PGNIDS(Pattern-Graph Based Network Intrusion Detection System) Design*

Byung-kwan Lee, Seung-hae Yang, Dong-Hyuck Kwon, and Dai-Youn Kim

Dept of Computer Engineering, Kwandong University, Korea
bklee@kd.ac.kr, yang7177@cho.com, taz0108@hotmail.com,
dy2300@freechal.com

Abstract. PGNIDS(Pattern-Graph based Network Intrusion Detection System) generates the audit data that can estimate intrusion with the packets collected from network. An existing IDS(Intrusion Detection System), when it estimates an intrusion by reading all the incoming packets in network, takes more time than the proposed PGNIDS does. As this proposed PGNIDS not only classifies the audit data into alert and log through ADGM(Audit Data Generation Module) and stores them in the database, but also estimates the intrusion by using pattern graph that classifies IDPM(Intrusion Detection Pattern Module) and event type, Therefore, it takes less time to collect packets and analyze them than the existing IDS, and reacts about abnormal intrusion real time. In addition, it is possible for this to detect the devious intrusion detection by generating pattern graph.

1 Introduction

PGNIDS is proposed to design network intrusion detection system based on pattern graph. When information is exchanged, PGNIDS can detect illegal connectivity and intrusion-related behavior such as DoS(Denial of Service) attack and port scan.

2 Related Works[12]

2.1 IDS Analysis

There are two primary approaches to analyzing events to detect attacks: misuse detection and anomaly detection. Misuse detection, in which the analysis targets something known to be "bad", is the technique used by most commercial systems. Anomaly detection, in which the analysis looks for abnormal patterns of activity, has been, and continues to be, the subject of a great deal of research. Anomaly detection is used in limited form by a number of IDSs.

* This work was supported by grant No. B1220-0501-0315 from the University fundamental Research Program of the Ministry of Information & Communication in Republic of Korea.

M. Gavrilova et al. (Eds.): ICCSA 2006, LNCS 3982, pp. 38 – 47, 2006.

2.2 The Kind of Intrusion Detection

Some IDSs analyze network packets captured from network backbones or LAN segments, to find attackers. Other IDSs analyze sources information generated by the operating system of application software for signs of intrusion.

2.2.1 Network-Based IDSs

The majority of commercial intrusion detection systems are network-based. These IDSs detect attacks by capturing and analyzing network packets. Listening on a network segment or switch. the network-based IDS can monitor the network traffic affecting multiple hosts that are connected to the network segment, thereby protecting those hosts. Network-based IDSs often consist of a set of single-purpose sensors or hosts placed at various points in a network. These units monitor network traffic, performing local analysis of that traffic and reporting attacks to a central management console. As the sensors are limited to running the IDS, they can be more easily secured against attack. Many of these sensors are designed to run in "stealth" mode, in order to make it more difficult for an attacker to determine their presence and location.

2.2.2 Host-Based IDSs

Host-based IDSs operate on information collected from within an individual computer system. This vantage point allows host-based IDSs to analyze activities with great reliability and precision, determining exactly which processes and users are involved in a particular attack on the operating system. Furthermore, unlike network-based IDSs, host-based IDSs can "see" the outcome of an attempted attack, as they can directly access and monitor the data files and system processes usually targeted by attacks.

2.2.3 Application-Based IDSs

Application-based IDSs are a special subset of host-based IDSs that analyze the events transpiring within a software application. The most common information sources used by application-based IDSs are the application's transaction log files. The ability to interface with the application directly, with significant domain or application-specific knowledge included in the analysis engine, allows application-based IDSs to detect suspicious behavior due to authorized users exceeding their authorization. This is because such problems are more likely to appear in the interaction between the user, the data, and the application.

3 PGNIDS Design

As shown Fig.1 PGNIDS consists of DCM(Data Collection Module), ADGM(Audit Data Generation Module), IDPGM(Intrusion Detection Pattern Generation Module), and PGGM(Pattern Graph Generation Module). DCM collects all the incoming packets in Network. ADGM generates the audit data that decides an intrusion and classifies them according to behavior characteristics. IDPGM generates patterns with the classified audit data. PGGM generates pattern graphs with the patterns that IDPGM generates and decides whether it is an intrusion with them.

3.1 Network-Based DCM

PGNIDS in this paper uses libpcap which is called packet capture library provided in LINUX. libpcap captures and filters packets. The necessary contents from the filtered packets are extracted according to packet filtering rule in order to generate only audit data that PGNIDS requires.

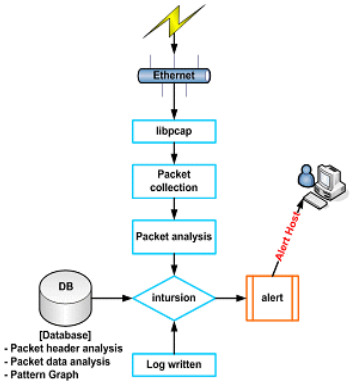

Fig. 1. PGNIDS design

```
struct pcap {
  int snapshot;
  int linktype;
  int tzoff;
  int offset;
  struct pcap_sf sf;
  struct pcap_md md;
   /* Read buffer. */
  int bufsize;
  u_char *buffer;
  u_char *bp;
  int cc;
   /* Place holder for
pcap_next().
  */
  u_char *pkt;
   /* Placeholder for filter
code if bpf not in kernel. */
  struct bpf_program fcode;
  char
errbuf[PCAP_ERRBUF_SIZE];
  }
 typedef struct pcap pcap_t;
```

Fig. 2. The algorithm for capturing packets using libpcap

3.1.1 Packet Capture

Packet capture confirms the contents of all the incoming packets in network. This can apply to various types such as monitoring, network debugging and sniffering for statistics and security for network use. The method capturing packets by using libpcap is as follows. First, the device(NIC) or the file which will be captured is opened, the packets are analyzed, and the device or the file is closed. Libpcap provides various interfaces according to its function. Fig .2 shows the algorithm for capturing packets using libpcap.

3.1.2 Packet Filtering Rule

The rule of packet filter uses source address, source port number, destination address, destination port number, protocol flag, and activity(pass/reject). With these fields, sequential ACL(Access Control List) for filtering packets has to be written. Screening router is the software that decides activity, that is, pass or reject in ACL sequentially. The filtering rule consists of one or several primitives and its format is shown in table. 1.

Table 1. The format of packet filter rule

action [direction] [log] [quick] [on interface] [af] [proto protocol] \ [from src_addr [port src_port]] [to dst_addr [prot dst_port]] \ [flag tcp_flags] [state]

- Action. The action to be taken for matching packets, either pass or block. The default reaction may be overridden by specifying either block drop or block return.
- Direction. The direction the packet is moving on an interface, either in or out.
- Log. Specifies that the packet should be logged via pflogd. If the rule specifies the keep state, modulate state, or syn proxy state option, then only the packet which establishes the state is logged. To log all packets regardless, use log-all.
- Quick. If a packet matches a rule specifying quick, then that rule is considered the last matching rule and the specified action is taken.
- Interface. The name or group of the network interface that the packet is moving through. An interface group is specified as the name of the interface but without the integer appended.
- Af. The address family of the packet, either inet for IPv4 or inet for IPv6.
- Protocol. The layer 4 protocol of the packet. : tcp, udp, icmp, icmp6, a valid protocol name from /etc/protocols, a protocol number between 0 and 255, a set of protocols using a list.
- src_addr, dst_addr. The source/destination address in the IP header.
- src_port, dst_port. The source/destination port in the layer 4 packet header.
- tcp_flags. Specifies the flags that must be set in the TCP header when using proto tcp. Flags are specified as flags check/mask.
- State. Specifies whether state information is kept on packets matching this rule.

3.1.3 Packet Filtering Flow Chart

Packet filtering rules are stored in a particular order and is applied to the packets in that order. Fig.3 shows the flow of packet filtering.

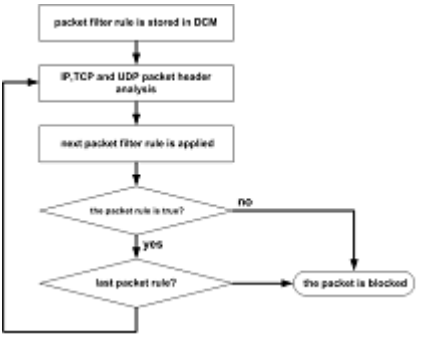

Fig. 3. Packet filter operational order

3.2 ADGM(Audit Data Generation Module)

As it is not proper to use the collected data for audit data, ADGM(Audit Data Generation Module) is used to extract only audit data that can decide an intrusion from the collected packets. First, ethernet frame in packet filtering must be divided into IP or ARP packet separately in ethernet layer. In IP layer, IP packet is divided into ICMP, TCP, or UDP separately. That is, in this paper ADGM classifies IP packet into TCP, UDP and, ICMP, generates the audit data and stores them in the database to detect an intrusion. Fig .4 passes the packets of unsigned char type to the pointer of ethernet header. If the packet is IP protocol, Fig .4 shows that it is transferred to the pointer of IP header and is classified into TCP, UDP, and ICMP.

```
void packet_analysis(unsigned char *user, const struct pacap_pkthdr *h,
 const unsinged char *p)
 {
unsigned int length = h->len;
 struct ether_header *ep;
 unsinged short ether_type;
 length -= sizeof(struct ether_header);
  ep = (struct ether_header *)p;
  p = += sizeof(struct ether_header)
                …
 if (ip->protocol == IPPROTO_TCP) {//TCP protocol
  tcph = (struct tcphdr *)(P + (iph->ihl*4) + (tcph->doff * 4));
  tcpdata = (unsigned char *) (p + (iph->ihl * 4) + (tcph->doff * 4));
 }
                …
  if (ip->protocol == IPPROTO_UDP) {//UDP protocol
   udph = (struct udphdr *) (p + iph->ihl * );
   udpdata = (unsigned char *) (p_iph->ihl * 4) + 8;
  }
                …
  if (ip->protocol == IPPROTO_ICMP) {//ICMP protocol
   icmp = (struct icmp *) ([+iph->ihl * 4);
   icmpdata = (unsigned char *) (p + iph->ihl*4) + 8;
                …
  }
```

Fig. 4. Packet Analysis Algorithm

3.3 IDPGM(Intrusion Detection Pattern Generation Module)

The Intrusion detection pattern proposed in this paper is based on that of Snort.

3.3.1 Snort

Snort is a lightweight network IDS that real time traffic analysis and packet logging can be done on the IP network. In addition, it is network sniffer based on libpcap.

That is, it is a tool which monitors, writes, and alarms network traffic that matches intrusion detection rule.

Snort can do protocol analysis, contents detection and the matching and detect the various attacks and the scans such as overflow, stealth port scan, CGI attack, SMB detection.

3.3.2 IDPGM

Intrusion detection pattern is based on that of Snort and Snort consists of the packets of TCP, UDP and ICMP, etc. These packets generates pattern format with backdoor.rules, ddos.rules, dns.rules, dos.rules, exploit.rules, finger.rules, ftp.rules, icmp.rules, info.rules, misc.rules, netbios.rules, policy.rules, rpc.rules, rservices.rules, scan.rules, smtp.rules, sql.rules, telnet.rules, virus.rules, and web-cgi.rules etc. These pattern format is shown in table. 2.

Table 2. Pattern format

```
alert tcp $EXTERNAL_NET any ->
$HOME_NET 80 (content:"|90C8 C0FF FFFF|/bin/sh" msg:
"IMAP buffer overflow!";)
```

The generated pattern proposed in this paper is stored in a database mysql in order to be used for intrusion detection. Fig .5 shows the structure of TCP data.

```
{
unsigned short s_port;     // source port number
unsigned short d_port;     // destination port number
 int flat;                 // TCP flag
char content[200];         // data
int c_size;                // content length of content
int p_size;                // size of payload
char msg[200];             // message
}
```

Fig. 5. Structure of TCP data

```
struct pattern_graph
{
char p_name[50]          // name of pattern graph
unsigned short n_id;     // node number
unsigned short s_port;   // source port number
char content[200];       // data
char msg[200];           // message
}
```

Fig. 6. Structure of Pattern Graph

3.4 PGGM(Pattern Graph Generation Module)

The patterns generated by IDPGM are stored in a database. PGGM generates pattern graphs by analyzing the relationship between patterns, stores them in the database, and prevents a devious intrusion by using the generated pattern graph.

3.4.1 Pattern Detection Algorithm

The Proposed PGNIDS changes AS(Attack Specification Language) to the pattern that is the data structure suitable for PGNIDS to process attacks. The pattern graph is to draw the process of Scenario in tree type and the last node of the tree has no transmission event. Each node of pattern graph means a message. Table. 3 shows the type of message.

Table 3. Message Format

type	<node ID, timestamp, attribute price>
node ID	Each node number in pattern graph
timestamp	Event occurrence hour
attribute price	Attribute price of event

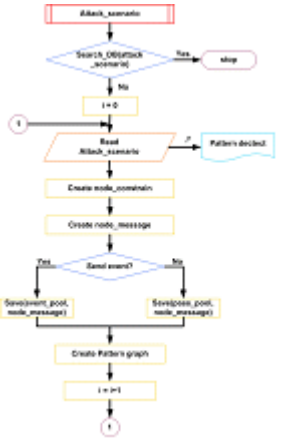

Fig. 7. Pattern graph generation flow chart

```
    pattern_detection(attack_scenario) {
    if
(search_DB(attack_scenario)==FALSE) {
    i=0;
    while(!EOF) {
    read(attack_scenario);
    create_constrain(node, i);
    message = create_message(node, i);
    if event(send) {
    save(event_pool, message); }
    else {
    save(pass_pool, message); }
create_graph
   (pattern, event_pool, pass_pool, i);
    i = i+1; }
    return(pattern); }
    }
```

Fig. 8. The algorithm detecting pattern

The message of each node, when the event of the node is transmitted, is stored in the event pool and the rest of events are stored in the pass pool. In the course of changing to pattern graph, some limitations about each node happens. At this time the conditions of the limitations consists of the static limitation condition that has a constant value and the dynamic limitation condition that has a variable value. Fig. 7 shows the flow chart of pattern graph. Fig. 8 shows the algorithm detecting pattern.

When pattern detection algoritm is applied by using the attack scenario of Fig. 9, the generated pattern graph is shown in Fig. 10.

```
ATTACK "sample" [a, b, c]
a {
send(c) : e1[$x=a0;] }
b {
e2[$y = a1;]
send(c) : e3[a2 == $x;] }
c {
e4[]
e5[a3 == $y;]}
```

Fig. 9. Attack scenario

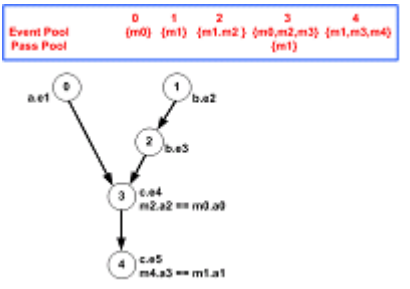

Fig. 10. Pattern graph

4 PGNIDS Simulation

The objects of PGINDS proposed in this paper are all the packets on the network and it is possible for PGNIDS to capture packets under TCP, UDP, and ICMP environment by using packet filter. As PGNIDS performs real-time network, it reduces the collection of packets and analysis time and reacts to abnormal attack real-time.

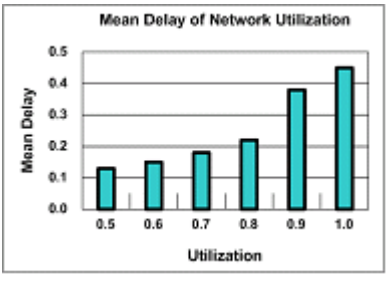

Fig. 11. Network utilization analysis

Table 4. The number of event according data type

Event / Type	24 hours
alert	32,448
Log	33,127

4.1 The Analysis of Network Utilization

Fig. 11, when PGNIDS reports the detection information to a server, shows the mean delay time of each event. That is, Fig. 11 shows the delay of transmission packet according to the change of network utilization and that its delay is increasing rapidly over utilization 0.8.

4.2 The Analysis According to Event Type

Table. 4 shows the analysis according to event type processed by packet filtering rule for 24 hours in PGNIDS. Fig. 12 shows the delay of alert and log event. The delay about each event shows the increasing trend according to the network utilization.

4.3 The Analysis Using Pattern Graph

PGNIDS generates pattern graph by using each event, stores it in a database, and detects a devious attack by making pattern graph with the relationship between each event. Fig. 13 shows the detection ration of devious attack according to network delay ratio by using pattern graph between each event.

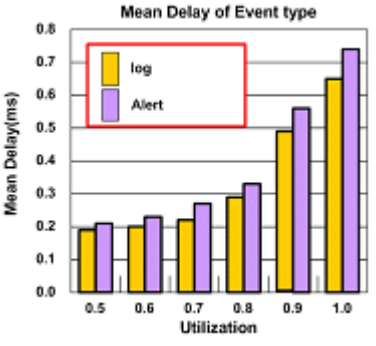

Fig. 12. The analysis according to event type

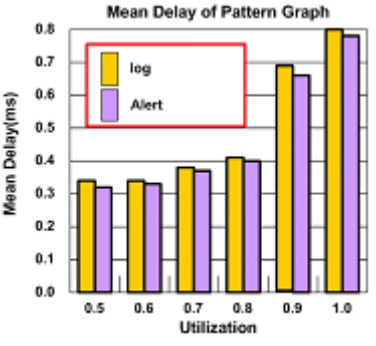

Fig. 13. The analysis with a pattern graph

5 Conclusion

PGNIDS is the system that decides an intrusion by collecting network packets real time. An existing IDS, when it estimates an intrusion by reading all the incoming packets in network, takes more time than the proposed PGNIDS does. As this proposed PGNIDS not only classifies the audit data into alert and log through ADGM and stores them in the database, but also estimates the intrusion by using pattern graph that classifies IDPM and event type, Therefore, it takes less time to collect packets and analyze them than the existing IDS, and reacts to abnormal intrusion real time. In addition, it is possible for this to detect the devious intrusion detection by generating pattern graph.

References

1. Byung-Kwan Lee, Eun-Hee Jeong, “Internet security”, Namdoo Books, 2005
2. LBNL's Network Research Group
3. http://www.linux.co.kr/
4. Kwang-Min Noh, It uses pacp library from linux and packets it catches and it sees v0.3, 2000.09.14, Linux Korean alphabet document project
5. http://www.snort.org
6. http://www.silicondefense.com/snortsnarf
7. http://my.dreamwiz.com/winmil/security/snort.htm
8. http://www.whitehats.com/

9. Tsutomu Tone, "1% network principal which decides a success and the failure“, Sungandang, 2004
10. http://www.windowsecurity.com
11. Dai-il Yang, Seng-Jea Lee “Information security surveying and actual training”, Hanbit Media, 2003
12. Rebecca Bace, Peter Mell, NIST Special Publication on Intrusion Detection Systems
13. http://www.openbsd.org/faq/pf/filter.html