available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Fast detection and visualization of network attacks on parallel coordinates

Hyunsang Choi, Heejo Lee, Hyogon Kim*

Division of Computer and Communication Engineering, Korea University, Anam-dong, Seongbuk-gu, Seoul 136-713, South Korea

ARTICLE INFO

Article history:
Received 7 August 2007
Received in revised form
17 November 2008
Accepted 12 December 2008

Keywords:
Internet attack visualization
Parallel coordinates
Internet worms
DDoS attacks
Parallel coordinate attack visualization (PCAV)

ABSTRACT

This article presents what we call the parallel coordinate attack visualization (PCAV) for detecting unknown large-scale Internet attacks including Internet worms, DDoS attacks and network scanning activities. PCAV displays network traffic on the plane of parallel coordinates using the flow information such as the source IP address, destination IP address, destination port and the average packet length in a flow. The parameters are used to draw each flow as a connected line on the plane, where a group of polygonal lines form a particular shape in case of attack. From the observation that each attack type of significance forms a unique pattern, we develop nine signatures and their detection mechanism based on an efficient hashing algorithm. Using the graphical signatures, PCAV can quickly detect new attacks and enable network administrators to intuitively recognize and respond to the attacks. Compared with existing visualization works, PCAV can handle hyper-dimensions, *i.e.*, can visualize more than 3 parameters if necessary, which significantly reduces false positives. As a consequence, Internet worms are more precisely detectable by machine and more easily recognizable by human. Another strength of PCAV is handling flows instead of packets. Per-flow visualization greatly reduces the processing time and further provides compatibility with legacy routers which export flow information, *e.g.*, as NetFlow does in Cisco routers. We demonstrate the effectiveness of PCAV using real-life Internet traffic traces. The PCAV program is publicly available.

1. Introduction

Plenty of intrusion detection techniques have been proposed so far, but they still have weaknesses. Conventional intrusion detection systems, which are based on known attack signatures, cannot detect unknown attacks. Intrusion detection systems based on anomaly detection mechanisms on the other hand can recognize some variants of known attacks, but they often generate a huge number of false alarms which overwhelm security engineers and render the security systems worthless.

To overcome the drawback, one promising approach is visualizing complex situations using a simple and intuitive way (Keim, 2001). Humans can easily recognize and intuitively infer patterns from complex visual images. The visual images can be obtained from raw data using computer graphics techniques and algorithms (Kim et al., 2004; Conti and Abdullah, 2004).

In this article, we introduce a simple but novel way of visualizing Internet attacks on parallel coordinates. Parallel coordinates have many desirable properties such as representing more than three values in a two dimensional space (Inselberg, 1985). In order to visualize most popular attacks such as Internet worms, we have carefully selected four fields available on most flows. And, nine graphical signatures are developed to detect ongoing attacks, which include Internet

* *Corresponding author*. Tel.: +82 2 3290 3208; fax: +82 2 953 0771.
E-mail addresses: realchs@korea.ac.kr (H. Choi), heejo@korea.ac.kr (H. Lee), hyogon@korea.ac.kr (H. Kim).
0167-4048/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2008.12.003

worms, DDoS (Distributed Denial of Service) attacks and network scanning attacks. Furthermore, we devise an O(1) hashing algorithm to identify these signatures. The effectiveness of the proposed approach is shown by running on real network traffic and revealing hidden attacks in a visual way. It is shown that this mechanism works for detecting notorious Internet attacks such as rapidly spreading Internet worms, which have not been investigated extensively in visualization studies such as in Kim et al. (2004).

We use flows for input data, instead of packets, because of performance, scalability and compatibility with legacy routers. A flow is a single network connection and can consist of millions of packets. Handling flow-level information greatly reduces the processing time so it enables the visualization algorithm to run on high-speed links. Furthermore, many legacy routers provide flow information and they are widely deployed, which includes NetFlow in Cisco routers. This compatibility with legacy routers greatly enhances the usability of the visualization mechanism.

The aim of this study is not to propose a new visualization technique. The main contribution is how to use parallel coordinates to display and detect Internet attacks. Displaying network flows using carefully chosen values forms a unique graphical image for each type of attack. And, such an attack can be detectable by the use of its graphical signature, even though the attack is not known a priori–*unknown attack*. Comparing with the 3-D visualization work (Kim et al., 2004), we can detect more attacks including Internet worms and greatly reduce false alarms on flash crowds and P2P applications. Recently proposed visualization techniques tend to simply display the network traffic and do not recognize attacks, thus require the human administrator to be constantly involved in monitoring. However, our system is designed to be intelligent enough to automatically detect and classify attacks before reporting, as well as intuitively visualize them.

2. Attack visualization

2.1. *Benefits of attack visualization*

There are significant benefits in applying information visualization to the problem of intrusion detection. First, attack visualization can easily deal with highly heterogeneous and noisy data. Network traffic can be complex, so for effective analysis it must be correlated with several variables such as source address, destination address, port number, packet length, and TCP flag among others, but the attack visualization simplifies the problem by presenting the traffic situation in an intuitive way. Even in the absence of complex mathematical or statistical algorithms, visual images can give perceptual clues to the administrators faced with an attack. Second, attack visualization can get us fresh insight into the analyzed data and allow us to deduce new hypotheses that are often lost in complex analysis. Even though an unknown attack may have occurred, if an image pattern (signature) from the unknown attack is obtained, the attack can be quickly detected. Consequently, visualization techniques can provide clues to most zero-day attacks which do not match the signatures of existing intrusion detection systems, and facilitate quick response analysis. Third, attack visualization can be much faster than other anomaly detection approaches. Many anomaly detection methods require training and comparing with history, but attack visualization can quickly identify an attack by noticing pre-defined or new image patterns.

2.2. *Attack characteristics*

In order to devise a visual mechanism for most popular Internet attacks such as DDoS attacks, worm attacks, or network scans, their characteristics must be considered in terms of visualization. Fortunately, these notorious attacks have one common characteristic, which we call ''one-to-many relationship'' between attackers and victims, whereas legitimate flows have one-to-one relationship.

A DoS attack is an attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and/or services by consuming the bandwidth of the targeted network or overloading the computational resources of the targeted system. In a DDoS attack, the attacking hosts are often personal computers with broadband connections to the Internet that have been compromised by viruses or Trojan horse programs. The perpetrator can remotely control the machines and direct the attack. With enough slave hosts, the services of even the largest and most well connected website can be denied. Therefore, in a DDoS attack, there are many attackers and one victim, which forms a one-to-many relationship between a victim (destination) and the attackers (source).

A worm is defined as a self-propagating malicious code. Once a machine is infected, target hosts are picked by pseudo random number generators, or hostscans to detect vulnerable machines (usually with a single vulnerability of the machine) in a certain network. Once targets are set, an infected machine transports the worm code to them. Therefore, a worm represents a one-to-many relationship between the infected machine (source) and the next targets (destination).

Network scanning (hostscans which are executed by several machines of network, not only a machine), used by hackers to probe hosts, also exhibits the symptoms similar to that of worm propagation, and has a one-to-many relationship between a hacker (source) and scanned network hosts (destination). Port scanning is a method used for probing available services of a certain host. There is one attacker, one or many target hosts, and many scanned ports. Therefore, a portscan is also characterized as a one-to-many relationship.

2.3. *Four fields as attack parameters*

Above, we have discussed an important characteristic of Internet attacks, namely, the one-to-many relationship. Now, let us consider which packet header variables manifest the characteristic. First, the source IP address and destination IP address in a flow information are selected as parameters because they specify the attacker and the victim host.

These values are stored in the fields of every packet header so that they can be used to distinguish the attacking packets

from legitimate packets and represent one-to-many relations between attacker(s) and victim(s).

Second, if the attack is an Internet worm, it usually targets one or more ports in TCP or UDP protocols so that the destination port number is selected as a parameter. This value identifies the targeted service of an attack and verifies port scanning attacks.

Third, the average size of packets in a flow can be used as a parameter that gives some clues whether the flow is suspicious or not. Network scannings and DDoS attacks exploit a flooding procedure and the procedure typically uses empty packets without payload. Even in case the packets have payloads, usually the length of packets are fixed — 40 or 48 bytes. And Internet worms have a payload to exploit the vulnerabilities they can use. Most worms propagate with a constant payload, the average packet size of a worm can be specified by a fixed length (Akritidis et al., 2005). Without using the packet length, we cannot distinguish Internet worms from scanning attacks. Comparing with the previous work (Kim et al., 2004), we can detect two more attack types (total six) including Internet worms and increase the correctness of decision by adding the length parameter.

Finally, the TCP flags in TCP headers and the protocol field in IP headers can be a candidate of another parameter. Networks scanning and DDoS attacks may transmit the same packet repeatedly so that these attack traffic has the same value of TCP flags. Thus, we can distinguish attack traffic from normal traffic using its TCP flags. For instance, some normal traffic has a one-to-many relationship, such as P2P communications, but they can be classified as legitimate traffic by comparing their TCP flag with that of a normal TCP handshaked flow. As well, the protocol field in IP headers can be an effective parameter. The protocol information enables us to distinguish among TCP scans, UDP scans, and ICMP scans. Furthermore, we can distinguish between TCP worms and UDP worms, and between SYN flooding attacks and UDP flooding attacks. The TCP flags and the protocol information can be considered as additional parameters, but we do not visualize them in this study. Because they do not provide any evidence of enough benefits while using them along with other four parameters.

2.4. *Per-flow visualization*

A ''flow'' can be defined as a set of packets with the same source IP, destination IP, source port and destination port that can be thought of as a connection between two remote processes. Available information on each flow includes the above mentioned attack parameters which are source address, destination address, destination port, average packet length, and TCP flags.

We use a flow, instead of a packet, as a basic unit of visualization because it drastically reduces processing time without much loss of necessary information. Furthermore, per-flow visualization provides the compatibility with legacy routers so that we can deploy the system without the change of current network infrastructure. One good example is to run a visualization system with Cisco routers that export NetFlow information (Cisco systems, 2006). Namely, the visualization system can use Cisco routers as sensors. Some parameters such as average packet length, cumulative OR of TCP flags, number of packets in a flow, number of bytes in a flow and flows per second are readily provided in such a flow-based system.

The IETF proposed a IPFIX (IP Flow Information eXport) standard (Quittek et al., 2004), which is being developed based on the experiences with Cisco NetFlow. Hence, routers and switches having the functionality of IPFIX can be used for sensing a network in our approach.

2.5. *Parallel coordinates*

One important aspect of information visualization is scalability. Parallel coordinates provide great scalability to multiple dimensions. They are not complex, yet allow hyperdimensional patterns to be analyzed. For instance, they lead to a quicker understanding and a more informational graph over that of a scattered plot matrix (The University of Utah). This technique has no theoretical limit in the number of parameters that can be visualized. Therefore, we can scale up the application by incrementally introducing new visualization parameters as necessary. Moreover, it does not introduce bias for any specific dimension, while showing prominent trends, correlations and divergences from the raw data. These advantages enable us to gain critical insight into the flows in the traffic under test and establish reliable intuitive hypotheses. Even if an unknown attack occurs, a specific image pattern can be gained and the attack can be detected in a timely manner.

3. Parallel coordinates attack visualization

3.1. *Attack signatures*

Now we show how parallel coordinates can be used to describe an attack in an intuitive graphical pattern, in our system called the Parallel Coordinate Attack Visualization (PCAV). The coordinates represent four different parameters in a flow. The first represents the source address, the second, the destination address, the third, the destination port and the fourth, the average packet length. These four values enable the flow to be plotted as a connected line on parallel coordinates.

Suppose an attacker wants to know which hosts are vulnerable in the target network. He will mount the host scanning attacks to check the targeted destination port of each host. Usually scan packets have no payload to increase the effectiveness of the scan process, resulting in 40 bytes packet. Some scanning programs also use the TCP selective acknowledgment (SACK) option, in which case the size of scanning packets becomes 48 bytes. The formation on the parallel coordinates corresponding to the host scans looks like a fish (diamond-line pattern) as shown in Fig. 1(c), which is obtained from a real-life Internet attack traffic trace.

Like this, we can define nine graphical signatures, which are shown in Table 1. In a portscan, there is one attacker and one victim, and the attacker wants to know which ports are opened. To accomplish this, the attacker may use a port scanning program which checks the destination port of the

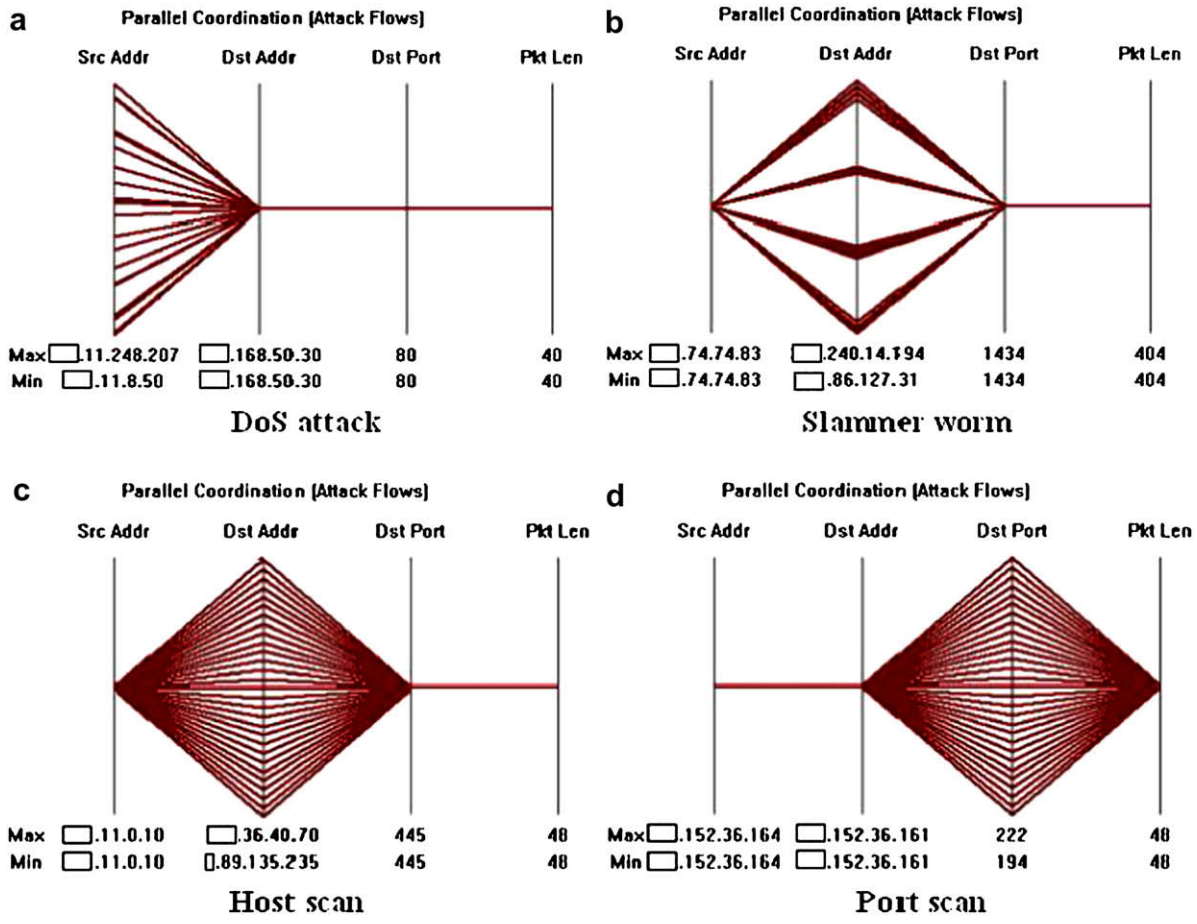

Fig. 1 – Rescaled attack graphs.

victim one by one, sequentially or randomly. This represents 1:1:many:1 patterns and a graph pattern looks similar to a kite (line-diamond) as shown in Table 1. All real-life attack graphs in Fig. 1 closely match the model signatures shown in Table 1. The DoS attack generated by a Blaster worm (Fig. 1(a)) uses a fixed destination port so the pattern resembles a triangle with a connected line. Fig. 1(b) is the attack graph from a Slammer worm, presenting a noticeable pattern. The Slammer worm attempts to infect other machines chosen randomly so that the destination addresses should have a random distribution. However, the pattern of the represented graph looks like a subnet scanning in the range of multicast IP addresses. This is due to a bug at the part of random number generation in the Slammer code (Moore et al., 2003), which generates only multicast address ranges in a certain condition. Even in this unusual situation, PCAV detects the worm on the limited range of destination addresses.

Average packet lengths can be used to distinguish seemingly similar attack patterns. For instance, a worm and a hostscan have the same graph patterns (diamond-line). But a hostscan may have no payload, whereas a worm should have a payload to infect other machines. Thus, the average packet lengths of all flows in a worm epidemic are constant and relatively larger than the average packet length of a flow in a hostscan, *e.g.*, 48 bytes in Fig. 1(c).

Backscatter packets are a non-colocated victim's response to several spoofed or non-spoofed DDoS attacks. The backscatter packets' source address is that of the victim, but the packet's destination address is randomly spoofed or non-spoofed addresses of DDoS attackers. Nonetheless, backscatter can be used to detect its matched attack, so we added it to the list of graphical signatures. When a DoS attack uses multiple source ports, its reflection has multiple destination ports. Thus, the pattern of backscatter looks like hexagon as shown in Table 1. A source-spoofed DoS (port fixed) is an attack with a fixed destination port. Usually the DoS has no payload so the graph appears as a triangle with a connected line. And, a source-spoofed DoS (port varied) is an attack with randomly chosen destination ports and usually has no payload of packets. Therefore, the graph looks like a rightward looking fish. A distributed hostscan is performed by a single perpetrator, but it is launched from multiple hosts to speed up the scanning process. So there are multiple source and destination hosts, where a particular (vulnerable) destination port is targeted. Network-directed DoS is a kind of DDoS attack that targets a network, so the union of its destination addresses will correspond to a network, instead of a single IP address. It can be regarded as a collection of multiple DDoS attacks.

We show that PCAV detects six different attacks including portscans, hostscans, Internet worms, source-spoofed DoS

Table 1 – Graphical signatures of nine attacks.

Implied Attack	Signature	Divergences
Portscan		1:1:*m*:1
Hostscan		1:*m*:1:1
Worm		1:*m*:1:1
Source-spoofed DoS (port fixed)		*m*:1:1:1
Backscatter		1:*m*:*m*:1
Source-spoofed DoS (port varied)		*m*:1:*m*:1
Distributed hostscan		*m*:*m*:1:1
Network-directed DoS		*m*:*m*:*m*:1
Single-source DoS		1:1:1:1

attacks (port fixed), backscatters, and source-spoofed DoS attacks (port varied). Since distributed hostscans and network-directed DoS attacks can be regarded as multiple hostscans and multiple source-spoofed DoS (port varied) attacks, respectively, PCAV actually detects all graphical patterns except single-source DoS attacks.

3.2. *System design*

In order to display and detect ongoing attacks using the attack signatures, we have designed a system of PCAV. The PCAV consists of four main modules: a sensor, analyzer, visualizer and database, as shown in Fig. 2. The sensor receives flow data from routers or application programs and stores the flow data. It abstracts important information from the gathered flow data and creates a PCAVflow. The PCAVflow is a compact data format with essential elements of accumulated flow data. It is designed for effectiveness of internal data stream which can unexpectedly overwhelm the computer resources. The analyzer receives the PCAVflow from the sensor, and checks a pattern, matching an attack signature, of Fig. 3. If a set of flows matches an attack signature, the attack data is sent to the visualizer. Three pre-allocated buffers are used for exchanging data between the sensor, the analyzer and the visualizer in order to minimize memory allocation and copy operations.

The visualizer displays the PCAVflow data using parallel coordinates. The flow data for both legitimate and attack flows originally comes from the sensor, from which the analyzer

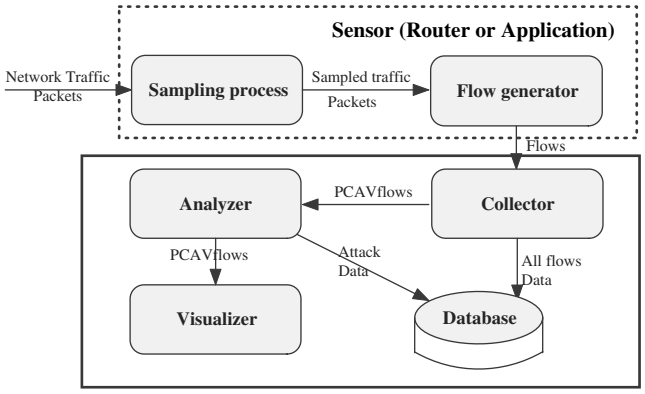

Fig. 2 – System design of PCAV.

pans out attack flows. We can store the attack data in an attack logs database which can be used for further investigation of the incident or for replaying. The system can receive flow data from hosts that run a monitoring program such as nProbe (nTop.org), or routers which enable to generate flow data such as NetFlow in Cisco routers.

3.3. *Attack detection algorithm*

In this subsection, we explain an attack detection algorithm which runs in the analyzer. The detection algorithm uses three hash tables for storing flows with respect to their source address, destination address and destination port, respectively. The hash tables are used to determine the pattern in the attack signatures.

There are several data structures that PCAV can use and a comparison of hash tables with other structures such as linked lists, balanced binary trees, and MULTOPS trees, is shown in Table 2. In particular, the MULTOPS tree is known to store IP addresses efficiently (Gil and Poletto, 2001). But it is hard to be used for storing IP addresses overwhelmed by a source spoofing DoS attack. From the comparison, we chose hash table because it provide fast lookups with sufficient accuracy.

Fig. 4 shows the proposed attack detection algorithm. The algorithm consists of two parts. The first part generates an attack ID for an input flow using three hash tables, which is described in Steps 1–12. The second part handles suspicious flows using the attack ID, which is described in Steps 13–30. The hash_insert() function in Fig. 4 inserts the parameters of the function into a hash table by the use of hash operations. The function returns TRUE if a parameter is a new member of the table. Otherwise it returns FALSE. An attack ID is a 3-tuple of binary values, where each binary element represents whether or not the PCAV hash has seen the given source address, destination address, and the destination port, respectively. A legitimate attack ID is either <0, 0, 0> or <1, 1, 1>. Otherwise, the flow is considered suspicious.

If one flow comes from the sensor, the detection algorithm in the analyzer inserts the source address, destination address and destination port of the flow to the corresponding hash table. Then, the three hash tables are used to generate the values of the flow ID. If a value already exists in its hash table,

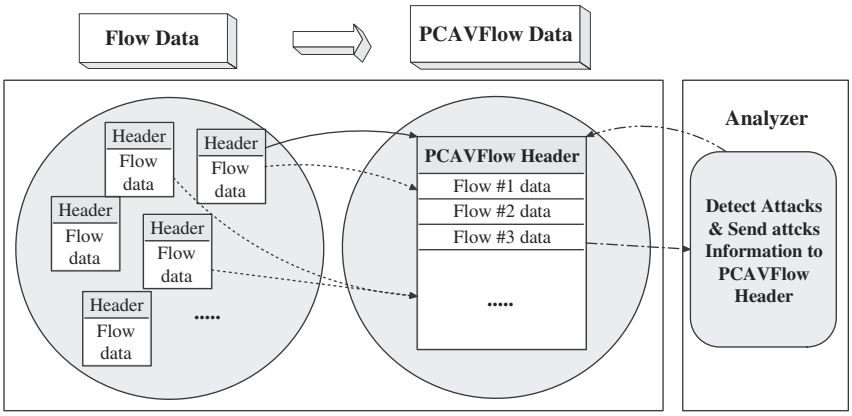

Fig. 3 – PCAVflow data generated from sensors.

then the tuple value becomes 1; otherwise, 0. For example, at time *T*1, if an input flow has a source IP address 1.2.3.4, destination address 5.6.7.8 and destination port 80, and at time *T*2, an input flow has a source IP address 1.2.3.4, destination address 5.5.5.5 and destination port 21, then the second flow at *T*2 has an attack ID of <1, 0, 0>. If at time *T*3, an input flow has source IP address 3.4.5.6, destination address 5.6.7.8 and destination port 80, then its attack ID becomes <0, 1, 1>.

In addition to the hashes used in the attack ID generation, there are 6 additional hashes corresponding to each detectable attack. Once the input flow is classified into a suspicious flow in the first phase and the length condition is satisfied then the suspicious flow is inserted into an attack hash corresponding to the attack ID. The length condition is that the scanning flows have smaller than or equal to 60 bytes, because the packets have only header. In case of TCP, the length becomes 40 bytes or 48 bytes. In case of UDP and ICMP, the length becomes 28 bytes and 60 bytes, respectively. So the maximum value of the length that contains only the header becomes 60 bytes, so we take the length of 60 bytes as a decision point.

4. Evaluation

In this section, we evaluate the performance of the PCAV algorithm on various aspects. First, we measure the processing rate of PCAV. Second, we checked the detection rate under multiple attacks. Both are instrumented with a flow generator that we implemented. Finally, we measure the false alarm rate. We run the algorithm over a real-life Internet backbone trace we have, and then over a live traffic on a fast campus network to test the feasibility of attack visualization.

We use the Pentium 4 processor PCs on Windows XP and default values of parameters are given as follows. Hash entry lifetimes are 2 s for IP-address hashes, and 1 s for a port hash.

Table 2 – Comparisons of data structures.

Algorithm	Pros	Cons	Complexity
Linked List	Small memory usage	High lookup complexity	$O(n^2)$
MULTOPS's tree data structure	O(1) lookup complexity	Weak to source-spoofed DDoS attack	O(1)
Binary search tree (Balanced)	Small memory usage	High lookup complexity	$O(n\log n)$
Hash table	O(1) lookup complexity	High memory usage, Hash collision	O(1)

```
Attack-Detection (F_n)   /*F_n n th input flow data*/
1   SA_h, DA_h, DP_n, AvgLen_n /*Src IP, Dst IP, Dst port, and Average Pkt length of F_n*/
2   T_s, T_d, T_p  /* Hash tables for SA_n, DA_n, DP_n*/
3   Attack_ID= 0x0111  /*Initialize Attack_ID*/
4   IF hash_insert({SA_h },T_s )= TRUE  /*If SA_n  is a new value of hash table T */
5      Attack_ID= Attack_ID  XOR  0x0100
6   ENDIF
7   IF hash_insert({DA_h },T_d )= TRUE
8      Attack_ID= Attack_ID  XOR  0x0010
9   ENDIF
10  IF hash_insert({DP_n },T_p )= TRUE
11     Attack_ID= Attack_ID  XOR  0x0001
12  ENDIF
13  DDoS_fix, DDoS_var, Hostscan, Worm, Portscan, Backscatter /*Attack hash tables*/
14  IF Attack_ID= 0x0011 and AvgLen_n < MAX_HLEN /*Max header length*/
15     hash_insert({DA_h, DP_n, AvgLen_n },DDoS_fix )
16  ENDIF
17  IF Attack_ID= 0x0010 and AvgLen_n < MAX_HLEN
18     hash_insert({DA_h, AvgLen_h },DDoS_var )
19  ENDIF
20  IF Attack_ID= 0x0101 and AvgLen_n < MAX_HLEN
21     hash_insert({SA_h, DP_n, AvgLen_n },Hostscan)
22  ELSE
23     hash_insert({SA_h, DP_n, AvgLen_n },Worm)
24  ENDIF
25  IF Attack_ID= 0x0110 and AvgLen_n < MAX_HLEN
26     hash_insert({SA_h, DA_h, AvgLen_n },Portscan)
27  ENDIF
28  IF Attack_ID= 0x0100 and AvgLen_n < MAX_HLEN
29     hash_insert({SA_h, AvgLen_h },Backscatter)
30  ENDIF
END of Attack-Detection
```

Fig. 4 – Attack detection algorithm in the analyzer.

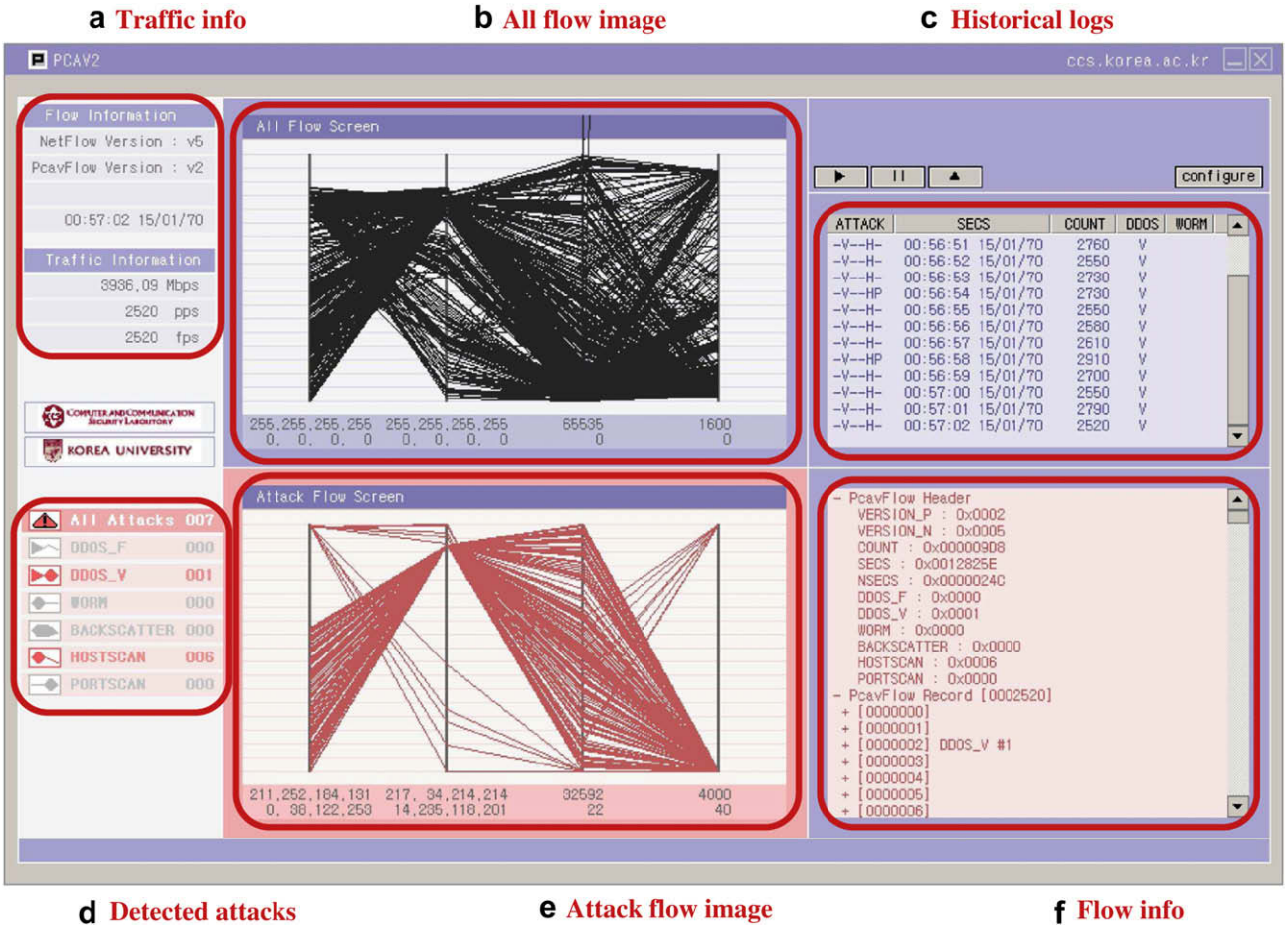

Fig. 5 – Screenshot of the PCAV application running on backbone traffic.

There is a gap between two lifetimes, because the range of IP addresses is much larger than the range of ports. The threshold for determining DDoS or worm attacks is 50 (flows per second). For scanning attacks, smaller threshold, *i.e.*, 5 flows per second, is used since the typical number of flows in scanning attacks is smaller than those of DDoS or worm attacks. Detecting algorithm runs every second to detect and report any suspicious activity, but the report period is adjustable.

PCAV is implemented as an application program,[1] where a screenshot is presented in Fig. 5. The upper graph in Fig. 5, namely (b) All flow image, shows entire flows on parallel coordinates. The pattern of polygonal lines looks crossed regularly due to the pairs of outward and inward flows. PCAV can show the ongoing attacks in both absolute coordinates and rescaled coordinates. In order to present the detected attack more effectively, PCAV provides the rescaling functionality to magnify an attack graph, which appears in the lower graph, (e) Attack flow image on the PCAV application.

PCAV can detect and visualize different types of attack, even they occur simultaneously. In Fig. 5, there are (d) Detected attacks buttons which shows the appearance of multiple attacks in different types. When an attack is detected, its corresponding button is activated. All attacks detected are displayed in (e) Attack flow image; whereas only one attack type selected in (d) Detected attacks buttons is displayed on (e) Attack flow image when clicking a button.

Even though an unknown pattern of image appeared in (b) All flow image, we can get the detail information of the attack such as IP address, protocol, the number of packets, the octets of bytes and port number in (f) Flow information. The detail information provides the first step towards the inspection of the unknown attacks, and we can add new types of attack for detecting more attacks.

4.1. *Case study*

We evaluated the PCAV with the traces from 3 different sources: a local network, an ISP backbone and public trace repositories.

First for the local trace, system was plugged to a campus network for over 48 h on March 13, 2006. The campus gateway routers exported NetFlow data to the PCAV. The campus traffic was about 1 Gbps with 1000 flows per second. In this experiment, the PCAV system was observed to process the NetFlow data from the giga-bit campus network without any loss. During the expanse of the one-day trace, 1139 attacks

[1] The application program of PCAV can be obtainable at http://ccs.korea.ac.kr/PCAV.

were detected. Among them, 1060 attacks were portscan (93% attacks) and 73 attacks were hostscan (6% attacks). 5 worm attacks were reported (0.44%). Although most attacks were detected correctly, the reported worms turned out to be messenger spams and P2P traffic. Therefore, the false positive rate is smaller than 0.44%. We discuss the false alarms issue in Section 4 D.

Secondly, we ran PCAV on a backbone traffic captured during 13 h in December 14, 2001 on two trans-pacific T3 links (90 Mbps) connecting the U.S. and a Korean Internet Exchange. PCAV successfully reported the attacks embedded in the trace. For instance, bandwidth consuming attacks, such as DDoS attacks, were clearly visible as shown in Fig. 5. PCAV detect 4587 attacks and among them, 1206 attacks were portscan (26% attacks) and 3223 attacks were hostscan (70% attacks). 6 DDoS attacks (0.13%) and 74 worms were reported (1.6%). We also discuss the false alarms issue in Section 4 D.

Thirdly, we ran the PCAV with backscatter traffic traces, which were provided by the Cooperative Association for Internet Data Analysis (CAIDA; http://www.caida.org). The dataset consisted of 7 week-long collections of responses to spoofed traffic sent by DoS victims and received by the UCSD Network Telescope between May 2004 and November 2005. PCAV detected all backscatters in the CAIDA traffics, which will be discussed in Section 4 E.

4.2. *Stress test*

We performed a stress test to estimate the stability of the system. In order to measure the maximum number of flows that can be processed, we defined the rate of processed flows over all input flows. Attack flows were launched with the flow generator that we implemented, which is able to inject several types of attack flows as well as normal flows. Also, we extended the socket buffer from 8 KB to 1 MB to avoid packet dropping and turned off the PCAV database system, in order to eliminate the effect of disk access latency from the measurement.

Fig. 6 shows the processing rate of the system. PCAV processes 90% of 100,000 flows given per second, *i.e.*, 90,000 flows per second, which could roughly match the traffic intensity on a 10 Gbps pipe based on our experience. Since the attack detection thresholds are a few orders of magnitudes smaller, the processed flows are more than enough to signal the attacks if any. As to the flow missing rate, PCAV gracefully degrades, rather than sharply drops after a certain point of flow input rate.

We can estimate the maximum detectable number of attacks numerically. Suppose there are m attacks at time x. Let τ be a time interval, and $a_i(x)$ be the total number of flows generated by attack i from time $x-\tau$ to x. We let $A(x)$ denote the total number of attack flows such that

$$A(x) = \sum_{i=1}^{m} a_i(x).$$

If $N(x)$ is the number of legitimate flows, the total number of flows $T(x)$, becomes $T(x) = A(x) + N(x)$. We let ρ denote the maximum number of flows that can be processed by the PCAV analyzer during τ, and λ be the threshold of an attack to be

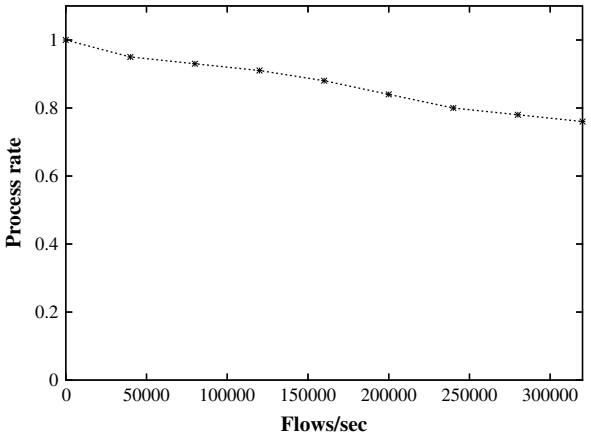

Fig. 6 – Processing rate of PCAV.

detected. Then the following conditions are satisfied. First, when $T(x) < \rho$ and $a_i(x) \geq \lambda$ for all i, every attack is detectable. We call it m-detectable. In case $\lim_{x\to\infty} N(x) = 0$, m is bounded from above as

$$m = \frac{T(x)}{\lambda}.$$

Second, if $T(x) > \rho$ and $a_i(x)\cdot(1 - T(x) - \rho/T(x)) \geq \lambda$ $(1 \leq i \leq m)$, it is also m-detectable. In case $\lim_{x\to\infty} N(x) = 0$, we have

$$m = \frac{\rho}{\lambda}.$$

For instance, given 10,000 flows of which 90% are processed by PCAV with the threshold of $\lambda = 50$, 1800 concurrent attacks are detectable. From this simple analysis, we can ensure that PCAV can detect attacks effectively, even when input flows are dropped significantly under congestion by DDoS or worm attacks. It also works consistently when input flows are sampled by routers or applications.

4.3. *Multiple attacks test*

Even though multiple attacks occur simultaneously, PCAV should work properly. One weak point of PCAV can be the hash collision under multiple attacks. The hash collision can make the PCAV analyzer generate a wrong flow ID and some attacks can be undetected. So, we evaluate the effectiveness of PCAV under multiple attacks.

Fig. 7 shows the detection rate of multiple attacks versus the number of concurrent attacks. The experiment is conducted with the aforementioned flow generator. The figure shows that the attacks with larger divergence on IP-address coordinates, *i.e.*, hostscan, worm and port fixed DDoS, are more detectable than the port-related attacks where the port numbers vary. It is because the range of IP addresses is wider than that of ports (65 K) so that hash collisions rarely occur in the IP-related attacks. Port-related attacks, *i.e.*, portscan, port varied DDoS and backscatter, are less detectable than IP-related attacks, since the collision probability of destination ports is much higher than that of IP addresses.

If a portscan interferes with a DDoS attack with varied ports, the DDoS attack can be incorrectly recognized as a DDoS

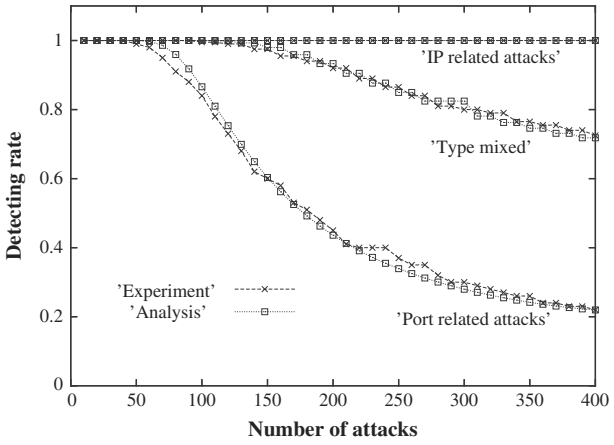

Fig. 7 – Detection ratio of multiple attacks.

attack with fixed ports. However, in a DDoS attack, it does not matter whether the attack uses fixed ports or varied ports. Thus, the interference of portscans with DDoS attacks can be considered negligible, which is represented by the dotted line in Fig. 8. Only the DDoS attack with varied port interferes with portscans significantly, which is represented by the solid line in Fig. 8.

Fig. 7 also shows that IP-related attacks do not interfere with port-related attacks, and *vice versa*. Only the same type of attacks interfere with each other. Moreover, notice that the detection ratio of mixed type attack is higher than the simple average of the detection ratios of single type attacks. In the figure, we mix the IP-related and port-related attacks half and half. At $m = 400$, we see that the detection ratio with the mixed attack is 0.7, which is larger than the average of the detection ratios of the single type attacks, *i.e.*, 0.6. Approximately, if the fraction of port-related attacks is α_p, the detection ratio of m mixed attacks is given by

$$D_x(m) \approx D_I(m \cdot (1 - \alpha_p)) + D_p(m\alpha_p)$$

where D_I and D_p are detection ratios with IP- and port-related attacks. The precise detection ratio gain in the mixed attacks depends on the collision probability function of the hashing scheme. Approximating the distribution of the number of hash collisions as Gaussian, Fig. 7 shows that the analytical estimation fits well with the experimental results.

We can measure the collision probability numerically. Suppose there are m concurrent IP-related attacks with scan rate r (scans or spoofs) per second and the lifetime of an attack is L. Then the number of host address entries would be mLr. When an attack randomly picks an address for the next address, the probability of picking up an already registered value will be estimated as $(mLr)/(2^{32})$. So the collision probability is given by

$$P_c(m) = \frac{mLr}{|S|},$$

where S is the whole range of values, and the upper limit of the probability can be bounded by 1 such that

$$P_c(m) = \begin{cases} P_c(m) & \text{if } P_c(m) \leq 1 \\ 1 & \text{otherwise} \end{cases}$$

For example, if there are 1000 attacks that generate 10,000 packets per second each and $L = 10$, the collision probability is approximately 1/400. It seems tolerably low, for such a large number of highly intensive attacks.

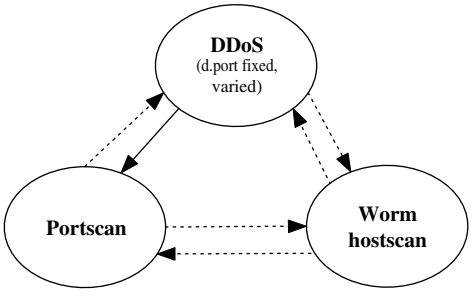

Fig. 8 – Interference relationship between attacks.

4.4. *False alarms*

The false alarm rate is an important metric to measure the performance of an intrusion detection system. We measure the false positive with sources from a local network, and an ISP backbone, which were smaller than 0.44% and 2.31%, respectively. It is shown that PCAV greatly reduces the false detections by using additional parameters. For example, let us consider a popular website. Numerous clients can access a popular server at TCP port 80. Without the use of length parameter, it is possible to report the traffic incoming to the server as a DDoS attack with fixed port, or the outgoing traffic as a hostscan. Not only the flash crowds in web traffic, but also game, chatting, mail, P2P, streaming can be misjudged as worms unless the length parameter is used. Thus PCAV utilizes the packet length parameter. We compare two cases: three parameters and four parameters. The difference between two cases comes from the use of the average packet length in a flow as the fourth parameter. The impact of the use of the length parameter is shown in Fig. 9, and it clearly shows that we can greatly reduce false alarms with one additional parameter. This shows the immediate improvements of PCAV with respect to accuracy, comparing with the previous work (Kim et al., 2004) which uses only three parameters.

Although the packet length significantly reduces the chances of false alarm, some of detected attacks were false

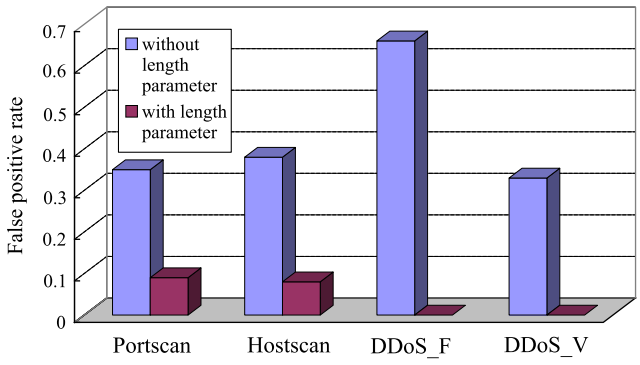

Fig. 9 – False alarm reduction by the use of the length parameter.

positives. For instance in our experiment with the live campus network traffic, several attacks were detected as a worm. However, we found that they are not worms but messenger spams or P2P traffics. P2P applications such as "FileGuri" use UDP packets for the file searching query. In that case, the P2P client will send the UDP packets with the same payload to multiple nodes. Then the flows of the UDP query look like a worm or a hostscan, as shown in Fig. 10(a). Also, when a P2P node disconnects from the P2P overlay, the other participants can query the node, not knowing the disconnection immediately. These queries will induce ICMP port unreachable messages. The ICMP messages can be considered as the pattern of hostcan, which is shown in Fig. 10(b).

To our surprise, many messenger spams were also detected as a worm as shown in Fig. 10(c). Messenger spams send an unsolicited message using the Microsoft Windows messenger service. Typically the messenger service runs on the UDP port 1026, but it can run on other ports. Spammers can send a message to a range of ports (typically 1026–1033), to accomplish the delivery with higher probability. Note that IP spoofing with UDP protocols is easy to perform but difficult to trace back to the spam transmitter. We can distinguish them by the use of well-known destination ports as a "white list". Furthermore, false alarms can be reduced by the use of additional parameters such as the cumulative OR of TCP flags. Additionally, more parameters can be added to parallel coordinates if they can enhance the correctness of attack detection. However, even though PCAV can detect attacks with an increased accuracy using more than four parameters, the space and time complexity increase can offset the benefit. So PCAV leaves open the possibility of using additional parameters, but in the current design, we use four.

4.5. *Reflection of attack*

Attack packets can induce the replies from victim(s) and the reply packets can be seen as another type of attack. We call this mutation the reflection. For example, if one attacker performs the hostscan, then victims reply, which would be seen as the packets of DDoS attack that is shown in Fig. 11(a). We list some such notable reflections in Table 3. Here, a subscript O (or R) means the number of original (or reflection) flows of an attack. For instance, Portscan_O is the number of attack flows used in a portscan attack, and Portscan_R is the number of reflection flows caused by the portscan attack. We maintain the view that a reflection is not an attack in itself, but an evidence of attack. The portscan-reflecting portscan in Fig. 11(b) and DDoS-reflecting backscatter are okay because they are already contained in the attack signatures. Moreover, under symmetric routing, the DoS traffic converging on the backscatter source (*i.e.*, DoS victim) should be detected. If only backscatter is noticed, then the DoS attack must be utilizing another path. Then backscatter can be leveraged to hint at the existence of the attack in other parts of the network. Fig. 11(c)

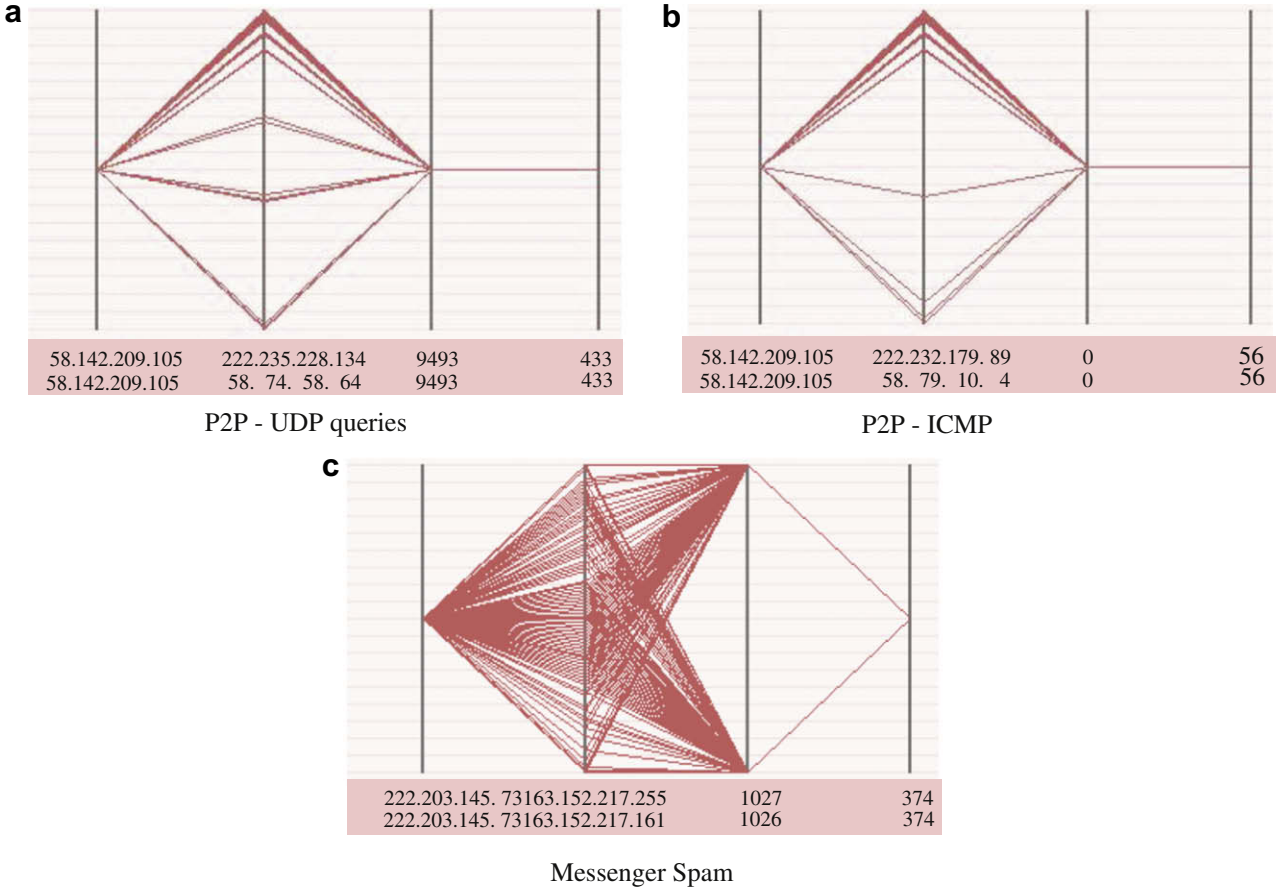

Fig. 10 – False detection of worms: (a) UDP queries in P2P applications, (b) ICMP port unreachable messages in P2P applications, (c) Messenger spams.

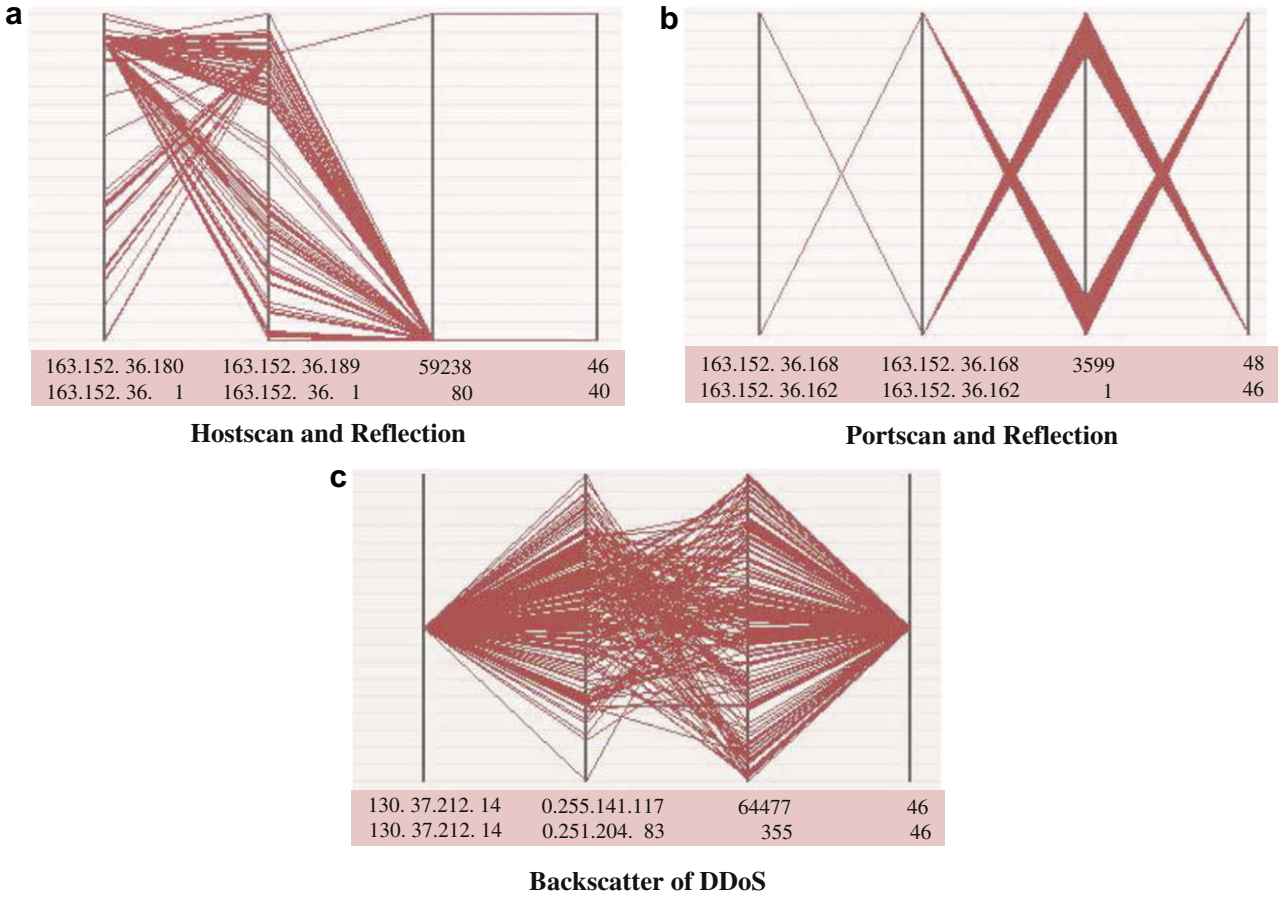

Fig. 11 – Reflections of hostscan, portscan and DDoS attacks.

shows the backscatter graph generated from the backscatter trace of CAIDA.

There could be two ways to discriminate the reflection. First, we could check if there is the inverse image. If there is a DDoS attack targeted at the apparent host scanning source and the number of DDoS flow is much smaller than original hostscan, it is likely that the DDoS is a reflection. Second, the hostscan as a reflection can be dominantly showing the characteristics of the responding traffic, such as TCP SYN/ACK or ICMP Echo Reply. The worm would be reflected to a DDoS attack, but the number of flows is much smaller. Therefore, we can determine that the reported DDoS attack is the reflection of a Internet worm.

Table 3 – Some notable reflections.

Original	Condition	Reflection	Note
Portscan	Src port is fixed	No	$\text{Portscan}_O > \text{Portscan}_R$
	Src port is not fixed	Portscan	
Hostscan	Src port is fixed	DDoS_F	$\text{Hostscan}_O > \text{DDoS}_R$
	Src port is not fixed	DDoS_V	
DDoS_F	Src port is fixed	Hostscan	$\text{DDoS}_O > \text{Hostscan}_R$ (If DDoS use source spoofing then, $\text{DDoS}_O \gg \text{Backscatter}_R$)
DDoS_V	Src port is not fixed	Backscatter	$\text{DDoS}_O > \text{Backscatter}_R$, (If DDoS use source spoofing then, $\text{DDoS}_O \gg \text{Backscatter}_R$)
Worm	TCP worms	DDoS_V	$\text{Worm}_O > \text{DDoS\_V}_R$
	UDP worms	DDoS_F	$\text{Worm}_O > \text{DDoS\_V}_R$ (ICMP packets)

5. Related work

Many visualization approaches have been proposed to display complex data in networks in order to enable humans to recognize abnormal network status intuitively (Lakkaraju et al., 2004; Nyarko et al., 2002; Kim and Reddy, 2005; Samak et al., 2008; Onut and Ghorbani, 2007; Fischer et al., 2008; Kim et al., 2004; Solka et al., 2000; Axelsson, 2003; Yin et al., 2004). Among them, SHADOW (Solka et al., 2000), BLINC (Karagiannis et al., 2005), Spotfire (Axelsson, 2003), and VisFlowConnect (Yin et al., 2004) use parallel coordinates to show abnormal behaviors.

NVisionIP (Lakkaraju et al., 2004) is a visual tool intended to improve the cognitive processing abilities of human operators. Combining the visualization and reasoning capabilities of humans, NVisionIP allows to recognize ongoing attacks on

a network. However, it only provides visual images but not algorithms for detecting attacks in a systematic way.

Network intrusion can be visualized by NIVA (Nyarko et al., 2002), which is an intrusion detection visual analyzer with haptic integration. The visual tool is able to augment the ability of network administrators to understand malicious activities on a network.

NetViewer (Kim and Reddy, 2005) provides the view of network traffic as a sequence of visual images so that various image processing techniques can be applied for detecting and visualizing attacks. NetViewer detect attacks by the use of image processing technologies.

Space-filling curves (SFCs) (Samak et al., 2008) characterize traffic flows and identify anomalous behavior. SFCs generates traffic images that provide both storage and bandwidth savings. The technique is evaluated with actual traces including DDoS attack and Code Red spread traffic and the result images are shown to withstand aggressive compression while preserving traffic properties.

SVision is proposed as a visualization technique for intrusion detection systems (Onut and Ghorbani, 2007). A network can be represented as a community of hosts roaming in a 3D space which is defined by the set of services. Since a network might have hundreds of hosts, the view in SVision highlights only the ones that might represent a potential threat to the network. The approach used in SVision is useful for giving an intuition for determining an attack, but it cannot deal with large-scale attacks effectively on a high-speed network.

NFlowVis (Fischer et al., 2008) analyzes NetFlow data using a relational database system. The monitored network is mapped to a TreeMap visualization, the attackers are arranged at the borders and linked using splines parameterized with prefix information. The tool can be used to judge the relevance of alerts, to reveal massive distributed attacks, and to analyze service usage within a network.

Attack visualization on 3-D graphs have been proposed to detect and display Internet attacks such as DDoS attacks and scanning activities (Kim et al., 2004). By the use of only three fields of IP headers, malicious attacks can be visualized instantly in a 3-dimensional space. One drawback of the scheme is the limited capability of detecting attacks due to the fixed three parameters. Internet worms cannot be detected properly and legitimate traffic patterns are falsely recognized as an attack, such as flash crowds, P2P communications, web-crawling activities and online games.

Parallel coordinates have been used in other studies such as SHADOW (Solka et al., 2000). In SHADOW, packet headers meeting pre-defined rules are dumped to a web-based file for examination by a human operator.

There is an approach (Krasser and Conti, 2005) using parallel coordinates for realtime and forensic data analysis. Combining the strength of parallel coordinate plots with the time-sequence animation of scatter plots, displaying 2D or 3D graphs provides insight into both legitimate and malicious network activities. A small set of attacks can be visually represented such as Slammer worms, botnet traffics and portscans.

Spotfire (Axelsson, 2003) is developed for exploring the possibilities of employing a trellis plot of parallel coordinate visualizations to the log of a small personal web server. The aim is to enable the operator to tell apart the access patterns of automated attacks and normal access patterns.

VisFlowConnect (Yin et al., 2004) is a visualization application to enhance the ability of an administrator to detect and investigate anomalous traffic between a local network and external domains. Central to the design is a parallel axes view which displays NetFlow records as links between two machines or domains while employing a variety of visual cues to assist the user.

As we mentioned above, there are several approaches that capture network attacks visually. However, they do not detect attacks systematically, but just show the images of attacks using their visualization method.

6. Conclusion

PCAV is a real-time visualization system for detecting anomalies from Internet attacks. PCAV visualizes Internet attacks using four header fields (source IP address, destination IP address, destination port, packet length) from a flow and displays on parallel coordinates. PCAV exploits the inherent property that each significant attack has a unique graphical pattern on parallel coordinates. PCAV enables the network administrator to rapidly detect and respond to malicious attacks.

When an unknown attack occur, a particular pattern can be displayed visually. Thus, PCAV allows us to add new signatures and their detection routines. In this way, PCAV greatly enhances the intelligence of visual systems for detecting more attacks.

A plan to adopt pattern recognition methods in computer graphics areas, recognizing signatures generated by PCAV, is currently being designed. Also, visualization research regarding spam mail distributions, P2P traffics, botnets and new types of DDoS and worm attack, is currently being undertaken.

Acknowledgement

This research was supported by the Ministry of Knowledge Economy, Korea, under the ITRC program (IITA-2008-C1090-0801-0016), IT R&D program of MKE/IITA [2008-S-026-01], the Basic Research Program of the Korea Science & Engineering Foundation, and the Defense Acquisition Program Administration and Agency for Defense Development under the contract UD060048AD.

REFERENCES

Akritidis P, Anagnostakis K, Markatos E. Efficient content-based detection of zero-day worms. In: Proc. of IEEE ICC; May 2005.

Axelsson S. Visualization for intrusion detection: Hooking the worm. In: Proc. of ESORICS; October 2003.

Cisco IOS NetFlow. Cisco Systems, Inc., http://www.cisco.com/warp/public/732/Tech/netflow; 2006.

Conti G, Abdullah K. Passive visual fingerprinting of network attack tools. In: Proc. of ACM VizSEC/DMSEC; October 2004.

Fischer F, Mansmann F, Keim DA, Pietzko S, Waldvogel M. Large-scale network monitoring for visual analysis of attacks. In: Proc. of ACM VizSEC/DMSEC; 2008.
Gil T, Poletto M. MULTOPS: a data-structure for bandwidth attack detection. In: Proc. of USENIX security symposium; August 2001.
Inselberg A. The plane with parallel coordinates. The Visual Computer 1985:69–91.
Karagiannis T, Papagiannaki K, Faloutsos M. Blinc: Multilevel traffic classification in the dark. In: Proc. of ACM SIGCOMM; 2005.
Keim DA. Visual exploration of large data sets. Communications of the ACM 2001:38–44.
Kim S, Reddy A. A study of analyzing network traffic as images in real-time. In: Proc. of IEEE INFOCOM; 2005.
Kim H, Kang I, Bahk S. Real-time visualization of network attacks on high-speed links. IEEE Network 2004.
Krasser V, Conti G. Real-time and forensic network data analysis using animated and coordinated visualization. In: Proc. of IEEE Workshop on Info. Assurance and Security; 2005.
Lakkaraju K, Yurcik W, Adam J. NVisionIP: netflow visualizations of system state for security situational awareness. In: Proc. of ACM VizSEC/DMSEC; 2004. p. 65–72.
Moore D, Shannon C, Voelker G, Savage S. Inside the slammer worm. IEEE Security and Privacy 2003:33–9.
ntop.org. nProbe: an extensible NetFlow v5/v9/IPFIX GPL probe for IPv4/v6, http://www.ntop.org/nProbe.html.
Nyarko K, Capers T, Scott C, Ladeji-Osias K. Network intrusion visualization with NIVA, an intrusion detection visual analyzer with haptic integration. In: The 10th symp. on haptic interfaces for virtual environment and teleoperator systems; 2002.
Onut I-V, Ghorbani AA. SVision: A novel visual network-anomaly identification technique. Computers and Security 2007:201–12.
Quittek J, Zseby T, Claise B, Zander S. Requirements for IP flow information export (IPFIX), vol. 3917. RFC; 2004.
Samak T, Ghanem S, Ismail MA. On the efficiency of using space-filling curves in network traffic representation. In: Proc. of IEEE INFOCOM; 2008.
Solka J, Marchette D, Wallet B. Statistical visualization methods in intrusion detection. Computing Science and Statistics 2000: 16–24.
The University of Utah. Information visualization resources, http://www.infovis.org.
Yin X, Yurcik W, Treaster M, Li Y, Lakkaraju K. VisFlowConnect: NetFlow visualizations of link relationships for security situational awarenes. In: Proc. of ACM VizSEC/DMSEC; 2004.

Hyunsang Choi received the BS, MS degrees from the Department of Computer Science and Engineering at Korea University, Seoul, Korea. He is currently working towards the PhD degree in the Division of Computer and Communication Engineering at Korea University.

Heejo Lee is an associate professor at the Division of Computer and Communication Engineering, Korea University, Seoul, Korea. Before joining Korea University, he was at AhnLab, Inc. as a CTO from 2001 to 2003. From 2000 to 2001, he was a postdoc at the Department of Computer Sciences and the security center CERIAS, Purdue University. Dr. Lee received his BS, MS, PhD degree in Computer Science and Engineering from POSTECH, Pohang, Korea. Dr. Lee serves as an editor of Journal of Communications and Networks. He has been an advisory member of Korea Information Security Agency and Korea Supreme Prosecutor's Office.

Hyogon Kim is a professor at the Korea University. He got his Ph.D. from the University of Pennsylvania in 1995. Prior to joining the Korea University, he was a research scientist at Bell Communications Research (Bellcore). His research interests include Internet protocols and applications, wireless networking, network security, and computational neuroscience.