# A Flow-based Method for Abnormal Network Traffic Detection

*Myung-Sup Kim, Hun-Jeong Kang, Seong-Cheol Hong, Seung-Hwa Chung, and James W. Hong*
*Dept. of Computer Science and Engineering*
*POSTECH*
*{mount, bluewind, pluto80, mannam, jwkhong}@postech.ac.kr*

**Abstract**

One recent trend in network security attacks is an increasing number of indirect attacks which influence network traffic negatively, instead of directly entering a system and damaging it. In future, damages from this type of attack are expected to become more serious. In addition, the bandwidth consumption by these attacks influences the entire network performance. This paper presents an abnormal network traffic detecting method and a system prototype. By aggregating packets that belong to the identical flow, we can reduce processing overhead in the system. We suggest a detecting algorithm using changes in traffic patterns that appear during attacks. This algorithm can detect even mutant attacks that use a new port number or changed payload, while signature-based systems are not capable of detecting these types of attacks. Furthermore, the proposed algorithm can identify attacks that cannot be detected by examining only single packet information.

**Keywords**

Network Security Attack, Abnormal Network Traffic Detection, Traffic Monitoring and Analysis.

## 1. Introduction

Today, the number of Internet users is continuously increasing, along with new network services. As the internet grows, network security attack threats have become more serious. Many security vulnerabilities are exposed and exploited by attacks. Recent reports on Internet security breaches indicate that the frequency and the damage costs are continuously rising.

One recent network attack trend is the use of network traffic. An attacker places networks or hosts in jeopardy, without intruding into the hosts. The attacks on a famous website, such as Yahoo, E-bay, and E*trade, are good examples [1, 2]. This type of Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks [3, 4, 5] will cause more damage for the following reasons. There are so many DoS/DDoS tools that even unskilled users can use easily. A successful DoS/DDoS attack shows its impact quickly and makes it difficult to trace back to

the intruder. Moreover, the bandwidth consumption by the attacks influences network performance. Even on highly over-provisioned links, malicious traffic causes an increase in the average DNS latency by 230% and an increase in the average web latency by 30% [6]. From the monitoring result of NG-MON [7], we can observe a more serious latency deficiency (up to 500%) in enterprise networks that contains a target or bypassing machine of the attacks. These menaces require us to make provisions against DoS/DDoS attacks.

This paper focuses on detecting abnormal network traffic which includes those generated by internet worms[6], DoS/DDoS and scanning[24], scanning which include both port and network scanning. We present a detecting method and a system prototype. We analyze network traffic based on flows, which is defined as collections of packets that travel between the same end points [8]. By aggregating packets that belong to the identical flow, we can reduce processing overhead in the system. In addition, we can easily find flow generating system or routers, such as NetFlow [9]. We characterize traffic patterns that appear during attacks. By using these traffic patterns, the proposed method can detect even mutant attacks that use a new port number or forged payload. Additionally, the method will identify attacks that cannot be detected by examining only packet information, by using complete traffic information.

The organization of this paper is as follows. The related work is described in Section 2. Section 3 describes our proposed abnormal traffic detection algorithm. The analysis result of our proposed algorithm is presented in Section 4. In Section 5, we describe a system prototype that implemented our detection algorithm. Finally, concluding remarks are given and possible future work is mentioned in Section 6.

## 2. Related Work

In this section, we provide a brief overview of scanning and DoS/DDoS attacks. Also, previous approaches to detect these types of attacks are discussed.

Through scanning, an attacker obtains information on a target system. By sending scanning packets to the target, it discovers which systems are working and which services are being offered [10]. DoS/DDoS attacks cause a waste of the resources in the host or networks, and make services work improperly. There are two principal classes of DoS/DDoS, logic and flooding attacks [11].

Logic attacks exploit existing software flaws to cause a malfunction in the system. For instance, in a Ping-of-Death attack [12], oversized ICMP ping packets can result in a denial of service. Also, a Land attack [13] may crash the system by sending packets with the source host and port the same as the destination host and port.

Flooding attacks transmit many spurious packets to the target system, and waste CPU, memory, and network resources. In case of TCP SYN flood [14], the victim receives packets that exceed buffer of the data structure limit and cripple its

service. Also, ICMP, TCP, and UDP flooding attacks [16] overwhelm bandwidth by sending useless traffic to the victim. Some attacks, such as Smurf [15] and Fraggle [17], amplify traffic by using reflecting services of the third party. There are other examples of a reflecting attack that cause packets to rebound between two hosts using reflecting ports, such as echo. We defined this kind of attacks as a Ping-Pong attack in this paper.

To detect the attacks described above, network-based Intrusion Detection Systems (NIDS), such as snort [19], in a packet header or payload. Signature-based detecting systems require a huge database that contains information on every attack. It causes much system overhead to compare every packet with the signatures in the database. Therefore, these systems are not appropriate in a high-speed network. Thus, if a new or mutant attack appears, the signature detecting method cannot catch it. In addition, packet information may be insufficient because some types of attack can be detected only by using from a series of packets information.

Other types of detecting methods have been suggested. These approaches monitor the volume of traffic which every single host has received [20, 21] or the number of new source IP addresses [22]. These methods identify attacks by checking the volume of traffic or the number of new source IP addresses. These methods may have low overhead, but can result false alarms. To reduce false alarms, it is necessary to use all the possible traffic parameters.

## 3. Flow-based Abnormal Traffic Detection Algorithm

In this section, we propose a method to detect abnormal network traffic. In this paper, we define abnormal network traffic as the traffic cause by malicious purpose including the traffic by DoS/DDoS, Internet worm and scanning. The detection module receives flow information from monitoring systems or routers. After detecting abnormal traffic, an alarm is emitted if an attack is detected. As illustrated in Figure 1, the overall process consists of two parts: the flow header detection and the traffic pattern detection.

The flow header detection takes part in checking the fields of the flow headers. By validating these values, this part mainly detects logic attacks. Also, it can catch some flooding attacks with specific values. For example, a fraggle attack traffic can be detected by verifying broadcast destination or specific port numbers.

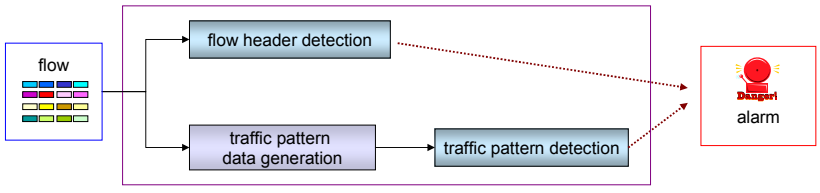


Figure 1. Overall Detection Process

Some attacks have traffic patterns that cannot be characterized by only one flow. To detect this type of attack, we need traffic information that can identify traffic patterns. Aggregating related flows can generate this information, which is called traffic pattern data. By examining parameters of traffic pattern data we can discover traffic used in attacks, such as flooding and scanning.

## 3.1. Detection from Flow Header

The flow header detection part checks the field values of a flow header. The diagram in Figure 2 classifies attacks by the field values of the flow header. This part can detect logic attacks or other attacks with a specific header such as broadcast destination or specific port numbers.

We define flow as a collection of packets with the same 5-tuple: source IP address, destination IP address, source port, destination port, and protocol number. The flow size and packet count, refer to the total bytes and the number of packets that belongs to the flow, respectively.

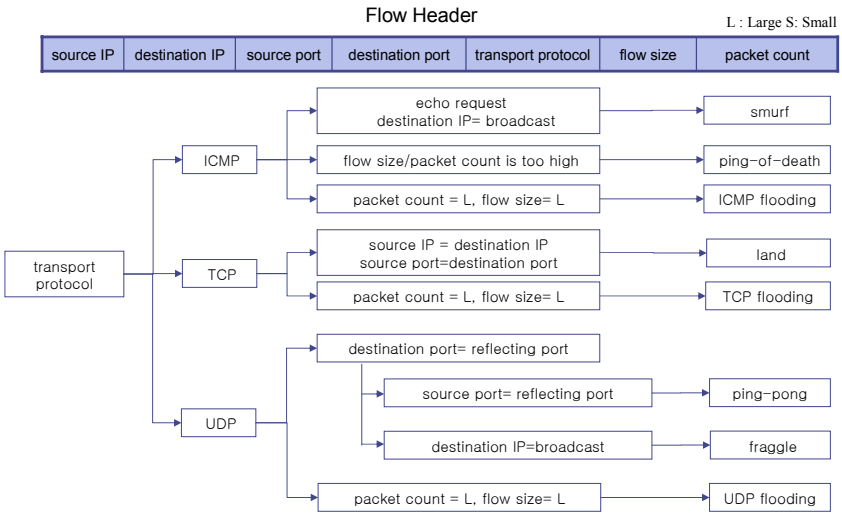


Figure 2. Flow Header Detection Sequence

If the transport protocol is ICMP and its type is echo request and destination is broadcast, then this flow is determined to be a smurf attack. The reason is that the attack mainly sends spoofed source IP packets to the destinations of broadcast. Additionally, this phase can detect a Ping-of-Death attack flow by validating whether the length of the de-fragmented packet is larger than the limited length that an IP packet can have.

In the case of a TCP transport protocol; this part certifies if the pair of source IP, source port is identical with the pair of destination IP, destination port for the purpose of detecting a Land attack.

In UDP flows, Fraggle and Ping-Pong attacks use UDP reflecting services, such as echo (port 7), chargen (port 19), daytime (port13), and qotd (port 17). Therefore, the port numbers of source and destination port are validated. If both destination and source ports are reflecting port numbers, then this flow is used for the Ping-Pong attack. Also, if the destination port is a reflecting port and the destination IP is a broadcast address, the flow is supposed to be a Fraggle attack, similar to the Smurf attack.

In each transport protocol, the flow header detection part searches flows with a large packet count and flow size in order to identify flooding flows. For example, if a large number of ICMP packets and its flow size is large, then it is determined to be an ICMP flooding. To determine whether the flow size and packet count of a flow are large or not, currently we are using a percentile threshold value to total flow size and total packet count. This threshold values are differently determined according to the network and link conditions, such as the number of flows, the number of distinct IP address appeared in the captured data, and so forth.

## 3.2. Detection from Traffic Patterns

Some peculiar traffic patterns are generated during attacks. For detecting this type of attack, we characterize these patterns by parameters of traffic based on flow as shown in Table 1.

L : Large S : Small

| Attacks / Property | scanning | | flooding | | | | |
|---|---|---|---|---|---|---|---|
| | host | network | TCP SYN | smurf | fraggle | ping-pong | general (ICMP,UDP,TCP) flooding |
| flow count | L | L | L | L | L | S | L or S |
| flow size/flow | S | S | S | L or S | L or S | L | L or S |
| packet count/flow | S | S | S | L or S | L or S | L | L or S |
| packet size | S | S | S | L or S | L or S | L or S | L or S |
| total bandwidth | L or S | L or S | L or S | L | L | L | L |
| total packet count | L or S | L or S | L or S | L | L | L | L |
| property | 1 destination | 1 port | | ICMP broadcast | UDP broadcast | reflecting port | |

Table 1. Characterization of Attack Traffic Patterns

During scanning, the attacker makes many connection attempts. Consequently, many flows are generated and the packet count in each flow is small, when a scanning occurs. In addition, the packet size is mostly small (about 40 bytes), because the attacker sends small packets and observes responses from these packets. If an attacker attempts to check open ports in a host, then this host

scanning causes traffic which has a specific destination IP address. On the other hand, a network scanning makes many destination IP addresses when searching for service availability in many hosts of the network. However, the total packet count and total bandwidth can be large or small according to the number of connected hosts and ports. These fields cannot be used to detect scanning.

The TCP SYN flood induces a lot of flow activities, because it sends many packets to a specific port of the victim. Also, the packet count and total packet length in each flow are small, as this attack sends small SYN packets. However, the total bandwidth and total packet count vary according to the number of transmitted packets.

The Smurf and Fraggle attacks force traffic gathered to the victim by using a third party. This type of attack creates as many flows as the number of hosts of the third party used in attacking. Consequently, the total bandwidth and total packet count increase. These attacks use third party and amplify traffic that is mainly destined to a broadcast address. Two attacks, Smurf and Fraggle, differ in the used protocol. Smurf makes use of ICMP and Fraggle uses UDP. As the number of repetition of transmission of spoofed packets determines the packet count in a flow, the total length of packets in each flow, and each packet size, these parameters are unavailable for detecting.

During a Ping-Pong attack, traffic appears only in the two hosts with the same ports. This can cause a large number of packets in a flow. Accordingly, the total packet length in each flow, total bandwidth, and total packet count are large.

In addition to the attacks described above, general ICMP, UDP, TCP flooding attacks have dynamic traffic patterns depending on how many packets and hosts are used for an attack. However, most attacks create a large total bandwidth and high total packet count. Additionally, such traffic has a small deviation in the packet and flow size of each flow.

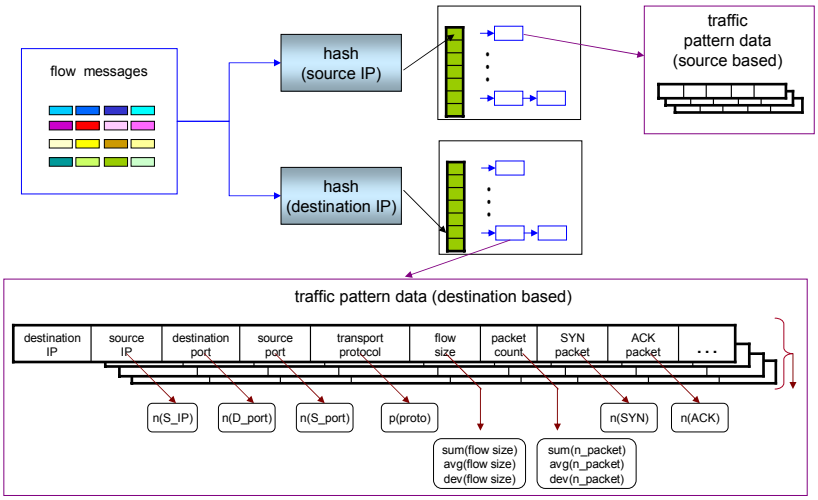


Figure 3. Generation of Traffic Pattern Data

When discovering the traffic patterns described in Table 1, some attacks are not possible to detect only with the flow information. Therefore, we generate traffic pattern data by aggregating related flows. In order to check traffic characteristics, we generate traffic information that are sent and received from a certain host.

As illustrated in Figure 3, the process aggregates flows that contain the same address by hashing. Two hash tables are used to record the traffic pattern data aggregated by either the same source or destination IP. During this phase, two types of traffic pattern data are generated: the source and destination based traffic pattern data that gather flows with the same source and destination IP, respectively.

The parameters and their description of the destination-based traffic pattern data are given in Table 2. The explanation of source-based traffic pattern data is omitted because of the similarity to that of destination-based traffic patterns.

| Notation | Description |
|---|---|
| *n(flow)* | the number of flows with same destination IP |
| *n(S_IP)* | the number of distinguished source IP with same destination IP |
| *n(D_port)* | the number of destination port with same destination IP |
| *n(S_port)* | the number of source port with same destination IP |
| *p(proto)* | the most frequently appeared transport protocol with same destination IP |
| *sum(flow size)*<br>*avr(flow size)*<br>*dev(flow size)* | Summation, average, and deviation of flow size with same destination IP |
| *sum(n_packet)*<br>*avr(n_packet)*<br>*dev(n_packet)* | Summation, average, and deviation of packet count with same destination IP |
| *n(SYN)*<br>*n(ACK)* | the total number packets of SYN, ACK, and other flags with same destination IP |

Table 2. Network Parameters used Anomaly Detection

By comparing attack traffic patterns with parameters of traffic pattern data generated in Figure 3, the traffic pattern detection part can identify abnormal network traffic. The comparison algorithm is illustrated in Figure 4.

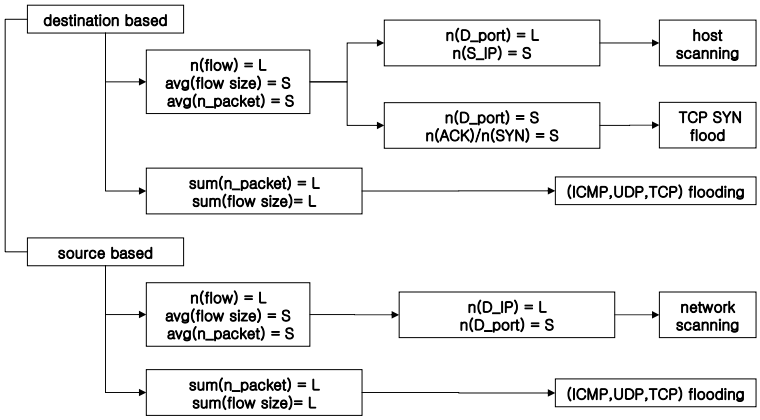


Figure 4. Traffic Pattern Detection Sequence

When checking a destination-based traffic pattern data, the detector checks whether a large number of flows appears, whether the flow size of an individual flow is small, and whether the number of packets per a flow is small. If so, and if the number of destination ports is high, and a small number of source IP traffic is generated, then that traffic is assumed to be a host scanning. If the traffic pattern data reports a small number of destination ports and a small fraction of *n(ACK)*/*n(SUN)*, this phenomenon implies that a TCP SYN flood attack traffic has occurred.

In addition, a source-based traffic pattern data is examined to investigate the traffic sent from a specific host. The detector checks whether the *n(flow)* is large, the *avg(flow size)* is small, and the *avg(n_packet)* is small in a manner similar to destination-based traffic pattern data. The detector checks if the data reports a large number of destination IPs and a small number of destination ports. If so, that traffic is suspected to be a flooding attack.

Regardless of the source and destination of traffic pattern data, traffic sent or received from a certain machine is investigated. The reason is that the system may use network resources by sending or receiving too much traffic when it is used as a flooding attack. Accordingly, if the analyzed result from traffic data indicates too many total packets and bandwidth, then that traffic is considered to be a flooding attack.

We are using threshold values to decide whether each traffic pattern parameter is large or small at each phase of the traffic pattern detection sequence in Figure 4. The threshold values are differently determined for each traffic pattern parameter with regard to each attack type. We do not use these threshold values directly in the detection sequence. Instead, we use these values in the attack detection functions to increase detection accuracy, which is described in the following section.

## 4. Formalization of Detecting Functions and Thresholds

In this section, we describe experimental results of the proposed network security attack detection method. From these results, we formalize detection functions suitable for attack detection, which are composed of several traffic pattern parameters and constant values. Using these detections functions and threshold values we can determine whether or not certain traffic is abnormal.

We have setup a security attack testing environment in our laboratory and generated various attack traffic using freely available attack tools. Figure 5 shows variations of traffic pattern parameters in the form of time series graph. A scanning and a TCP SYN flood attack are occurred at time t1 and t2 respectively. We observed traffic data that the victim receives. The time series graphs (a)~(d) in Figure 5 illustrate the variation of each network traffic parameter (such as *sum(flow_size)*, *n(flow)*, *n(S_IP)*, etc.) in the victim side.

Although some parameters, such as *n(flow)*, partly reflect the traffic changes, the decision with only one individual parameter may generate a false alarm appeared at t3, and this phenomenon implies the occurrence of scanning or TCP

SYN flood. However, we discover that many of these flows are caused by a popular peer-to-peer application called 'eDonkey' [23]. As shown in Figure 5, the use of a single parameter in the detection of these attacks cannot give high detection accuracy because many traffic patterns from newly born network-based applications are similar to the attack traffic.

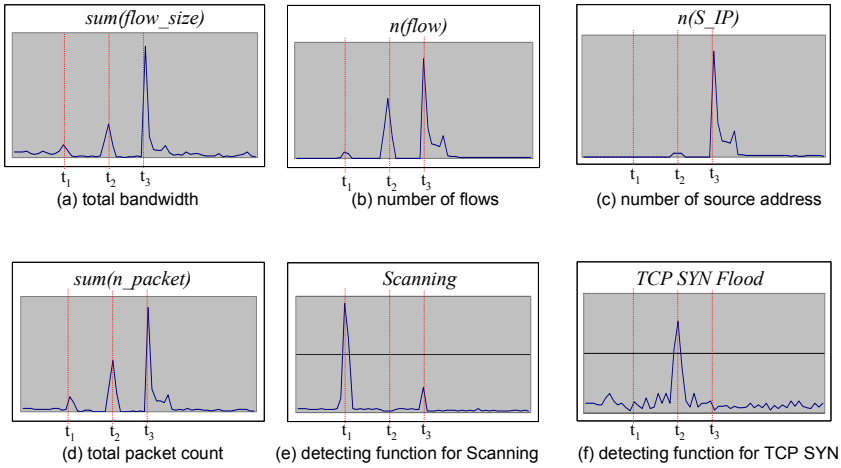


Figure 5. A Result of the Proposed Detection Algorithm

In order to detect attacks, we use detecting functions that could fully reveal the traffic patterns of attacks. Figure 5 (e) and (f) represent values of functions to detect a scanning and TCP SYN flood, respectively. These detecting functions for the scanning and TCP SYN flood attack are formalized as follows:

$$f_{scan} = v_{N\_flow} * \alpha_{flow} + v_{L\_flow} * \alpha_{L\_flow} + v_{N\_packet} * \alpha_{N\_packet} + v_{IP} * \alpha_{IP} + v_{port} * \alpha_{port}$$

where $v_{N\_flow} = n(flow) / T_{N\_flow}, v_{L\_flow} = T_{L\_flow} / avg(flow\ size)$,
$v_{N\_packet} = T_{N\_packet} / avg(n\_packet), v_{IP} = T_{IP} / n(S\_IP), v_{port} = n(D\_port) / T_{port}$;

$$f_{syn\_flood} = v_{N\_flow} * \alpha_{flow} + v_{L\_flow} * \alpha_{L\_flow} + v_{N\_packet} * \alpha_{N\_packet} + v_{port} * \alpha_{port} + v_{syn\_ack} * \alpha_{syn\_ack}$$

where $v_{N\_flow} = n(flow) / T_{N\_flow}, v_{L\_flow} = T_{L\_flow} / avg(flow\ size)$,
$v_{N\_packet} = T_{N\_packet} / avg(n\_packet),\ v_{port} = T_{port} / n(D\_port), v_{syn\_ack} = n(SYN) / n(ACK)$.

$\alpha_X$ refers to the weight of term $X$, and $T_Y$ means the threshold of term $Y$. The values used in our detection are given in Table 3. To determine the suitable weight and threshold values, we experienced lots of trial and errors. These functions can identify a scanning at t1 and TCP SYN flood at t2. Currently, we are using static threshold values, which are determined by the comparison of normal traffic and

attack traffic. To generate attack traffic we used freely available attack tools. The threshold value of same traffic pattern parameter, such as $T_N$, is different according to attack type. The threshold and weight values need to be elaborate to adopt this proposed system in various network environments.

| Attack | Type | Value |
|---|---|---|
| Scanning | weight | $\alpha_{N\_flow} = 0.3$, $\alpha_{L\_flow} = 0.1$, $\alpha_{N\_packet} = 0.2$, $\alpha_{IP} = 0.1$, $\alpha_{port} = 0.3$ |
| | threshold | $T_{N\_flow} = 1024$, $T_{L\_flow} = 128$, $T_{N\_packet} = 2$, $T_{IP} = 3$, $T_{port} = 1024$ |
| TCP SYN flood | weight | $\alpha_{N\_flow} = 0.2$, $\alpha_{L\_flow} = 0.1$, $\alpha_{N\_packet} = 0.1$, $\alpha_{port} = 0.1$, $\alpha_{syn\_ack} = 0.5$ |
| | threshold | $T_{N\_flow} = 3500$, $T_{L\_flow} = 64$, $T_{N\_packet} = 1$, $T_{port} = 1$ |

Table 3. The constant and threshold value of proposed detection function

## 5. Prototype Implementation

We have developed a system prototype for detecting abnormal network traffic based on flows. This system utilizes flow information from NG-MON [7]. As illustrated in Figure 6, the monitoring tasks of NG-MON are divided into several phases, which are serially interconnected using a pipelined architecture. One or more systems may be used in each phase to distribute and balance the processing load. This provides good scalability. We have also defined a communication method between each pair of phases. Each phase can be replaced with more optimized modules as long as they provide and use the same interfaces. The divided architecture provides flexibility. By assigning tasks to each phase, this architecture enables us to easily append or remove modules for added work, such as security attack analysis.

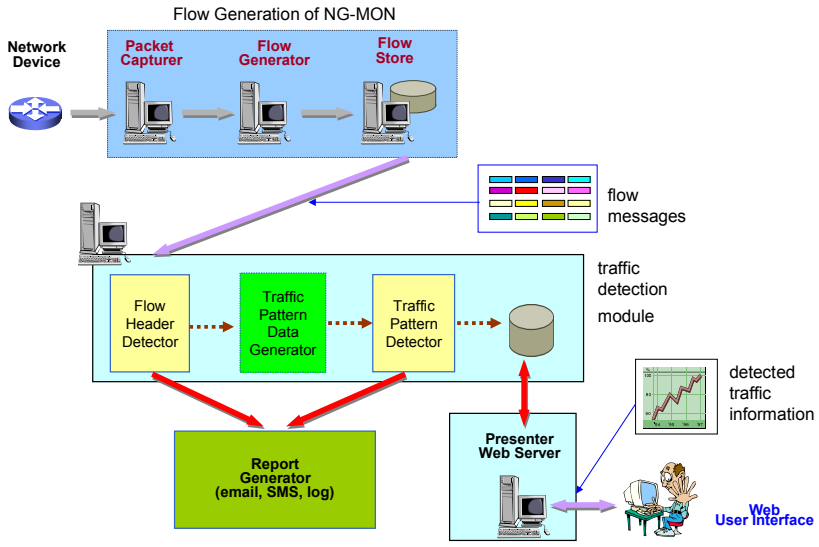


Figure 6. Abnormal Traffic Detection System Architecture

The flow generation module of NG-MON provides flow information. This module consists of a packet capturer, a flow generator, and a flow store. The packet capturer collects packets that pass a probing point. Another function of the packet capturer is to extract information from the packet header and to send it to the flow generator. Then, the flow generator creates a flow by collecting a series of packets. Next, the flow is periodically stored into a database of the flow store. Here, the period can be configured according to the flow time-out in order to aggregate flow information during a predetermined time, such as one minute.

The traffic detection module discerns abnormal network traffic. This module checks the flow header fields first to discover specific addresses, port numbers, or logic attacks. Then, it generates traffic pattern data. By matching this data, detect functions identify the attacks, as described in Figure 6.

If abnormal network traffic is detected, the report on attack can be provided to the network administrator by email, SMS, and log. Also, the presenter shows information on detected abnormal network traffic, by replying to user's request through the Web user interface. Figure 7 is a sample screen shot of web-based user interface.

We implemented the traffic detection module using C language and MySQL on Linux environment. The Apache web server and PHP are used to show the abnormal traffic information which is stored in the MySQL DB. We used a single Pentium III 800 MHz system with 256Mbytes memory for the traffic detection module to analyze traffic flow data captured in total 400 Mbps network links. Our prototype system has been deployed in the our campus Internet junction, and gives useful information about abnormal traffic to the campus network administrators.

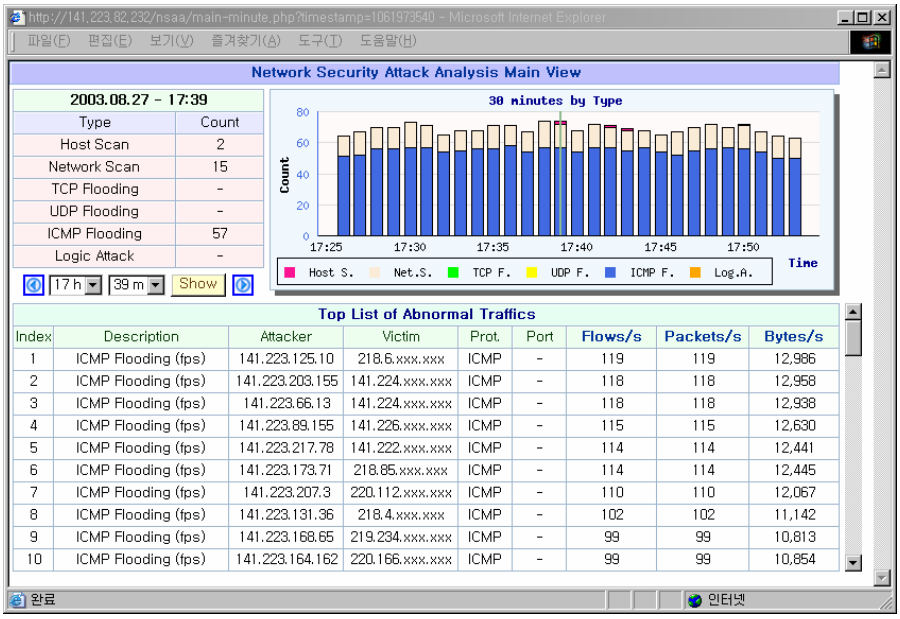

Figure 7. A User Interface for Security Attack Analysis System

Figure 7 shows an analysis results for 1 minute traffic flow in Aug. 27, 2003. As you can see, Lots of ICMP flooding attacks are detected which is caused by Welchia Internet worm [25].

## 6. Conclusion

This paper presented a flow-based abnormal network traffic detection method and its system prototype. This method is efficient, since it can reduce system overhead in the processing of packet data by aggregating packets into flows. It can detect the traffic of several attacks with a similar traffic patterns using one detecting function. This function can cover even mutant attacks that use new port numbers or a changed payload. We also increased the detecting accuracy. When detecting abnormal traffic, we use parameters that can reflect changes in traffic characteristics during attacks. The traffic information of this system is extensible. Traffic pattern data extracted from a group of flows include various types of information. Therefore, this information can be easily compounded to detect new types of attacks.

However, the proposed method strictly focuses on the detection of DoS/DDoS attacks. If an attack does not influence network traffic, it is difficult to detect this type of attack. In future, we plan to develop detecting algorithms that can detect more attacks. Furthermore, the traffic pattern of some P2P applications, which occupies more than 50% of current Internet traffic, is very similar to attack traffic pattern. It is necessary to reduce the false-alarm cause by these P2P traffic. Thirdly, currently we are using static threshold values in the detection function, which is determined by lots of experimentation. But this value can not be suitable to every network environment. So, we need to find a new method to determine the threshold value adaptively for various network conditions.

## References


[1] CNN, Immense network assault takes down yahoo, February 2000, http://www.cnn.com/2000/TECH/computing/02/08/yahoo.assault.idg/index.html.

[2] CNN, Cyber-attacks batter web heavyweights, February 2000, http://www.cnn.com/2000/TECH/computing/02/09/cyber.attacks.01/.

[3] Drew Dean, Matt Franklin, and Adam Stubblefield, “An algebraic approach to ip traceback,” Proc. of Network and Distributed System Security Symposium, NDSS '01, San Diego, California, February 2001.

[4] L. John Ioannidis and Steven M. Bellovin, “Implementing pushback: Router-based defense against DDoS attacks,” Proc. of Network and Distributed System Security Symposium, NDSS '02, San Diego, California. February 2002.

[5] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson, "Practical network support for IP traceback," Proc. of the 2000 ACM SIGCOMM, Stockholm, Sweden, August 2000.
[6] Kun-chan Lan, Alefiya Hussain, and Debojyoti Dutta, "Effect of Malicious Traffic on the Network," Proc. of PAM 2003, San Diego, California, April 2003.
[7] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju, and James W. Hong, "The Architecture of NG-MON: A Passive Network Monitoring System," Lecture Notes in Computer Science 2506, 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2002), Montreal, Canada, October 2002, pp. 16-27.
[8] Siegfried Lifler, "Using Flows for Analysis and Measurement of Internet Traffic," Diploma Thesis, Istitute of Communication Network and Computer Engineering, University of Stuttgart, Germany, 1997.
[9] Cisco, White Papers, "NetFlow Services and Applications," http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm.
[10] Fyodor, "The Art of Port Scanning," http://www.insecure.org/nmap/nmap_doc.html.
[11] D. Moore, G. M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," Proc. of USENIX Security Symposium, Washington D.C, Aug 2001.
[12] Common Vulnerabilities and Exposures (CVE), "Ping-of-Death (CVE-1999-0128)," http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0128.
[13] Common Vulnerabilities and Exposures (CVE), "Land (CVE-1999-0016)," http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016.
[14] Common Vulnerabilities and Exposures (CVE), "SYN flood (CVE-1999-0116)," http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0116.
[15] Common Vulnerabilities and Exposures (CVE), "Smurf (CVE-1999-0513)," http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0513.
[16] Common Vulnerabilities and Exposures (CVE), "UDP packet storm (CVE-1999-0103)," http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0103.
[17] Common Vulnerabilities and Exposures (CVE), "Fraggle (CVE-1999-0514)," http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0514.
[18] Common Vulnerabilities and Exposures (CVE), "HTTP request flood (CVE-1999-0867)," http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0867.
[19] M. Roesch, "Snort - lightweight intrusion detection for networks," http://www.snort.org.
[20] Rudolf B. Blazek, Hongjoong Kim, Boris Rozovskii, and Alexander Tartakovsky, "A novel approach to detection of denial-of-service attacks via adaptive sequential and batch sequential change-point detection methods," Pro. of IEEE Systems, Man and Cybernetics Information Assurance Workshop, New York, June 2001.
[21] David K. Y. Yau, John C. S. Lui, and Feng Lian, "Defending against distributed denial-of-service attacks with max-min fair server-centric router

throttles," Proc. of IEEE International Workshop on Quality of Service (IWQoS), Miami Beach, Florida, May 2002.
[22] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao, "Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring," http://www.ee.mu.oz.au/pgrad/taop/research/detection.pdf.
[23] eDonkey, http://www.edonkey2000.com/.
[24] Urupoj Kanlayasiri, Surasak Sanguanpong, and Wipa Jaratmanachot, "A Rule-based Approach for Port Scanning Detection," Proc. of the 23nd Electrical Engineering Conference (EECON-23), Chiangmai, Thailand, November 2000.
[25] Welchia Internet Worm, http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html.