Towards Flow-based Abnormal Network Traffic Detection

Hun-Jeong Kang, Seung-Hwa Chung, Seong-Cheol Hong, Myung-Sup Kim and James W. Hong

DP&NM Lab.
Dept. of Computer Science and Engineering
Pohang University of Science and Technology
Email: {bluewind, mannam, pluto80, mount, jwkhong}@postech.ac.kr

POSTECH
DP&NM Lab

One recent trend in network security attacks is increase in indirect attack on systems. Instead of directly entering the system, an attacker interferes with the proper working of the network or system. In future, damage from this type of attack is expected to become more serious. In addition, the bandwidth usage of those 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 detection algorithm using changes in traffic patterns that appear during attacks. The proposed algorithm can also detect mutant attacks that use a new port number or changed payload while signature based systems are not capable of detecting these kinds of attacks. Additionally, the algorithm will identify attacks that cannot be detected by examining only packet information, by using entire traffic information. We present the design of a flow-based abnormal network traffic detection using the proposed algorithm.

Introduction

- Motivation : network security attack threats
  - increase of the number of Internet user & Network services
    → highly dependent on Internet
  - high damage cost by network security attack
- Attack Trend : increase of indirect attacks
  - indirectly interfere with proper working of system
  - danger factors
    - ease of attack, fast impact on target system
    - difficulty tracing back to attacker
    - effect on entire network performance
- Flow-based Abnormal Network Traffic Detection
  - characterize network attack traffic patterns
  - propose detecting algorithms and a system prototype

Today, the number of Internet users is dramatically increasing, along with network services. As the Network grows, network security attack threats become more serious. Many vulnerabilities of security are exposed and exploited by attacks. And recent reports on Internet security breaches say that the frequency and damage costs are continuously rising.

One of the recent network attack trends is the use of indirect cracking. An attacker places networks or hosts in jeopardy, without intruding into the hosts. The attacks on famous website, such as Yahoo, E-bay, and E*trade, are good examples [1, 2]. This type of Denial of Service (DoS) attack [3, 4, 5] will cause more damage for the following reasons. There are many DoS tools that unskilled users can use easily. A successful DoS attack shows its impact quickly and makes it difficult to trace back to the intruder. Moreover, the bandwidth usage of 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 more serious latency deficiency (up to 500%) in the Enterprise network that contains a target or bypassing machine of the attacks. These menaces require us to provide provision against DoS attacks.

This paper focuses on detecting abnormal network traffic generated by DoS attack and its preparation, namely scanning. We present detecting algorithms and a system prototype. We analyze network traffic based on a flow, which means the collection of packets that travels 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 obtain flow information, such as Netflow [9], from many traffic monitoring systems or routers. We characterize traffic patterns that appear during attacks. By using these traffic patterns, the suggested algorithms can detect even mutant attacks that use a new port number or changed payload. Additionally, the algorithms will identify attacks that cannot be detected by examining only packet information, by using complete traffic information.

Related Work

- Network Security Attacks
  - Scanning : preparation for attacks
    - send scanning packets to the victim & receive responses
      → gather information on the host and services
  - DoS : cause improper services
    - logic attack
      - send packets exploiting software flaws → malfunction in the system
      - examples: Ping of Death, Land
    - flooding
      - transmit a lot of spurious packets to victim
        → waste of CPU, memory, network resource
      - examples: TCP SYN flood, reflecting DoS(Smurf, Fraggle, Ping-Pong)
        general (ICMP, UDP, TCP) flooding
- previous detecting approaches
  - monitor :
    - patterns that appears in a packet header or payload
    - volume of traffic the victim receives
    - number of new IP addresses that appear

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

Through Scanning, an attacker obtains information on a target. By sending scanning packets to the victim, he/she checks which systems are working and which services are being offered [10].

Dos attacks cause a waste of the resources in the host or networks and make services work improperly. There are two principal classes of DoS : *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. And a Land attack [13] may crash the system by sending a packet with the source host and port the same as the destination host and port.

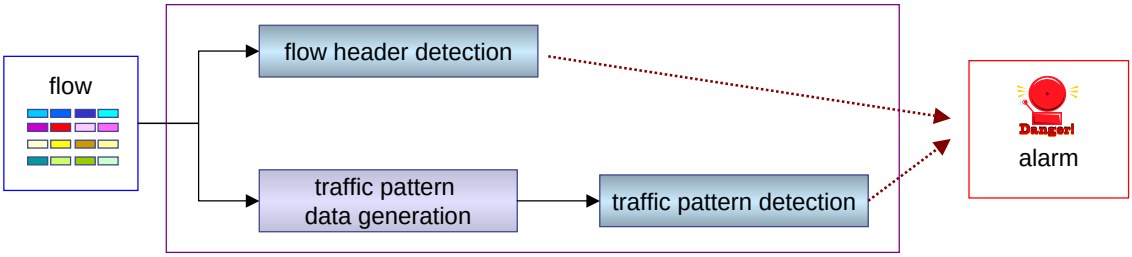
Flooding attacks transmit many spurious packets to the victim, and waste CPU, memory, network resources. In case of TCP SYN flood [14], the victim receives packets that exceed data structure limit and cripple its service. Also, ICMP, UDP, TCP flooding attacks [15, 16, 17, 18] overwhelm bandwidth by sending useless traffic to the victim. Some attacks, such as Smurf [15] and Fraggle [17], amplify traffic by using reflecting services of the third party. And there is another of examples of a reflecting attack that causes packets to rebound between two hosts using reflecting ports, such as echo. This attack is designated as a *Ping-Pong* attack in this paper.

To detect the attacks described above, many Network based Intrusion Detection Systems (NIDS), such as snort [19], check signatures contained 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 proper in a high-speed network. And, 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 complete traffic information.

And other types of detecting methods have been suggested. These approaches monitor the volume of traffic which the victim has received [20, 21] or the number of new source IP addresses [22]. However, these methods idnetify attacks by checking only a few parameters of traffic properties. They may result in false alarms. For better attack detection, it needs to analyze synthetically more parameters of traffic information.

Overall Process

- Two detection parts
  1. flow header detection
     - validate fields of flow header
     - detect logic attacks or attacks with specific field value
  2. traffic pattern detection
     - generate traffic pattern data
     - compare traffic parameter with attack patterns
     - detect scanning and flooding attacks



Detecting Process

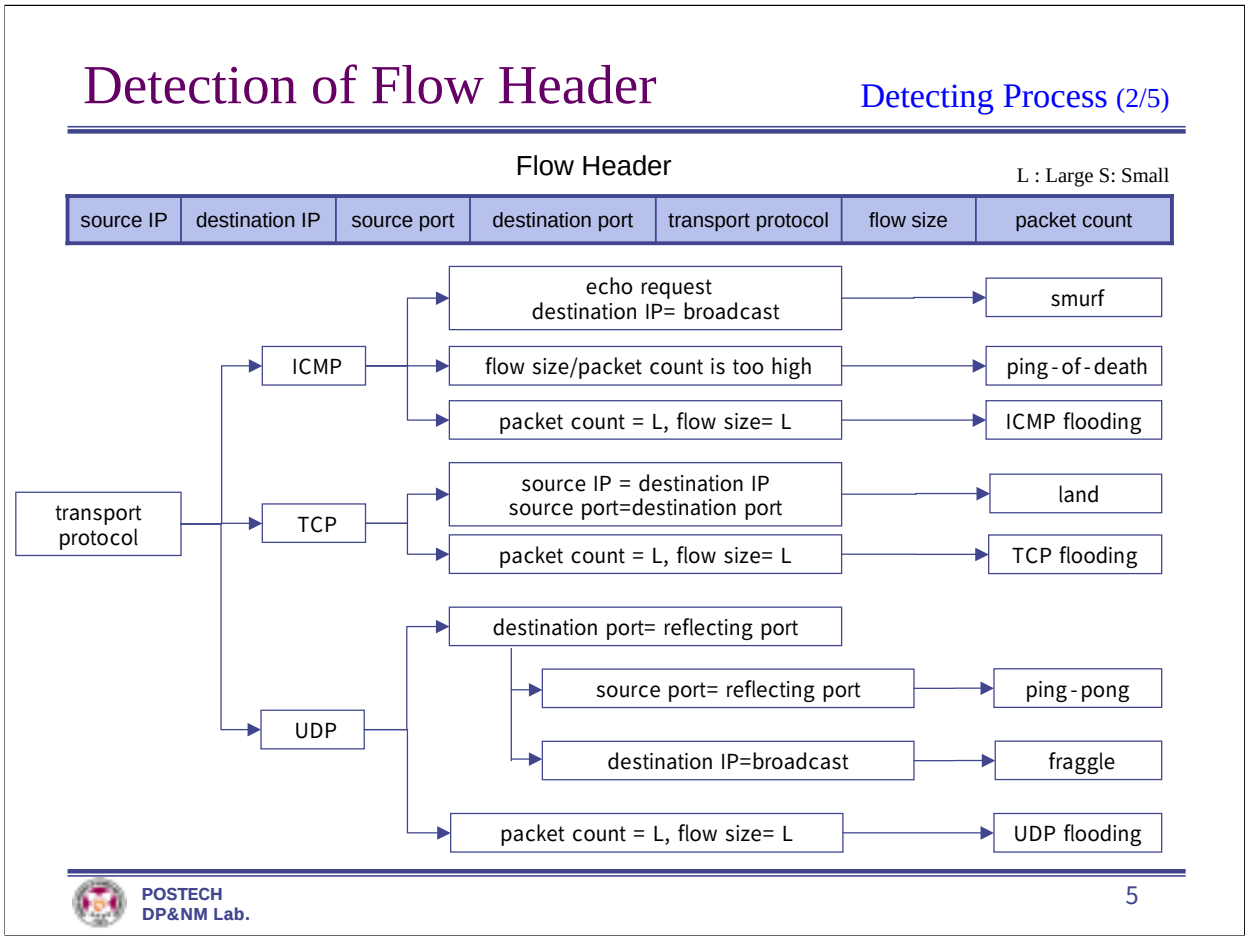
In this slide, we propose a process to detect abnormal network traffic. The process receives flow information from monitoring systems or routers. After detecting process, an alarm is emitted if an attack is detected. As illustrated in the above slide, the overall process consists of two parts: a flow header detection and a traffic pattern detection. We will describe each part minutely in the next slides.

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

Some attacks have traffic patterns that cannot be characterized by only one flow. To detect this type of attack, we need to traffic information that can identify traffic patterns. Aggregating related flows can generate this information, which is called *traffic pattern data*. By comparing parameters of traffic pattern data with the traffic pattern, the traffic pattern detection can discover traffic used in attacks, for instance, flooding and scanning.

Attacks to be detected in each part are described in following table.

Detecting Part	Detected Attack
flow header detection	Ping-of-Death, Land Smurf, Fraggle, Ping-Pong
traffic pattern detection	(host, network) Scanning, TCP SYN flood (distributed) [ICMP, UDP, TCP] flooding



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

A flow header is illustrated in the above slide. We define a flow as a sequence of packets with the same 5-tuple: source IP address, destination IP address, source port, destination port, and protocol number. The others, flow size and packet count, mean the total length and the number of packets that belongs to the flow, respectively.

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

In the case of 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 Ping-Pong attack. Also, if the destination port is reflecting port and the destination IP is broadcast address, the flow is supposed to be a Fraggle attack similar to the reason for a 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.

Characterization of Traffic Patterns Detecting Process (3/5)

L : Large S : Small

	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	

POSTECH
DP&NM Lab.

6

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.

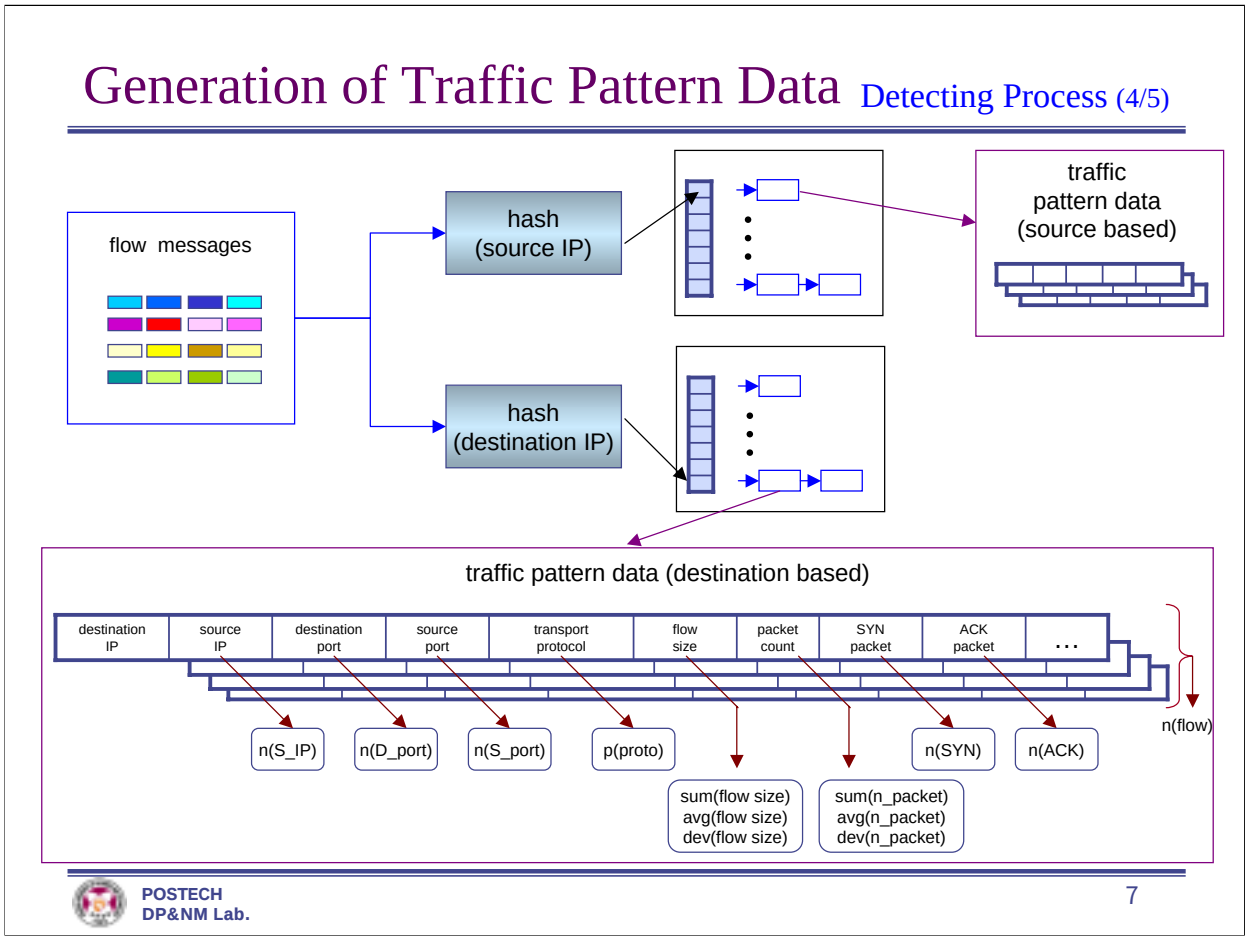
During Scanning, the attacker makes many connection attempts. Consequently, many flows are generated and the packet count in each flow is small, when a Scanning occurs. In addition, the packet size is mostly small, because the attacker sends short packets and observes responses from these packets. When the attacker attempts to check for open ports in the host, the 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 in detecting scanning. Unavailability of fields that varies with other traffic parameters is identical with that in detecting other attacks.

TCP SYN flood induces a lot of flow activity, 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.

Smurf and Fraggle attack 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 broadcast address. Two attacks, Smurf and Fraggle, differ in the used protocol. Smurf makes use of ICMP; however, Fraggle does UDP. As the number of repetition of transmission of spoofed packets determines the packet count in a flow, total length of packets in each flow, and each packet size, these parameters are unavailable for detecting

During a Ping-Pong attack, traffic appears in only the two hosts with the same ports. This can cause the high packet count 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 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.



When discovering traffic patterns described in the previous slide, it is impossible to compare traffic patterns with only 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 the slide, 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: source and destination based traffic pattern data that gathers flows with the same source and destination IP, respectively.

The parameters of the destination-based traffic pattern data are as follows- the explanation of source-based traffic pattern data is omitted because of similarity to that of destination based.

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), avr(flow size), dev(flow size) : Summation, average, and deviation of flow size with same destination IP

sum(n_packet), avr(n_packet), dev(n_packet) : Summation, average, and deviation of packet count with same destination IP

n(SYN), n(ACK), and so on : the total number packets of SYN, ACK, and other flags with same destination IP

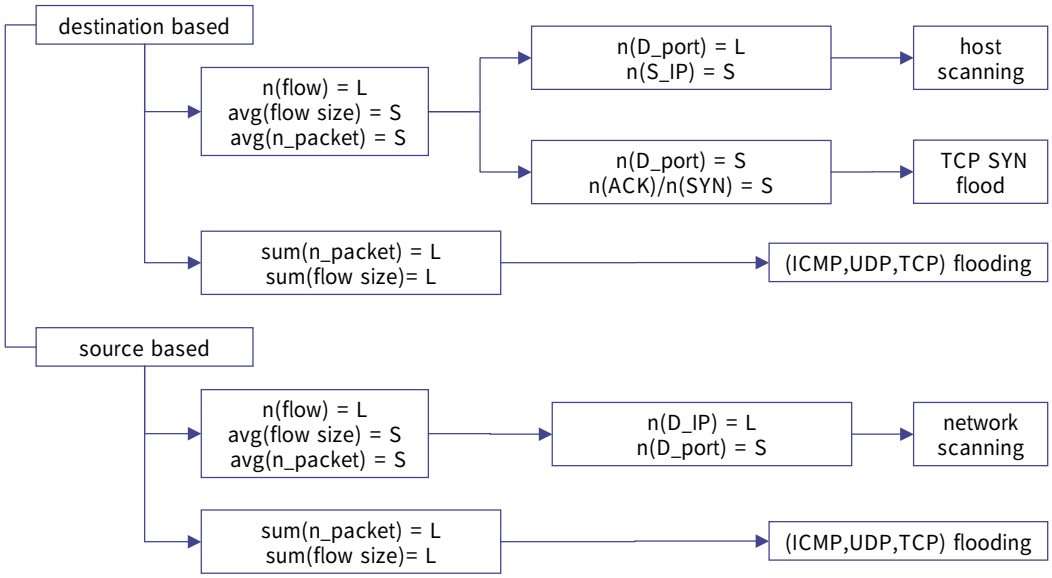
Detection of Traffic Patterns

Detecting Process (5/5)

Detecting Function

L : Large S : Small

- match traffic pattern data with attack type

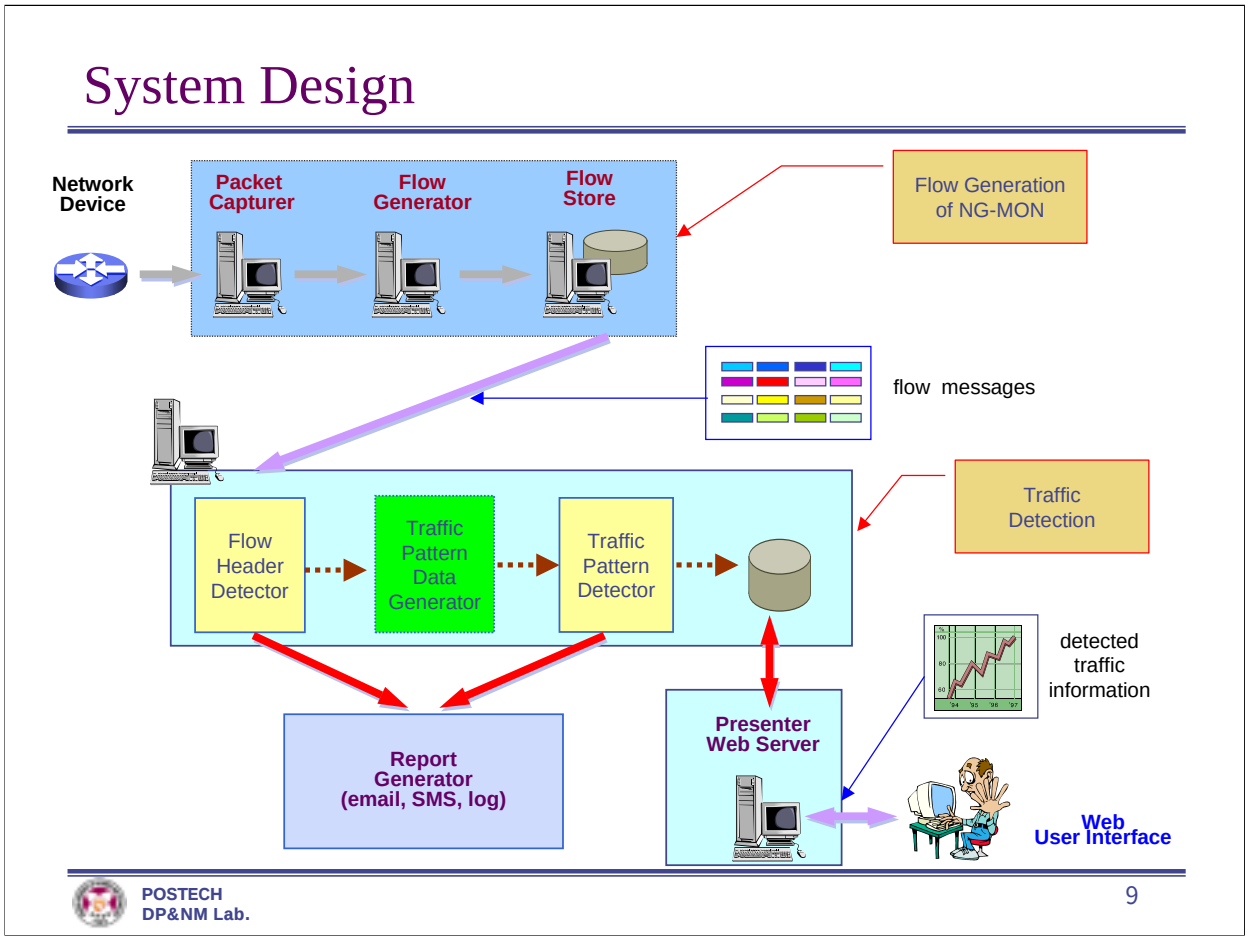


By comparing attack traffic patterns with parameters of traffic pattern data generated in the previous slide, the traffic pattern detection part can identify abnormal network traffic. The comparison algorithm is as follows.

When checking a destination-based traffic pattern data, the detector validates whether a large number of flows appears, the flow size of an individual flow is small, and 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, that traffic is assumed to be a host scanning. If the traffic pattern data reports a small number of destination port 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 for monitoring traffic sent from a specific host. The detector checks whether the n(flow) is large, the avg(flow sizie) is small, and the avg(n_packet) is small in a similar way of destination-based traffic pattern data. However, the algorithm certifies that if the data reports a large number of destination IPs and a small number of destination ports. If so, that traffic is suspected of having experienced a flooding attack.

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



We have developed a system prototype for detecting abnormal network traffic based on flows. This system can receive flow information from NG-MON [7]. As illustrated in the above slide, tasks 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. Each phase performs its defined role in the manner of a pipeline system. This architecture can improve the overall performance, with each phase configured with a cluster architecture for load distribution. 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.

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 passing 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 fields of the flow header 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 the previous slides.

If abnormal network traffic is detected, the report on attack is 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.

Conclusion and Future Work

Abnormal Network Traffic Detection Method

- efficiency : use and process information on flows
- simplicity : detecting function based on traffic pattern data that covers mutant attacks
- accuracy : use of various traffic characteristics as detecting parameters
- extensionality : generation of general traffic pattern data → easiness response new type of attacks by compounding traffic

Future Work

- find algorithm that covers other types of attacks that do not cause changes of traffic
- study schemes to cooperate with traffic monitoring systems existing IDSs
- extend to Intrusion Protection System

This paper presented a flow-based abnormal network traffic detection method and system prototype. This method is efficient. It can reduce system overhead to process packet data, by aggregating packets that belong to an identical flow. And the method provides simplicity. It can detect the traffic of several attacks that have similar traffic pattern by one detecting function. This function can cover even mutant attacks that use new port numbers or a changed payload. We increased accuracy. When detecting abnormal traffic, we use many parameters that can reflect change of traffic characteristics during attacks. The traffic information of this system is extensible. Traffic pattern data extracted from group of flow include various types of information. Therefore, this information can be easily compounded to detect new types of attacks.

However, this method purposed to detect mainly DoS attack that causes network traffic changes. If an attack does not influence network traffic, then it is difficult to detect that type of attack. In future, we plan to find detecting algorithms that can detect many types of attacks. These types include attacks that do not cause changes of traffic. To reduce false reject, we need to study schemes that use traffic history from general traffic monitoring systems and to cooperate with existing IDSs. We are planning to extend our work to Intrusion Protection System (IPS) that offers strong network security management.

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/index.html.

[3] Drew Dean, Matt Franklin, and Adam Stubblefield, "An algebraic approach to ip traceback," In Network and Distributed System Security Symposium, NDSS '01, February 2001.

[4]LJohn Ioannidis and Steven M. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," In Proceedings of Network and Distributed System Security Symposium, California. February 2002.

[5] Stefan Savage, David Wetherall, Anna Karlin and Tom Anderson, "Practical network support for IP traceback," In Proceedings of the 2000 ACM SIGCOMM Conference, August 2000.

[6] Kun-chan Lan, Alefiya Hussain, and Debojyoti Dutta, "Effect of Malicious Traffic on the Network," In the Proceedings of PAM 2003, 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), October 2002.

[8] Siegfried Lifler, "Using Flows for Analysis and Measurement of Internet Traffic," Diploma Thesis, Istitute of Communication Network and Computer Engineering, University of Stuttgart, 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," Phrack Magazine Volume 7, article 11, Issue 51, September 1997.

[11] D. Moore, G. M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," In Proceedings 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 batchsequential change-point detection methods," In Proceedings of IEEE Systems, Man and Cybernetics Information Assurance Workshop, 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," In Proceedings of IEEE International Workshop on Quality of Service (IWQoS), FL, 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.