A Proposal of Protocol and Policy-Based Intrusion Detection System

Tatsuya Baba and Shigeyuki Matsuda

Research and Development Headquarters, NTT Data Corporation
Kayabacho Tower, 1-21-2, Shinkawa, Chuo-ku, Tokyo 104-0033, Japan
{babatt, matsudasg}@nttdata.co.jp

ABSTRACT

Currently, intrusion detection systems (IDSs) are widely deployed in enterprise networks for detecting network attacks. Most existing commercial IDSs are based on misuse detection model. In misuse detection, although known attacks can be detected, unknown ones cannot be detected because attack signatures for unknown attacks cannot be generated.
In this paper, we propose a method for detecting network attacks including unknown ones against servers such as web servers, mail servers, FTP servers, and DNS servers, using protocol specifications and site access policy. Furthermore, we propose a method to predict damage from detected attacks using neural networks.

Keywords: Intrusion Detection, IDS, Network Security, Neural Network, Damage Prediction

1. INTRODUCTION

As the Internet is becoming an important part of the business and social activities, damage from network attacks is becoming a serious issue. Firewall is generally used to defend the site from the attacks. However, there are attacks that cannot be defended with firewall such as CGI attack. Therefore, many sites deploy intrusion detection systems (IDSs) or intrusion prevention systems (IPS) to detect and block the attacks. Moreover, to oppose the attacks that use forged source addresses, the IP traceback technologies are researched and developed [1] [2] [3].
A new attack method appears one after another. Existing firewalls, IDSs, and IPSs maintain information on past attacks in the database called "attack signatures." However, unknown attacks cannot be detected because there is no information in attack signatures for them.
In this paper, we propose a new attack detection method which can detect unknown attacks and a method to predict damage from detected attacks, and evaluate them.

2. EXISTING METHODS OF INTRUSION DETECTION

Currently, there are two intrusion detection methods that are called "misuse detection" and "anomaly detection" [4]. In the misuse detection method, the intrusion detection system maintains attack signatures, and compares packet data with them. If the packet data is matched with one of the signatures, it is detected as an attack. On the other hand, in the anomaly detection method, the intrusion detection system maintains usual access patterns which are learned from actual accesses of the site, and compares packet data with them [5]. If the packet data is greatly different from usual access patterns, it is detected as an attack.
In the misuse detection method, there is a problem that unknown attacks which are not registered in attack signatures cannot be detected. Although anomaly detection method can detect attacks including unknown ones, there is a problem that the usual access pattern can be operated by the attacker so that the attacker's access is not judged as an attack by the intrusion detection system because the usual access pattern is acquired from the data of actual accesses.

3. PROPOSED ARCHITECTURE

Detecting Network Attacks
To detect network attacks, we defined the conditions of the normal accesses. At first, we defined the conditions of the accesses that match the specifications of the protocols used. However, there are some protocol functions that are not currently used such as IPv4 options and fragmentations. Therefore, we defined the additional conditions that prohibit using these functions. Second, we defined the conditions of the accesses that match the site access policy. Finally we defined the "normal accesses" as the accesses that match all the conditions defined above (Figure 1).

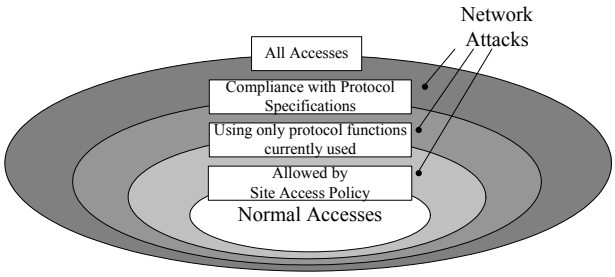

Figure 1. Definition of Normal Accesses

In our method, an intrusion detection system captures packets on the network destined for the internet servers such as web servers, mail servers, FTP servers, and DNS servers, and compares them with the conditions of the normal accesses. These conditions are like follows:

- Must be based on the specification of IP.
- Must be based on the specification of a transport protocol (e.g. TCP or UDP) used.
- Must be based on the specification of an application protocol (e.g. HTTP, SMTP, FTP, or DOMAIN) used.
- The protocol type and port number used must be permitted by the site access policy.
- The protocol parameters (commands / values) used in the application protocol must be permitted by the site access policy.

We will show the examples of IP check conditions (Table1), TCP check conditions (Table2), and HTTP check conditions (Table3).
If the captured packet is conformed to all conditions, it is considered as a "normal packet". On the other hand, if the packet is not conformed to at least one of the above conditions, it is considered as an "attack packet". Above conditions can be defined based on existing protocol specifications published as Request for Comments (RFCs) and the site access policy. Protocol specifications and site access policy define a range of normal access, not a range of attack, and it detects the packet that is "not a normal". Thus many attacks including unknown ones can be detected by this method.

Table 1. Example of IP Check Conditions

Fields	Conditions
Version	Must be equal to 4.
Internet Header Length (IHL)	Must be equal to 5. (no IPv4 options)
Total Length	Must be grater than 20.
Flags (Reserved)	Must be equal to 0.
Flags (MF) Fragment Offset	Must be equal to 0. (no fragmentation)
TTL	Must be grater than 3.
Header Checksum	Must be correct.
Source Address	Must be global IP address.

Table 2. Example of TCP Check Conditions

Fields	Conditions
Source Port Destination Port	Either of them must be 1024 or more.
Data Offset	Must be grater than or equal to 5.
Reserved	Must be equal to 0.
Control Bits (URG, ACK, PSH, RST, SYN, FIN)	Must be SYN, RST, ACK, FIN+ACK, SYN+ACK, RST+ACK, ACK+PSH, or FIN+ACK+PSH.
Checksum	Must be correct.

Table 3. Example of HTTP Check Conditions

Fields	Conditions
Method	Must be allowed by site access policy.
Request-URI	Must consist of characters usually used (0-9, a-z, A-Z, tilde, hyphen, underscore, period, "+", "$", "#", "%7E", and "%7e").
	"?" must not be included when not using CGI.
	The length of URL must be within the limits of the length defined by the site access policy.
HTTP-Version	Must be "HTTP/1.0" or "HTTP/1.1" (if any).
HTTP headers	Must be specified in the specifications.
	HTTP header length must be less than the value specified in site access policy.

Predicting Damage from Network Attacks
The function for predicting damage from network attacks is required because IDSs which don't use attack signatures cannot distinguish a type of damage that a host will suffer when it accepts the detected packet, thus administrators cannot know what is going on. To predict damage from detected attacks, we use neural networks. By making neural networks learn the relationship between the features of attack packets and the damages they cause, the neural networks can predict damages from detected attack packets. By using neural network, even when a non-learned packet data is inputted to it, it can predict damage somewhat correctly based on the past learned data. And, even when the wrong damage has been predicted, it is possible to improve its ability easily by carrying out re-study using the data. Furthermore, since it is impossible to analyze a calculation algorithm of learned neural network from the outside, it is difficult for an attacker to make the neural network be confused and predict slight damage.
When an attack is detected by our detection method, features of the captured packet are inputted to the learned neural network. The neural network outputs the predicted damage as "no damage", "information retrieval", "performance down", "system down", or "unauthorized manipulation" suffered from the detected attack. If unknown attack is occurred, it can predict damage from the knowledge from past attacks. Figure 2 shows an example of the neural network for predicting damage.

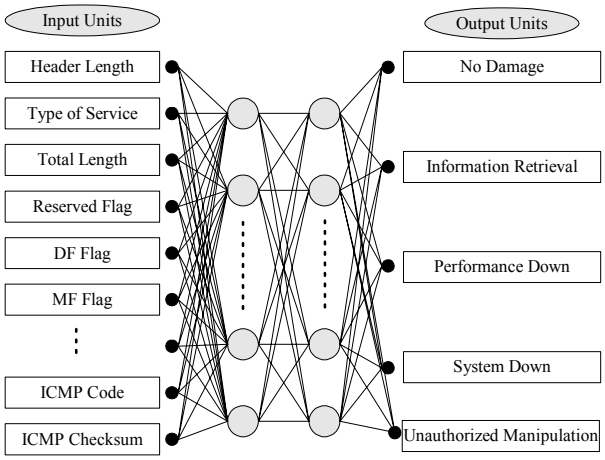

Figure 2. Example of Neural Network for Predicting Damage

Process Flow

Figure 3 shows the process flow in our system. The intrusion detection system monitors networks and captures packets destined for internet servers. And it checks the IP header, TCP/UDP/ICMP header, and application protocol layer contents respectively. At each check, the system checks the fields of the captured packet compared with the conditions from protocol specification (called "protocol specification check") and site access policy (called "access policy check"). Furthermore, the system checks the number of packets compared with site access policy to detect flooding attacks.

If the packet is not conformed to at least one of the conditions, the results of checks are sent to the damage prediction section, and the system predicts damage from the results of checks. Finally, an alarm with predicted damage is sent to the administrator by E-mail etc.

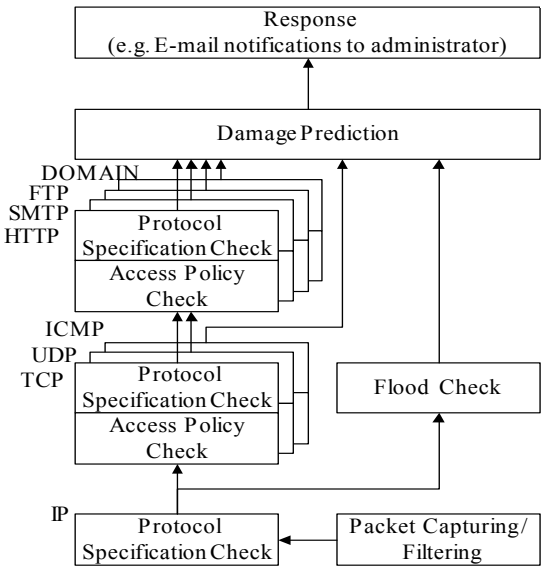

Figure 3. Process Flow

4. IMPLEMENTATION AND EVALUATION

Implementation

We developed a prototype system implementing our proposed methods based on the process flow in Figure 3. Table 4 shows the specifications of the prototype system. We implemented IP, UDP, TCP, ICMP, HTTP, SMTP, FTP, and DOMAIN protocol check on the prototype system.

Table 4. Prototype Specifications

OS	FreeBSD 4.2-RELEASE
CPU	2GHz Pentium 4
Memory	512MB RAM
Network Interface Card	Intel Pro/100+ (100Mbps)
Capture Program	BPF + libpcap

Evaluation of Accuracy in Intrusion Detection

We evaluated an accuracy of detection using developed prototype system. To evaluate accuracy of detection, we measured a true positive rate (detection rate or sensitivity) and false positive rate (false alarm rate).

To measure the true positive rate, we selected 221 attacks from the Whitehats, Inc.'s arachNIDS attack database [6] according to the following conditions:

- Attacks using IP, UDP, TCP, ICMP, HTTP, SMTP, FTP, or DOMAIN protocols, for which checks are implemented on the prototype system
- Attacks that target server programs, not client programs

Figure 4 shows the experimental network to measure the true positive rate. From the attacker's host, we generated the selected attacks which targeted the web server, mail server, FTP server and DNS server, and let the prototype system monitor the network.

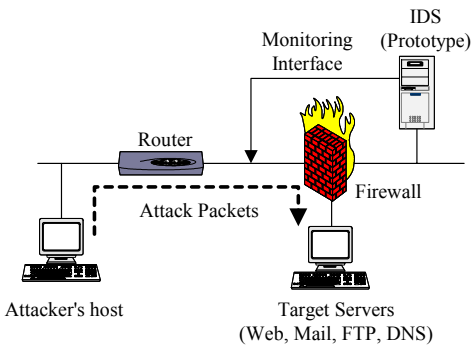

Figure 4. Experimental Network (True Positive Rate)

To measure the false positive rate, we deployed our prototype system in a real environment and let it monitor the accesses from the Internet to the web server, mail server, FTP server and DNS server.

Figure 5 shows the network to measure the false positive

rate. We deployed the prototype systems and the hosts which dump the packets using tcpdump program, and let them monitor the accesses to the servers.

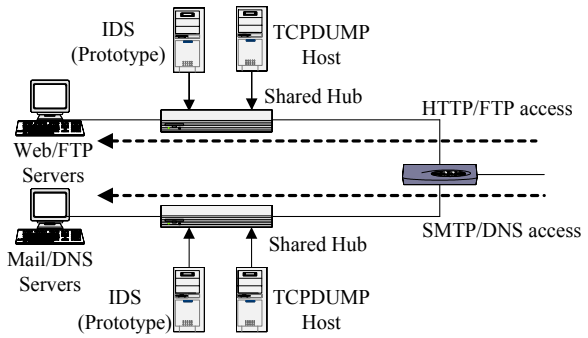

Figure 5. Network (False Positive Rate)

Table 5-7 shows the protocols, DOMAIN parameters, and protocol commands that were permitted by the access policy of the intrusion detection system. URL length, HTTP header length, SMTP command line length, and ICMP message length were permitted below 200 bytes, respectively. FTP command line length was permitted below 100 bytes. FTP users who have the permission to access were set to "anonymous" and "ftp." And at the measurement of the true positive rate, the access policy was set up so that use of CGI in web access is forbidden, and at the measurement of the false positive rate, it was set up so that use of CGI is permitted.

Table 5. Protocols Permitted by Access Policy

TCP	HTTP (Port 80), SMTP (Port 25), FTP (Port 21), FTP-DATA (Port 20, 1024-65535)
UDP	DOMAIN (Port 53)
ICMP	Destination Unreachable (Type 3) Time Exceeded (Type 11) Parameter Problem (Type 12)

Table 6. DOMAIN Parameters Permitted by Access Policy

Parameters	Permitted Values
OPCODE	QUERY
QTYPE	A, NS, CNAME, PTR, MX, ANY

Table 7. Protocol Commands Permitted by Access Policy

Protocols	Permitted Commands
HTTP	GET, HEAD
SMTP	HELO, EHLO, MAIL, RCPT, DATA, QUIT
FTP	USER, PASS, CWD, CDUP, PORT, PASV, TYPE, MODE, RETR, ABOR, PWD, XPWD, LIST, NLST, SYST, SIZE, NOOP, QUIT

Table 8 shows the results of true positive rate. As a result, over 89% of 221 attacks were detected. Especially, attacks using IP, TCP, UDP, ICMP, or DOMAIN were all detected.

Table 9 shows the results of false positive rate. Because detected alarms include both real alarms caused by attacks and false alarms, we differentiated them by comparing the data captured by tcpdump with existing attack data. Then, we got false alarms extracting the alarms which were caused from attack packets from all alarms, and got false positive rate dividing the number of false alarms by the number of normal packets received. Although the false positive rate for DOMAIN (UDP) is 5.1%, others are only 0-2%.

Table 8. Results (True Positive Rate)

Protocols	Number of Detected Attacks	Number of Undetected Attacks	True Positive Rate
HTTP	71	12	85.5%
SMTP	12	3	80.0%
FTP	31	8	79.5%
DOMAIN	8	0	100%
ICMP	41	0	100%
TCP	20	0	100%
UDP	1	0	100%
IP	14	0	100%
Total	198	23	89.6%

Table 9. Results (False Positive Rate)

Protocols	Number of Packets	Number of Alarms		False Positive Rate *1
		Real Alarms	False Alarms	
HTTP	187,092	531	2,190	1.2%
SMTP	13,968	3	281	2.0%
FTP	537	34	0	0%
DOMAIN (TCP)	85	1	0	0%
DOMAIN (UDP)	1,987	6	101	5.1%
ICMP	11,860	158	0	0%
Total	215,529	733	2,572	1.2%

*1 False Positive Rate =
False Alarms / (All Packets - Real Alarms)

Evaluation of Accuracy in Damage Prediction

We also evaluated an accuracy of damage prediction. To measure the damage prediction accuracy rate, we selected attack data which have strange features in IP, ICMP, TCP, or UDP headers from the CVE (Common Vulnerabilities and Exposures) [7] attack database. Then, we got attack programs which generate the selected attacks referring the CVE references, and got the actual attack packet data. Then, we built the following neural networks.

- IP-ICMP neural network, which predicts damage from attacks which have unusual feature in IP or ICMP
- IP-TCP neural network, which predicts damage from attacks which have unusual feature in IP or TCP
- IP-UDP neural network, which predicts damage

from attacks which have unusual feature in IP or UDP

We bisected the attack packet data for evaluation and for learning. Table 10 shows the number of data for evaluation and the number of data for learning. Table 11 shows some input units of the neural networks. In table 11, "raw" means that the normalized raw data of the header field is inputted to the neural network.

Table 10. Number of Data and Input Units

	Number of Data for Evaluation	Number of Data for Learning
IP-ICMP	448	459
IP-TCP	161	171
IP-UDP	263	272

Table 11. Input Units of Neural Network

Protocols	Input Units (extracts)	
IP	Header Length (0-4) Header Length (5) Header Length (6-15) TOS (0) TOS (recommended value) TOS (other) Total Length (raw) Total Length (< Header Length) Reserved Flag (raw) DF Flag (raw) MF Flag (raw)	Fragment Offset (0) Fragment Offset (>= 65536) Time to Live (raw) Checksum (correct) Src Address (private) Src Address (other bogus) Src Address (= Dst Address) Src Address (broadcast) Dst Address (broadcast)
ICMP	Type (8,13,15,17) Type (0,14,16,18) Type (3,11) Type (4,5,12)	Type (other) Code (allocated value) Checksum (correct)
TCP	Src Port (>= 1024) Src Port (< 1024) Src Port (= Dst Port) Dst Port (FTP) Dst Port (DOMAIN) Dst Port (HTTP) Dst Port (SMTP) Dst Port (Telnet) Dst Port (>= 1024) Dst Port (< 1024) Ack Number (!=0) Data Offset (raw) Reserved (raw) Flag-URG (raw)	Flag-ACK (raw) Flag-PSH (raw) Flag-RST (raw) Flag-SYN (raw) Flag-FIN (raw) Flag (normal combination) Flag (illegal combination) Window (raw) Checksum (correct) Urgent Pointer (!=0) Options (exists)
UDP	Src Port (>= 1024) Src Port (< 1024) Src Port (= Dst Port) Dst Port (FTP) Dst Port (DOMAIN) Dst Port (HTTP) Dst Port (SMTP)	Dst Port (Telnet) Dst Port (>= 1024) Dst Port (< 1024) Length (>= IP Header Length + 8) Checksum (correct)

We measured the damage prediction accuracy rate for each neural network changing the parameters of it. Table 12 shows the number of hidden units, the number of times of learning, and the damage prediction accuracy rate of each neural network whose prediction accuracy was the best. As a result, over 98% of the predicted damages were correct.

Table 12. Prediction Accuracy Rate

	Number of 1st Hidden Units	Number of 2nd Hidden Units	Number of Times of Learning	Prediction Accuracy Rate
IP-ICMP	5	0	2318	99.78%
IP-TCP	10	0	4866	98.20%
IP-UDP	5	0	4377	99.48%

5. CONSIDERATIONS

In this section, we will consider the effectiveness of this system from the results of detection accuracy (true positive rate and false positive rate) and damage prediction accuracy.

Detection Accuracy

From the results of the evaluation, it is shown that this detection system has a high detection rate. Although the detected attacks are known for network security experts, the information for these attacks is not registered into our system. That is, for our system, they are unknown attacks and it is expected that other unknown attacks are detectable by our system.

Especially, attacks that abused IP, TCP, UDP, ICMP, and DOMAIN are all detected. From this, our detection method is considered to be effective to attacks which abused the protocols which defined the format strictly. Attacks which issue inaccurate commands or cause buffer overflows were detected. However, the following attacks were not detected.

(1) Intrusion to the FTP server gaining root privilege by using "CWD ~root" command
(2) Investigations using scanner programs
(3) Accesses to secret files, such as password files using HTTP or FTP
(4) Attacks which abused external programs, such as CGI and mailing list server programs

As for (1), it becomes possible to detect by extending the protocol specification check so that the order of issued commands can be compared with a normal order (for example, CWD command must be issued after the USER and PASS commands). As for (2), it is considered to become detectable if the system checks the password used when a scanner logs into a target system. However, it is not a big problem since they are only the accesses for investigations and do not cause actual damage. As for (3), it is considered to become detectable if the IDS checks the name of a file which an attacker accesses. However, registering the all the file names that can be accessed into the detection system have a problem in an

aspect of practical use. As for (4), since the format of the data passed to the external program which should be checked is not specified in the specification of the external program, not a protocol specification, it is thought that there was a portion which has not been checked only with the protocol specification check or access policy check. In order to detect these attacks, it is necessary to take in the specification of an external program to IDS and to confirm whether it is based on the specification of an external program.

Moreover, it was shown that our system has a low false positive rate by the evaluation. However, false alarms were generated since the access policy was not set up suitably. Some false alarms were generated when non-standard headers or new headers of HTTP or SMTP which are not registered into the prototype were used. These false alarms were generated since client programs used the headers specified in the draft specifications, or cashing servers or proxy servers added non-standard headers which are not specified in the specifications. The prompt action to new specifications and the correspondence to the non-standard headers currently used are required. Moreover, although only the access from port 53 to port 53 were permitted by the access policy as the normal DNS access, Microsoft Windows sometimes use the port 137 as the source port of the DNS access, and this caused false alarms. Although an access policy should be set up after investigating the actual use enough in advance, it may not be suitable for actual use. For this reason, experimental employment of this system is necessary. If it generates many alarms during the experimental employment, after checking the packet data which cause alarms enough, the access policy should be correct suitably. It is thought that this becomes possible to reduce false alarms.

Damage Prediction Accuracy

The data used for evaluation are non-learned data for the neural networks. In spite of it, we got high prediction accuracy rate at the evaluation. This is the result of demonstrating an advantage of using neural networks, and it was shown that the proposed method is effective also for unknown attacks.

Attacks which the neural networks mistook prediction of damage were port scans and accesses which cause performance down. These cannot be predicted collectively if the neural networks do not analyze two or more packets simultaneously. On the other hand, since our system predicts damage from a single packet, it is not able to predict damage of these network attacks correctly. To increase the prediction accuracy rate, it is necessary to make input data and learning algorithm of the neural networks deal with the sequence of packets.

6. CONCLUSIONS

We proposed a method for detecting network attacks including unknown ones using protocol specifications and site access policy. Furthermore, we proposed a method to predict damage from detected attack using neural networks.

We developed a prototype system implemented our proposed method and showed effectiveness using prototype system. In misuse detection method, although it is effective for detecting known attacks which are registered to IDS as attack signatures, it can not detect unknown attacks. On the other hand, in our proposed method, it can detect attacks with high detection rate without attack signatures. Therefore, it is effective for detecting unknown attacks.

7. ACKNOWLEDGEMENTS

This work was funded by the Telecommunications Advancement Organization of Japan (TAO, reorganized as NICT: the National Institute of Information and Communications Technology in April, 2004).

8. REFERENCES

[1] T. Baba and S. Matsuda, "Tracing Network Attacks to Their Sources," IEEE Internet Computing, March/April 2002, pp.20-26.

[2] S. Savage et al., "Practical Network Support for IP Traceback," Proc. 2000 ACM SIGCOMM, vol. 30, no. 4, ACM Press, New York, Aug. 2000, pp. 295-306.

[3] S. Bellovin, M. Leech, and T. Taylor, "ICMP Traceback Messages," draft-ietf-itrace-04.txt, Internet Draft, Work in Progress, Feb. 2003.

[4] B. Mukherjee, L.T. Heberlein, and K.N. Levitt, "Network Intrusion Detection," IEEE Network, pp. 26-41, May/June 1994.

[5] P.A. Porras and P.G. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," In Proceedings of the 20th National Information Systems Security Conference, pp. 353-365, October 1997.

[6] Whitehats, Inc., arachNIDS (advanced reference archive of current heuristics for network intrusion detection systems), http://www.whitehats.com/ids/

[7] The MITRE Corporation, Common Vulnerabilities and Exposures (CVE), http://cve.mitre.org/