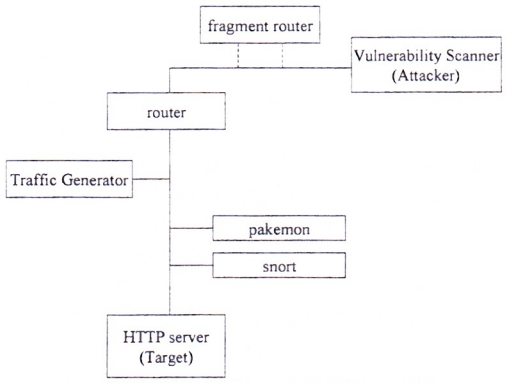

Figure 8. Simulated Network

Then, to investigate the load performance of the system, a program named "tcpreplay," a program that can replay recorded traffic with arbitrary transfer speeds, was used to generate load traffic. Clear traffic of HTTP (Hyper Text Transfer Protocol) was artificially generated with an automated download tool to dump files to replay. Vulnerability scan, that simulate intrusions, was performed against the HTTP server with the simulated load traffic.

Finally, we have placed a system at a perimeter of an operating sub-network in a campus network in which there is a web site containing our research materials and 3 client users exist in average. This experiment was conducted to examine the stability of the system.

4.2 Result

Figure 9. is the result of the experiment to test compliance to evasion techniques. In the result, pakemon present the ability to handle various traffic-based evasion techniques such as fragmentation, disordering and de-synchronization. In contrast "snort" failed to process these crafted traffic. However, pakemon failed to detect attacks with %encoded URLs (Uniformed Resource Locators), the reason for this is that current version of pakemon does not decode encoded URLs. (The function will be implemented in the future version.) Dot-insertion technique place "/./" description in the URL request of HTTP header. This will nullify simple pattern matching since it changes the form of request. Both intrusion detection systems occasionally detect these attacks only when the rest of the part of request matched rules they had.

Figure 10 shows degradation of the detection rate by load traffic. Since network intrusion detection have the heavy processing load to analyze packets they received the load traffic should not be majored by transfer rate presented in bits per second (bps) but should be majored by frames per second (fps). Pakemon reconstruct IP packets and TCP segments. Thus, it should cause heavy calculation load. On the other hand snort does not do the reconstruction. Then calculation load must be lower than pakemon. The result shows that pakemon performs packet reassembly with reasonable degradation of detection performance.

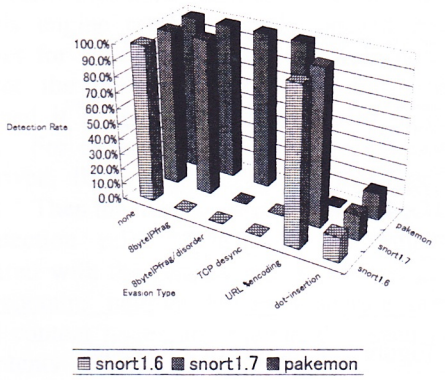

Figure 9. Detection Rate by Evasion Type

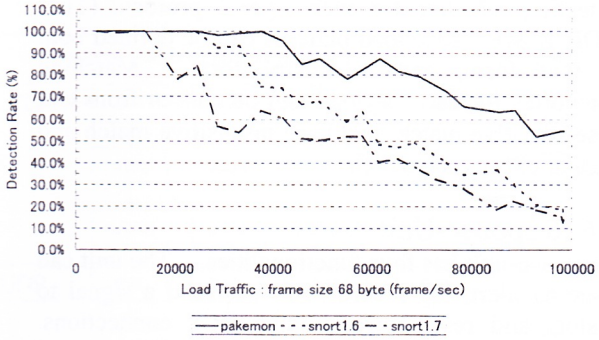

Figure 10. Detection Rate by Load Traffic

In figure 11, computational resource utilization at the monitoring for each detection systems are shown in the form of percentage of processor time it used. This also represents that pakemon does de-fragmentation with reasonable resource consumption.

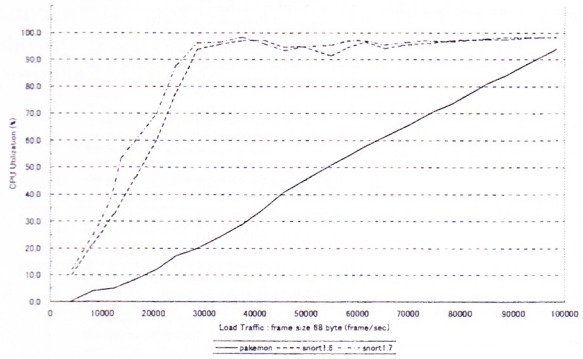

Figure 11. CPU Utilization by Load Traffic

In the real-world environment, it was confirmed that the system is able to operate continuously for the duration of more than 19 days without any trouble and degradation of performance. In 19 days period, 386292.88 IP frames/day were

received on average. The distribution of the TCP, UDP and ICMP protocols was 38.72%(149557.25 frames/day), 29.16%(112642.51 frames/day) and 0.08%(319.25 frames/day) respectively. This result is given in Table below.

Protocol	Frames	per day	%
IPs	11395640	386292.88	100.00
TCP	4411939	149557.25	38.72
UDP	3322954	112642.51	29.16
ICMP	9418	319.25	0.08
OTHER	3651329	123773.86	32.04

Jan 16 2001 10:41- Feb 4 2001 23:39 (19.5 days)

Table. Number of Packets processed in Real-world Environment

5. Conclusion

In this paper, design and implementation of a rule-based intrusion detection system is described. This paper introduces an approach to improve current intrusion detection systems. And quantitative evaluation was presented to show the effectiveness of the approach. This would be useful for researchers to understand issues and limitation on the technology and can be used as a case study of an implementation of such systems. The system, "pakemon" introduced in this paper is publicly available at "http://www.sfc.keio.ac.jp/~keiji/ids/pakemon/" .

References

[1] D. Russell and G.T. Gangemi Sr. Computer security basics. O'Reilly & Associates, Inc., Sebastopol, CA., December 1991.

[2] S. Garfinkel and G. Spafford. Practical Unix and Internet Security. 2nd Edition. O'Reilly and Associates, Inc., Sebastopol, CA. 1996.

[3] P. Neumann and D. Parker. A summary of computer misuse techniques. Proceedings of National Computer Security Conference, pp.396-407, 1989.

[4] S. M. Bellovin. Security problems in the TCP/IP protocol suite. Computer Communication Review, Vol.19, No.2, pp.32-48, April, 1989.

[5] R. Power. 1999 CSI/FBI Computer crime and security survey. Computer Security Journal, Vol.15, No. 2, Computer Security Institute, San Francisco, CA. 1999.

[6] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P Anderson Company, Fort Washington, PA. April 1980.

[7] D. E. Denning, An intrusion detection model. IEEE Transactions on Software Engineering, Vol.SE-13, No.2, pp.222-232, IEEE Computer Society Press. Los Alamos, CA., February, 1987.

[8] D. E. Denning and P. G. Neumann. Requirements and Model for IDES - A Real-Time Intrusion Detection System. Technical report, Computer Science Laboratory, SRI International, August 1985.

[9] S. E. Smaha. Haystack :an intrusion detection system. Proceedings of Fourth Aerospace Computer Security Applications Conference, pp.37-44, 1988.

[10] H. S. Javitz, A. Valdes. The SRI IDES Statistical Anomaly Detector. Proceedings of the IEEE Symposium on Security and Privacy. 1991.

[11] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood and D. Wolber. A network security monitor. Proceedings of the IEEE Symposium on Security and Privacy, IEEE Press, 1990.

[12] B. Mukherjee, L. T. Heberlein, K. N. Levitt. Network intrusion detection. IEEE Network, Vol.8, No.3, pp.26-41, 1994.

[13] S. Kumar and E. H. Spafford. A pattern matching model for misuse intrusion detection. Proceedings on Large-Scale Digital Calculating Machinery, pp. 141-146. Harvard University Press, Cambridge, MA., 1994.

[14] S. P. Shieh and V. D. Gligor. On a pattern-oriented model for intrusion detection. IEEE Transactions on Knowledge and Data Engineering, Vol.9, No.4, pp.661-667, IEEE. July/August 1997.

[15] T. Ptacek and T. Newsham. Insertion, Evasion, an Denial of Service: Eluding Network Intrusion Detection. Technical reports, Secure Networks, Inc., January 1998.

[16] R. Wojtczuk. Libnids. http://www.packetfactory.net/Projects/Libnids/

[17] libpcap http://www.tcpdump.org/

[18] R. S. Boyer and J. S. Moore. A fast string searching algorithm. Communication of the ACM, Vol.20, No.10, pp.762-772. 1997.

[19] J. Puketza, K. Zhang, M. Chung, B. Mukherjee, R. A. Olsson. A methodology for testing intrusion detection systems. IEEE Transactions on Software Engineering, Vol.22, No.10, pp.719-729, IEEE Computer Society Press, Los Alamos. 1996.

[20] R. Durst, T. Champion, B. Witten, E. Miller, L. Spagnuolo, Testing and evaluating computer intrusion detection systems, Communication of the ACM Vol.42, No.7, pp.53-61, ACM. 1999.

[21] J. McHugh. The 1998 Lincoln Laboratory IDS Evaluation A Critique. Proceedings of Third International Workshop Recent Advances in Intrusion Detection. H. Debar, L. Me, And F. Wu (Eds.): RAID 2000, LNCS 1907, pp.145-161, Springer-Verlag, Berlin Heibelberg. 2000.

[22] Anzen Computing. nidsbench a network intrusion detection system test suite. 1999. http://www.anzen.com/research/nidsbench/

[23] M. Roesch, et.al. snort - The Lightweight Network Intrusion Detection System. 2001. http://www.snort.org/