PAPER Special Section on Towards Management for Future Networks and Services

Extended Darknet: Multi-Dimensional Internet Threat Monitoring System

Akihiro SHIMODA[†a)], *Nonmember*, **Tatsuya MORI**[††], *and* **Shigeki GOTO**[†], *Members*

SUMMARY Internet threats caused by botnets/worms are one of the most important security issues to be addressed. Darknet, also called a dark IP address space, is one of the best solutions for monitoring anomalous packets sent by malicious software. However, since darknet is deployed only on an inactive IP address space, it is an inefficient way for monitoring a working network that has a considerable number of active IP addresses. The present paper addresses this problem. We propose a scalable, lightweight malicious packet monitoring system based on a multi-dimensional IP/port analysis. Our system significantly extends the monitoring scope of darknet. In order to extend the capacity of darknet, our approach leverages the active IP address space without affecting legitimate traffic. Multi-dimensional monitoring enables the monitoring of TCP ports with firewalls enabled on each of the IP addresses. We focus on delays of TCP syn/ack responses in the traffic. We locate syn/ack delayed packets and forward them to sensors or honeypots for further analysis. We also propose a policy-based flow classification and forwarding mechanism and develop a prototype of a monitoring system that implements our proposed architecture. We deploy our system on a campus network and perform several experiments for the evaluation of our system. We verify that our system can cover 89% of the IP addresses while darknet-based monitoring only covers 46%. On our campus network, our system monitors twice as many IP addresses as darknet.

key words: *darknet, multi-dimensional monitoring, Internet threat, sensor*

1. Introduction

Internet threats caused by botnets, worms, and other malwares are one of the most important security issues to be solved urgently [1]. Referece [2] reports that the annual worldwide economic damages from malwares exceeded $13 billion in 2006. According to Ref. [3], malware samples have increased 22% over 2010, with 6 million unique malwares found in the second quarter of 2011. From the view point of network administrators, it is a demanding task to protect networks from malicious attacks. One of the common approaches to locate malicious packets is to deploy *darknet* [4], also called a dark IP address space on their network. Darknet is a collection of global IP addresses that vacuums packets from the Internet and never responds to them. Darknet intends to detect scanning packets or other anomalous packets sent to unspecified destinations. Anomaly packets are typically sent by malwares, the backscatter of distributed denial-of-service (DoS) attacks or the misconfiguration of software. Darknet monitoring logs help network administrators to identify attackers' IP addresses, targeted service ports, or the scanning behavior. These logs can be utilized for a firewall configuration or for alerting network users to take some action to deal with the security issue.

However, darknet monitoring has a disadvantage in that it requires a certain amount of unused global IP address space. Currently, the exhaustion of the IPv4 address pool is a serious problem. IANA reported the exhaustion of its address pool on January 31, 2011, and APNIC's address pool was exhausted on April 15, 2011. Therefore, it is difficult to reserve a certain amount of global IP address space for Internet monitoring. Thus, there is a demand for monitoring *active* IP address spaces.

In order to tackle these problems, we extend the concept of darknet by introducing novel flow-based monitoring. A network flow is described as a tuple that consists of four parameters: source/destination IP addresses and source/destination ports. Thus, our flow-based monitoring can also be called port-based monitoring. We introduce the idea of multi-dimensional monitoring in which each TCP port on each IP address is monitored. We isolate malicious packets from the Internet backbone traffic by monitoring TCP flags and delays of syn/ack responses. Flow-based monitoring also monitors malicious packets that are destined for active hosts, because multi-dimensional monitoring can monitor all packets, even those that are blocked by firewalls on active hosts. Thus, our method makes it possible to monitor almost the entire network address space, even when there are many active hosts. Our system also implements policy-based forwarding and classification in order to prevent adverse effects on legitimate traffic. We can exclude a specific address space for threat monitoring. Legitimate traffic is excluded from monitoring in real-time by using *forwarding table* and *delay queue* functions of our system.

The rest of this paper is organized as follows: Sect. 2 reviews related works and compares them with our work. Section 3 describes the architecture and implementation of our system. In Sect. 4, we describe our experimental environment and present empirical results. Finally, Sect. 5 concludes our work and discusses future works.

2. Related Works

The Darknet [4] project provides an overview of darknet implementation for collecting malicious packets. Darknet is

Manuscript received October 14, 2011.
Manuscript revised January 25, 2012.

[†]The authors are with Computer Science Dept., Waseda University, Tokyo, 169-8555 Japan.
[††]The author is with NTT Service Integration Laboratories, NTT Corporation, Musashino-shi, 180-8585 Japan.
a) E-mail: shimo@goto.info.waseda.ac.jp

DOI: 10.1587/transcom.E95.B.1915

also called a dark IP address space. It monitors a large network address space with one packet vacuum server. The monitored packets are definitely malicious because no legitimate packets can be sent to a darknet. Darknet is used for deploying backscatter detectors, packet sniffers, or IDS boxes. Internet Telescope [5] assigns a very large IP address space (e.g., /8, /16) for darknet which enables us to collect considerable attack logs for further analysis. This IP address space is completely dedicated to threat monitoring and never used for other Internet applications. Therefore, it can collect clean malicious packets, and there is no possibility of mixing legitimate traffic with the sensor logs. A large address space for passive monitoring is restricted to very few institutions, since obtaining a large address space is difficult because the impending IPv4 address exhaustion.

Threat monitoring systems such as NICTER [6], ISDAS [7], and WCLSCAN [8] distribute sensors on various address blocks over the Internet. The placement of distributed sensors enables the collection of widespread botnets/worms activities on various networks. However, the reservation of IP address blocks for various organizations is difficult because of the costs for placing and maintaining sensor hardware. Moreover, these systems often hide their sensor locations on the basis of their privacy policy. Hence, the threat information from these systems does not contain information about sites of attacks. Network administrators who require more detailed logs for their networks should place IDS or other monitoring systems in their networks.

Potemkin [9] is a large-scale honeypot system that enables the distribution of more than 10,000 honeypots to large address blocks. Potemkin automates the process of packet forwarding with virtual servers and an intelligent gateway that implements a highly managed session control mechanism. The implementation and configuration are fairly complex. The intelligent gateway is hard-coded, and the virtual servers consume considerable hardware resources. This system utilizes unused address blocks with static routing changes. Therefore, it is not suitable for a working network.

DarkPots [10] is a large-scale honeypot platform that enables the deployment of thousands of honeypots or sensors on a working network. This system has an advantage to utilize an isolated unused address space for monitoring, while earlier darknet should be a continuous IP address space which is unused. These unused addresses are located by two methods: a black-box inspection of a router's firewall rules, and the monitoring of IP address activities by packet capturing. This system performs passive monitoring using sensors, and active monitoring using a Linux-process-based honeypot with a sub-interface configuration. Therefore, the consumption of hardware resources is low, and the configuration is easy. However, the address coverage depends on the usage of the working network. The system should not collect many packets in networks with a few unused IP addresses. Further, a large address space, such as a/8 address block or an IPv6 address space, might increase the load of the system because the system must maintain the list of IP addresses to be monitored.

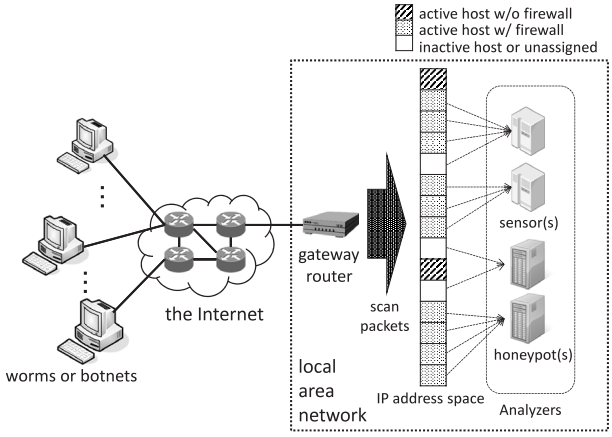

Fig. 1 Overview of our system.

3. Architecture

In this section, we provide an overview and details of our proposed architecture, including the multi-dimensional monitoring, functions of components, and policy-based forwarding. We then illustrate the implementation of our architecture.

3.1 Overview

Figure 1 illustrates an overview of the system. We focus on TCP sessions initiated by botnets/worms activities from the Internet. TCP packets are sent from the Internet to a local area network. A local area networks is typically an enterprise or a campus network inside a gateway router. The existence of a TCP syn/ack packet depends on the configuration of each network host or intermediate router. For example, an active host with a firewall that allows no incoming connection never sends a syn/ack packets against any incoming packet. In contrast, an active host with no firewall enabled should send a rst or a syn/ack packet against incoming syn packets. There is no response packet sent against incoming syn packets when the destination IP address is inactive if the host machine is turned off or the address is unassigned. Given this response behavior, we propose a flow-based malicious packet monitoring system that monitors the response behavior for each session. In addition, we propose an architecture that combines sensors or honeypots with an intelligent flow forwarding mechanism. In the rest of this section, we describe the details of flow-based monitoring, which we call multi-dimensional monitoring. We also describe our architecture, including policy classification, and integration with a programmable flow switch.

3.2 Multi-Dimensional Monitoring

Figure 2 illustrates a TCP diagram working with a sensor or a honeypot. Botnets/worms typically send a TCP syn packet for network scanning or DoS attacks. Usually, as

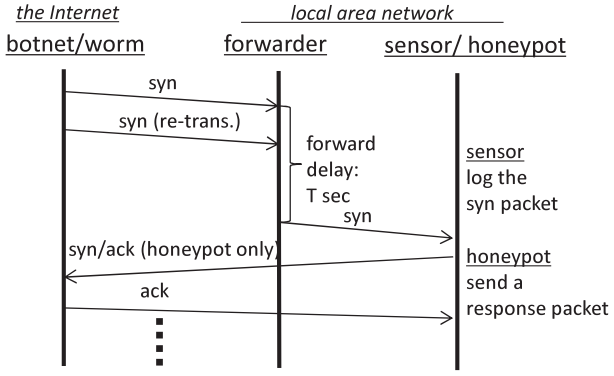

Fig. 2 TCP diagram for flow-based monitoring.

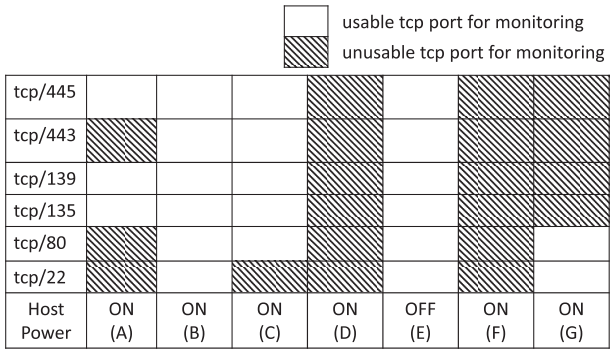

Fig. 3 Multi-dimensional monitoring example.

a part of the TCP three-way handshake sequence, the destination host responds to the TCP syn with a TCP syn/ack packet. In this paper, the TCP syn/ack packet is called a *response* packet. When an attacker's TCP syn packets are sent to an unused/unassigned IP address, no *response packet* is sent because there is no host at the destination of TCP syn packets. If the attacker's TCP syn packets are destined to an IP address that has been blocked by a firewall on an intermediate router or an end host, the end host never respond to the TCP syn with a syn/ack flagged TCP packet because it does not receive the initial TCP syn packet. We focus on a TCP three-way handshake where no syn/ack flagged packet (response) is replied. Our flow-based monitoring considers these flows to be malicious when there are no syn/ack packets (responses).

In order to isolate malicious flows from legitimate flows, it is necessary to decide whether there is a response (i.e., syn/ack) packet. This is not an easy decision because some hosts send TCP syn/ack packets after a long time interval after they receive the initial TCP syn packet. We set a time interval when we monitor TCP syn/ack or rst packets that are sent from the local network. If we have no response (syn/ack or rst) within a specified time interval, the flow is classified as malicious. We call this monitoring time interval a *forward delay*. This is a heuristic approach because a TCP syn/ack or a rst packet may be delayed for a long time. To verify it works correctly, we surveyed the actual delay of syn/ack or rst packets on a campus network. We found only a few delayed packets when we set an appropriate forward delay as described in Sect. 4.2. Figure 2 illustrates the TCP protocol; our flow-based monitoring can be applied to other protocols when a response packet is expected for an original incoming packet in the same session. For example, ICMP can be analyzed using flow-based monitoring because an ICMP reply packet is expected. However, some UDP protocols cannot be monitored by the flow-based method because UDP-based protocols have a one-directional packet flow such as an SNMPv1 trap [11]. In these protocols, a response packet is not expected.

Figure 3 illustrates an example of multi-dimensional monitoring. Our system can monitor each firewall-blocked port on each IP address. This is the reason we call it *multi-dimensional* monitoring, since it utilizes all TCP ports on an IP address while darknet is *single-dimensional* with an IP address. In Fig. 3, the label "Host Power" represents the status of the power of the host along with the status of the assigned IP addresses. The TCP port labels in the figure are example ports used to explain what kinds of ports can be monitored. Our method is able to monitor both active and inactive IP addresses. We can monitor all TCP ports on an IP address when the host is inactive (e.g., Host E) or when its firewall blocks all incoming connections (e.g., Host B). Host A shows a case where an ssh/http/https server is running on the host. In this case, all service ports (tcp/22, 80, 443) are unusable for monitoring though the rest of the ports are available for threat monitoring. An earlier darknet can monitor only the case of Host E and its monitoring address space must be an IP subnet. An conventional DarkPots [10] extends the darknet to utilize isolated address spaces for monitoring, although it covers only the case of Host E. Our multi-dimensional monitoring can utilize all IP addresses in a local network that covers all the cases of Host A to Host G without an active service port. Thus, our system greatly extends the coverage of threat monitoring.

3.3 Component Functions

Figure 4 illustrates the component functions of the proposed architecture. Our system has three ports: an ingress port, a back-end port, and an egress port. The ingress port is connected to a programmable flow switch such as the OpenFlow switch [12] that is located outside of the firewall between the Internet and the local area network. Our system is designed for deployment on the gateway of a campus/enterprise network. This observation point can cover all malware/botnet packets sent from the Internet to the local area network. The flow switch selectively forwards traffic that belongs to the *monitoring class* described in Fig. 5. The back-end port is connected to the monitoring network. Furthermore, we place honeypot and/or sensor servers in the monitoring network. The system forwards anomalous packets through the back-end port and receives honeypot responses to send back to the Internet. The egress port is used to forward honeypots responses from the back-end port to the Internet. This port is necessary for honeypots to establish connections with

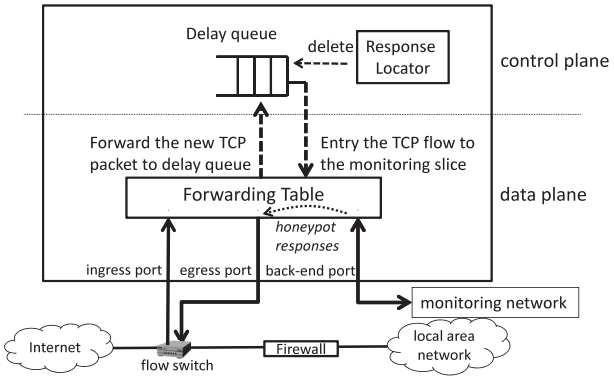

Fig. 4 Component functions.

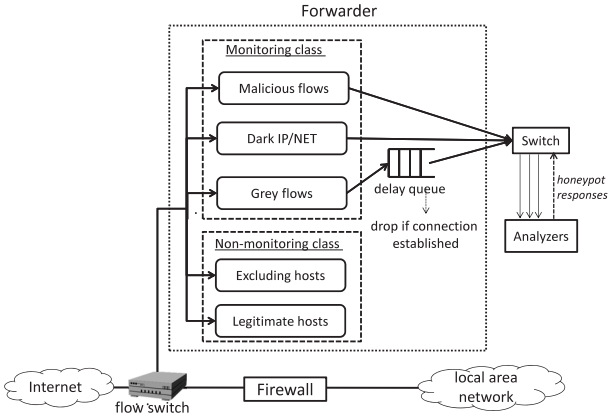

Fig. 5 Overview of the flow class allocation.

botnets/worms in the Internet.

Packets from the ingress port are forwarded to a *forwarding table*. The forwarding table performs flow matching to identify if an incoming packet belongs to one of the records in the table. If no matched flow exists in the forwarding table, and the packet is a *syn* packet, then the packet is pushed to the *delay queue*. The delay queue keeps syn packets for a specified number of timeout seconds, which we call *forward delay*. When the timeout elapses for a packet, its flow is registered in the forwarding table and its syn packet is transmitted to the monitoring network. We place analyzers such as sensors or honeypots on the monitoring network. When we observe a syn/ack flagged TCP packet (response) in the traffic on the ingress port, the response locator immediately removes the matching syn packet kept in the delay queue. It also deletes the corresponding flow record from the *Forwarding Table*.

Our system is installed at the gateway of a campus or an enterprise local network and can monitor all of the malwares/botnets packets sent from the Internet to the local area network.

3.4 Policy-Based Forwarding

In this section, we describe flow-class allocation and policy-based forwarding of our architecture, illustrated in Fig. 5.

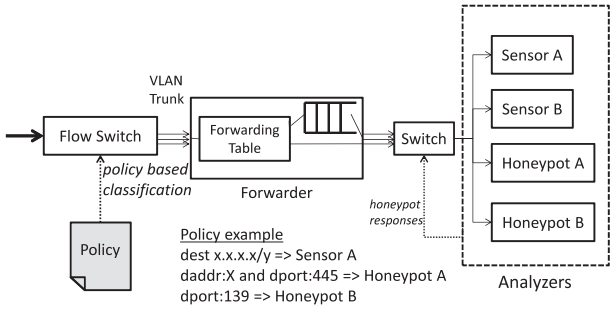

Fig. 6 Overview of the policy-based forwarding.

We develop a *forwarder* system that includes the functions of delayed syn packet detection (Fig. 4), flow classification (Fig. 5), and policy-based forwarding (Fig. 6). All defined classes and hosts/flows in Fig. 5 are subsets of the IP address space of the local area network. Our architecture can cover all the IP addresses on a local area network in principle. However, there is a good reason for excluding some hosts with high-priority services from threat monitoring. It should be noted here that our analyzer includes a honeypot that returns some packets to the original sender. Our system utilizes unused ports even if the IP address is actually used. Furthermore, there is a risk that our honeypot generates malicious packets, which are then sent to a spoofed IP address. If the malicious packets are detected at a remote site, our local network may be warned even if the TCP port is not used on our local network host. This is therefore a good reason for excluding legitimate local host IP addresses from threat monitoring.

Moreover, the performance of our system is affected by the number of TCP sessions, as described in Sect. 4.4 An IP address that creates a significant number of syn packets (e.g., high-load web servers and NAT routers) should be excluded when a performance problem occurs in our system.

To meet these demands, our architecture has a flow classification mechanism. When we place a programmable flow switch on a source of gateway traffic, we can classify flows into two classes: monitoring class and non-monitoring class. Initially, all syn packets in the monitoring class belong to *grey flows*. Grey flow indicates an unidentified flow, regardless of whether it is legitimate or malicious. We determine a flow as malicious only after it has passed through the delay queue. Malicious flows and flows in the class "Dark IP/NET" are forwarded to analyzers. When honeypots are operated as analyzers, response packets are sent back to the Internet through the forwarder.

Figure 6 illustrates the details of policy-based flow forwarding that can be configured with a programmable flow switch. We can flexibly assign a policy group to one of the analyzers with a VLAN assignment. The forwarder components are capable of processing VLAN tagged packets. Therefore, we can perform flexible threat monitoring taking into account various demands of the working network.

Both policy-based forwarding and flow class allocation can be implemented by a flow table in the programmable

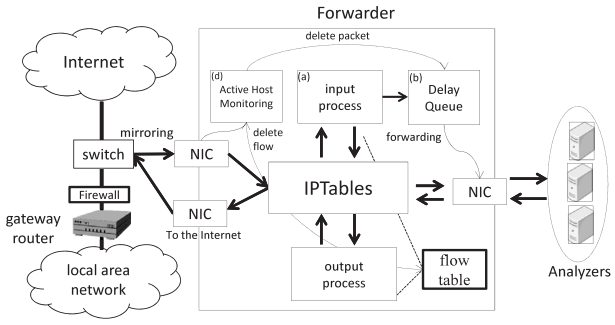

Fig. 7 Implementation of forwarder.

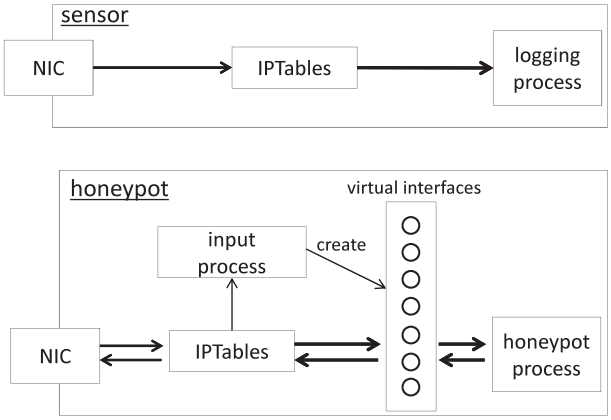

Fig. 8 Analysis server implementation.

flow switch. Figure 5 and Fig. 6 describe the logical components of our system. The actual implementation is described in Sect. 3.5.

3.5 Implementation

In this section, we describe the software implementation of our system. Our proposed architecture is actually designed for hardware implementation useful for a large amount of heavy traffic. However, we first developed a prototype that implements some of our architecture components. We developed a software-based forwarder with the flow table and the delay queue, along with the forwarding mechanism. We also implemented a honeypot wrapper program to enable cooperation with the forwarder. The following section describes each of the details of our developed system.

3.5.1 Forwarder Implementation

Figure 7 illustrates the forwarder implementation. We intend to use a programmable flow switch like OpenFlow for the input source. In our prototype, we use a port mirroring function on an existing gateway switch in the campus network. The forwarder is implemented using Linux OS with three network interfaces. The system is written in C and C++ and the code is approximately 1,500 lines, excluding comments and blanks. We extend the function of iptables using a netfilter-queue library [13]. There is no modification for a kernel source. The netfilter-queue enables the program to hook input or output packets for processing any user process at any chain of iptables with firewall rules.

We developed the forwarder components described in Fig. 7. The input process mainly scans all input traffic from the external switch. Incoming syn packets are stored in the *delay queue* and are never forwarded directly to the analysis servers. The input process refers to the *flow table*. The flow table stores information of current active flows. An entry in the *flow table* is removed after observing a TCP fin or a rst after waiting for a specified number of timeout seconds (30 s in our implementation). When the input flow matches one of the records in the *flow table*, it is forwarded to the analysis servers without adding it to the *delay queue*.

The delay queue keeps syn packets to identify if they are malicious or not. All the syn packets are preserved in the delay queue for the *forward delay* period. The delay queue timer is set for each of packet individually. After the *forward delay* period elapses for a packet, it is removed from the *delay queue* and forwarded to analysis servers, along with adding the flow record to the *flow table*. When a response packet such as a TCP syn/ack or rst is observed by *response monitoring*, it is immediately deleted from the delay queue.

The output process reviews all output traffic from the analyzers. The output also refers to the *flow table* and all packets directly forward to the Internet, as long as their flows exist in the *flow table*. If a flow does not exist in the *flow table*, the packet is not forwarded and it is dropped.

The response monitoring monitors mirrored traffic whether there exists a TCP syn/ack or a rst response in the gateway traffic. When any response packet is observed by response monitoring, it immediately deletes the appropriate syn packet from the delay queue, along with removing the appropriate flow from the *flow table*. We verified that the forwarder works at an average of 300 Mbps for campus traffic with up to 108,402 records. We performed further evaluations in Sect. 4.4.

Our system is implemented in front of the gateway router in Fig. 7. If a firewall is installed such that it bypasses the gateway, the flow switch or the mirroring point should be placed in front of the firewall so that our system can monitor all botnet/worm packets.

3.5.2 Analyzers Implementation

Figure 8 illustrates the implementation of a sensor and a honeypot. A sensor is configured by iptables to receive all incoming packets and block any outgoing packet. We use Nepenthes [14] version 0.2.0 as a honeypot service. Nepenthes is a legacy, low-interaction honeypot that works as a Linux process, though its development was stopped in 2009. Dionaea [15] is a successor to it. Our paper focuses on a flow-based detection system and the forwarding mechanism. Although Nepenthes is obsolete, we use it in our experiment because honeypot is only needed to verify that any process-based honeypot can be deployed without software modification. Another reason for using Nepenhtes is that it

is written in C++, light-weight, and suitable for processing thousands of sessions simultaneously. Dionaea is written in Python and it is difficult to handle a large number of concurrent sessions.

We implemented a wrapper for the honeypot that can handle any session for any IP address. The wrapper is implemented with iptables and netfilter, like the forwarder. When a TCP syn packet arrived from an interface, a virtual interface (also called a sub-interface) is created automatically and the TCP syn packet is forwarded to the honeypot process. The IP address of the sub-interface is the destination address of the TCP syn packet. The honeypot process must be configured to bind to any interface (e.g., 0.0.0.0, or INADDR_ANY). Thus the honeypot process can bind to any IP address and respond to it. The virtual interface is automatically removed after several seconds have elapsed since the last access to the interface.

4. Experiments

In this section, we first describe the configuration of our campus network. Then we present the evaluation results of multi-dimensional monitoring and system performance, with some results related to comparisons between our system and the conventional DarkPots [10] system.

4.1 Configuration

We deploy our system at the gateway on the campus network. This network has a/16 address space with a bandwidth of 10 Gbps. The average daily traffic is around 300 Mbps to 500 Mbps, including some experimental short-term traffic of more than 1 Gbps. All experiments are performed carefully so that legitimate traffic is not affected, with the cooperation of the campus network administration. We configure a span port on the gateway for mirroring all inbound and outbound traffic at the gateway. The mirrored traffic is forwarded to the *forwarder*. We configure a sensor and honeypot as analyzers and place them at the back-end of the forwarder.

In this section, we describe our evaluation of the proposed system, and discuss some cases from the analyzers' operation. The experiment is performed over several days in July, 2011.

4.2 *Forward-delay* Evaluation

We first evaluate the forward delay parameter that plays an important role in multi-dimensional monitoring. The proposed system classifies a flow with no syn/ack response as malicious. We define a forward delay timeout to wait for a syn/ack packet response. To identify this parameter, we first inspect the actual delay of the syn/ack packets on the campus traffic.

Figure 9 illustrates the distribution of the delay of syn/ack with four different time scales from 0 to 30 s. The delay of syn/ack represents the time between a syn packet and a syn/ack packet for each flow. We find that 99.997% of the syn/ack packets are responded within 5 s. However, 0.0029% packets take more time to respond to the syn/ack packets. To make a deep inspection, we track the sessions of the syn-ack-delayed flow. Table 1 describes the tracking of the sessions in which the syn/ack packet is delayed for more than 2, 5, and 10 s. These times correspond to the *forward delay* time of our system. According to the results, a short *forward delay* implies that a considerable number of packets are malicious, although there are established flows that may cause false-positives. When we wait for 5 s or more, there are no established flows.

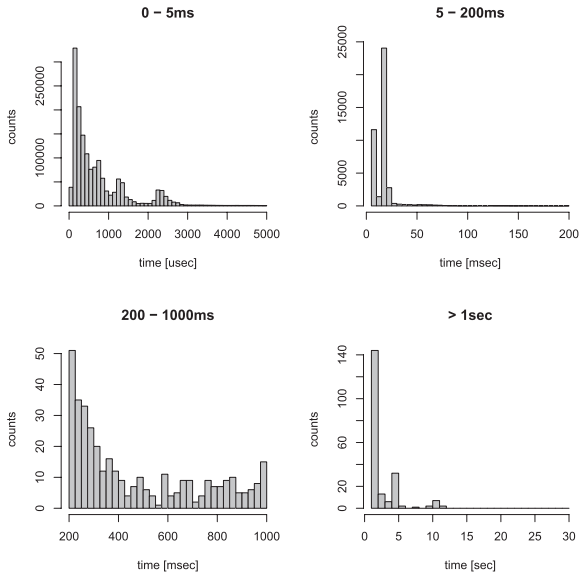

Fig. 9 TCP syn/ack delay of response observed in the campus traffic.

Table 1 The number of delayed syn/ack with session trace.

forward delay	2 sec	5 sec	10 sec
delayed syn/ack pkts ratio	0.0063%	0.0029%	0.0011%
delayed syn/ack pkts [#]	99	46	18
established [#]	19	0	0
rejected [#]	51	28	3
no response [#]	29	18	15

Figure 10 illustrates the analysis of the trade-off between the delay queue size and the delayed syn/ack ratio. The elongation of the *forward delay* has the merit of decreasing the delayed syn/ack ratio, although the delay queue requires a relatively large amount of memory to preserve syn packets. Note that we observed no established sessions when we set the *forward delay* to 4 s or more. Considering the overall results, we conclude that the 5 s *forward delay* is the most balanced delay in our environment.

It must be mentioned that there are many delayed abnormal syn-ack packets sent from some specific hosts. These hosts are Planetlab [16] nodes, and their TCP session characteristics are totally different from other hosts. For this reason, we exclude all packets from Planetlab nodes through our experiments with flow class allocation. Excluding PlanetLab nodes is sufficient to apply our system to our campus

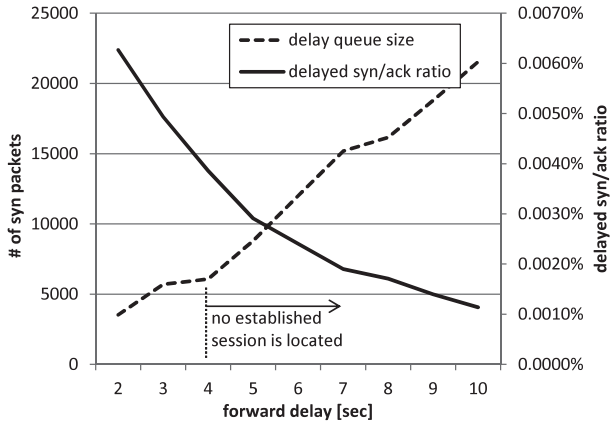

Fig. 10 Trade-off between the delay queue size and the delayed syn/ack ratio.

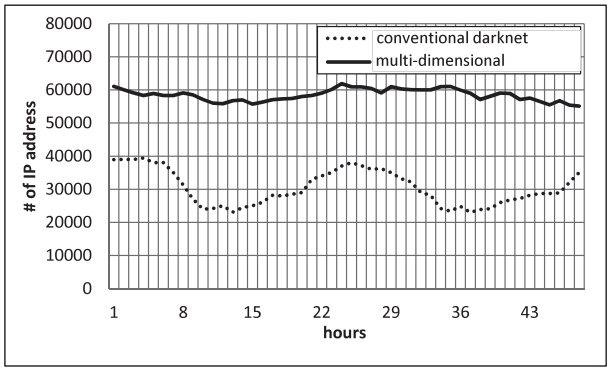

Fig. 11 The IP coverage used for threat monitoring.

network. However, in other network environments, there may be other reasons for the delay in syn/ack packets, such as a high server load, freezing or bugs in kernel or network applications. It future, we intend to clarify the causes of the syn/ack delay, and modify our system accordingly.

4.3 Effect of Multi-Dimensional Monitoring

In this section, we evaluate the effects of multi-dimensional monitoring in terms of address coverage and sensor/honeypot operation using a comparison between our system and the conventional DarkPots system.

4.3.1 Address Coverage

Figure 11 describes the coverage of the monitoring IP address, comparing the multi-dimensional monitoring with the conventional darknet. The campus network has a/16 subnet; thus, there are a maximum of 65,536 IP addresses, including the network/broadcast address. Inactive IP addresses for the darknet are located by the estimation of the firewall rules on the gateway of the DarkPots system [10], [17]. Our system can utilize both inactive and active IP addresses for monitoring. We note that our system can cover an average of 89% of the total addresses, while darknet-based monitoring only

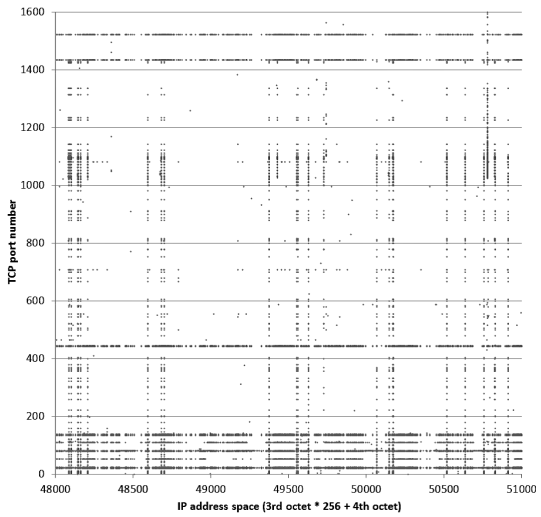

Fig. 12 The IP coverage distribution.

covers an average of 46%.

Figure 12 illustrates the two-dimensional analysis of the monitoring IP-port distribution on active hosts. The x-axis represents the one-dimensional mapping for a subnet of the campus network. The y-axis represents TCP port numbers from 1 to 1600. Inactive IP addresses are excluded from this figure; hence, the result intends to display malicious packets against the active IP addresses. The observation is performed for one hour. A dot in the figure represents that this port received at least two packets for one hour. In this result, we can admit several traversal hosts or port scanning behaviors through the address space. The vertical dotted line indicates that this IP address is port scanned. The horizontal dotted lines indicate a frequent host scan on this port. It is a novel result that allows us to monitor malicious packets against active hosts. Network administrators can perform sensor or honeypot monitoring even if most of their network is occupied by active hosts.

4.3.2 Analyzer Logs

In this section, we present case studies of the analysis carried out by operating a sensor and a honeypot with our system. Table 2 illustrates the TCP protocol breakdown with the number of packets and unique hosts captured by our system and the conventional DarkPots. Both systems use the same sensor configuration. Only the monitoring method is different: our system uses multi-dimensional monitoring, and the DarkPots utilizes unused address-based monitoring. We verify that our system can collect more anomalous packets than the conventional DarkPots. This is attributed to the fact that our system can monitor both active and inactive IP addresses, while darknet and DarkPots can only monitor inactive IP addresses.

Table 3 describes the session count and malware logged by Nepenthes honeypots on both our system and the conventional DarkPots. Session represents the attack attempt

Table 2 TCP protocol breakdown of packets captured by sensors.

	packets [#]		unique [#]	
	our system	DarkPots	our system	DarkPots
n (IP address)	7,291		3,997	
tcp/21	12,441	7,910	66	21
tcp/22	305,016	190,405	359	174
tcp/80	310,507	89,999	8,385	1,855
tcp/135	82,914	55,116	373	210
tcp/139	38,169	22,267	902	453
tcp/443	89,217	43,304	7,373	1,599
tcp/445	6,616,146	2,894,336	478,968	227,033
tcp/1433	1,340,395	891,779	1,823	1,048
tcp/3389	169,670	113,301	6,262	3,534

Table 3 Honeypot sessions and malware logged by honeypots.

	our system	conventional DarkPots
session	91,112	51,231
n (hash)	24	19
n (malware)	11	7
n (IP address)	6,511	3,132

logged by honeypots, n(hash) represents the hash-based malware counts and n(malware) denotes the name-based malware counts. The monitoring address space is a specified/19 subnets including a considerable number of active hosts. We show that our system can collect more number of attack logs than conventional DarkPots. The number of malware hashes and the malware names also show the priority of our system. This result is attributed to the fact that our system can cover more IP addresses with multi-dimensional monitoring.

4.4 Performance Evaluation

In this section, we describe the log analysis of the delay queue and performance measurement. Figure 13 illustrates the number of add/send/delete events on the *delay queue* for a syn packet. An *add* event denotes that a syn packet is pushed by an *output process*. A *send* event denotes that a syn packet is forwarded to the analyzers after the elapsed *forward delay*, whereas a *del* event denotes that a syn packet is deleted by *active host monitoring* because an actual syn/ack or a rst packet is located from the monitoring network. When we define the number of *add* events as ΔA, *send* events as ΔS, and *del* as ΔD events, we can have $\Delta S = \Delta A - \Delta D$. The *delay queue* filters malicious packets. We note that there are several spikes in Fig. 13. These are caused by the host scanning by botnets/worms; these spikes are mostly forwarded by the *send* events. We also note that the transit of *send* events is fairly flat, while the number of *del* events increases around 40,000 to 55,000 s. The *del* events correspond to the number of legitimate flows, and the transition in the number of located malicious packets is not affected by an increase in legitimate traffic.

Figure 14 illustrates the memory usage of our system along with the sizes of the flow table and the delay queue. The performance of our flow-based system is not affected by the address space range but by the number of flows that

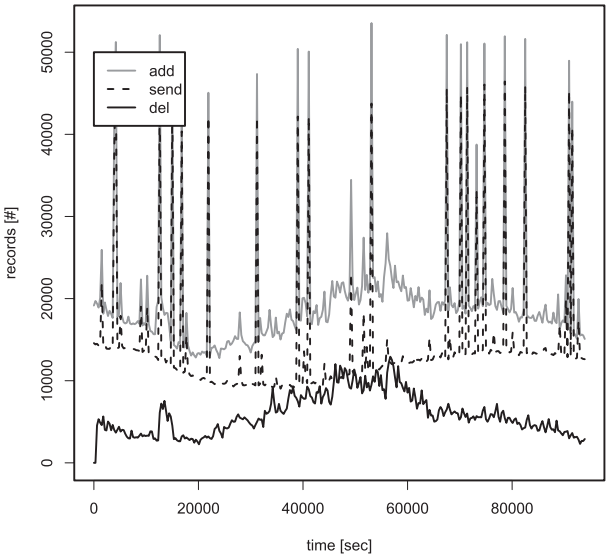

Fig. 13 The number of syn packets with *delay queue* events.

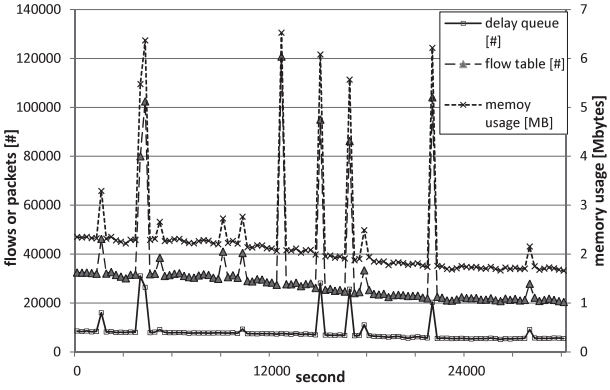

Fig. 14 Memory usage of the system.

the system handles at a particular moment. The memory usage for one syn packet is low. In our implementation, we require a maximum of 72 bytes for one syn packet including syn IP/TCP headers, buffer for option headers, and some metadata such as the *delay queue* timer, and a maximum of 46 bytes for a flow in the flow table, including a flow hash, timers, and other metadata. As long as we operate our system in our campus network, we use a maximum of approximately 10 Mbytes of memory, and there are no performance problems. In future, we intend to evaluate the system for relatively large traffic that includes a large number of flows and interface bottlenecks, such as packet drops.

5. Conclusion

This paper proposes a multi-dimensional malicious packet monitoring architecture for Internet threat detection. We implemented an efficient mechanism to forward packets to sensors or honeypots. The proposed system has high IP address coverage for threat monitoring compared to conventional darknets. The new multi-dimensional monitoring can utilize both inactive and active IP addresses. Thus it can

be applied to the network which has plenty of active IP addresses. In addition, the proposed flow-based method only monitors active flows with the forwarding table. The system uses less memory than conventional DarkPots, even if the monitored address space is large. The new system is also suitable for monitoring a very large address space with low address utilization like an IPv6 environment.

Our new system is applied to a campus gateway and we performed several experiments. The new multi-dimension monitoring is found to be effective. We verify that there is no established session if a TCP syn/ack packet is delayed by more than the *forward delay* time on the campus network. After a trade-off analysis of the delay queue size and the delayed syn/ack ratio, we set the *forward delay* time to 5 seconds. Then, we compared the IP coverage of our system with the conventional DarkPots. The new system achieved 89% IP address coverage while the conventional DarkPots covers 46%. We then examined the performance of our system. The memory usage of the proposed system depends on the number of active flows. Finally, we demonstrate that the new system can collect more logs than the conventional DarkPots for sensor and honeypot operations. We are currently working on a more detailed inspection of the sys/ack delay time. We are also planning to apply our system to an IPv6 environment as a future work.

References

[1] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G.M. Voelker, V. Paxson, and S. Savage, "Spamcraft: An inside look at spam campaign orchestration," Proc. Second USENIX Workshop on Large-scale Exploits and Emergent Threats (LEET), Boston, USA, April 2009.
[2] C. Economics, "Malware report: The economic impact of viruses, spyware, adware, botnets, and other malicious code," Tech. Rep., Computer Economics, 2007.
[3] "McAfee Threats Report: Second Quarter 2011," http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q2-2011.pdf, McAfee Inc., accessed Oct. 10. 2011.
[4] T. Cymru, "The darknet project," Internet: http://www.cymru.com/Darknet, 2004.
[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," IEEE Security & Privacy, vol.1, no.4, pp.33–39, 2003.
[6] D. Inoue, M. Eto, K. Yoshioka, S. Baba, K. Suzuki, J. Nakazato, K. Ohtaka, and K. Nakao, "nicter: An incident analysis system toward binding network monitoring with malware analysis," WOMBAT Workshop on Information Security Threats Data Collection and Sharing, pp.58–66, 2008.
[7] Y. Toda, N. Matsumoto, and Y. Miyagawa, "Isdas: Internet scan data acquisition system," Joho Shori Gakkai Shinpojiumu Ronbunshu, vol.2004, no.11, pp.3A–4, 2004.
[8] M. Ishiguro, H. Suzuki, I. Murase, and H. Ohno, "Internet threat detection system using bayesian estimation," Proc. 16th Annual Computer Security Incident Handling Conference, 2004.
[9] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. Snoeren, G. Voelker, and S. Savage, "Scalability, fidelity, and containment in the potemkin virtual honeyfarm," ACM SIGOPS Operating Systems Review, vol.39, no.5, pp.148–162, 2005.
[10] A. Shimoda, T. Mori, and S. Goto, "Sensor in the dark: Building untraceable large-scale honeypots using virtualization technologies," 2010 10th Annual International Symposium on Applications and the Internet, pp.22–30, July 2010.
[11] "A simple network management protocol (SNMP) — RFC 1157," http://www.ietf.org/rfc/rfc1157.txt, May 1990.
[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol.38, no.2, pp.69–74, 2008.
[13] R. Russel, M. Boucher, J. Morris, and H. Welte, "The netfilter/iptables project," 1999.
[14] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, "The nepenthes platform: An efficient approach to collect malware," Lect. Notes Comput. Sci., vol.4219, pp.165–184, 2006.
[15] "dionaea - catches bugs," http://dionaea.carnivore.it/, The Honeypot Project, accessed Oct. 10. 2011.
[16] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: An overlay testbed for broad-coverage services," ACM SIGCOMM Computer Communication Review, vol.33, no.3, pp.3–12, 2003.
[17] A. Shimoda and S. Goto, "Flow-based Internet threat detection with dark IP," IEICE Trans. Commun. (Japanese Edition), vol.J92-B, no.1, pp.163–173, Jan. 2009.

Akihiro Shimoda is currently a research assistant at Waseda University, Tokyo, Japan. He received B.E., M.E., and Doctor of Engineering degrees from Waseda University in 2007, 2008, and 2011, respectively. He is actively engaged in the research of network security and network management. He was a research associate of the Global COE program from Mar 2008 to Mar 2009. He became a research assistant of the Waseda University in Apr 2009. He received Best Student Paper Award from IEEE/SAINT in 2010.

Tatsuya Mori is currently a senior researcher at NTT, Tokyo, Japan. He received B.E. and M.E. degrees in applied physics, and Ph.D. degree in information science from the Waseda University, Tokyo, Japan, in 1997, 1999, and 2005, respectively. Since joining NTT corporation in 1999, He has been engaged in the research of management of large-scale networked systems and network security. From Mar 2007 to Mar 2008, he was a visiting researcher at the University of Wisconsin-Madison. He received Telecom System Technology Award from TAF in 2010 and Best Paper Awards from IEICE and IEEE/ACM COMSNETS in 2009 and 2010, respectively. Dr. Mori is a member of ACM.

Shigeki Goto is currently a professor at the Department of Computer Science and Engineering, Waseda University, Japan. He received B.S. and M.S. degrees in mathematics, and Ph.D. degree in Information Engineering from the University of Tokyo, in 1971, 1973, and 1991, respectively. He joined NTT Research Laboratories in 1973. He is actively engaged in research projects in computer architecture, natural language processing, deductive computer program synthesis, and computer networks. From Aug 1984 to Aug 1985, he was a visiting researcher at Stanford University. He became a professor at Waseda University in 1996, He is the president of JPNIC. He is also a visiting professor at NII, Japan. Dr. Goto is a member of IPSJ, JSSST, JSIAM, JSAI, ACM, and IEEE.