4

Intrusion Detection and Prevention in High Speed Network

Kuo Zhao and Liang Hu
Jilin University,
China

1. Introduction

With the rapid development and comprehensive application of network technology, network security problems gradually appear serious. Traditional firewall technologies can't provide sufficient security protection against various attacks and intrusions (Anderson, 1980), while intrusion detection systems (IDS) are faced with compromise between false alarms and false positives (Denning, 1987). In this chapter, we investigate intrusion detection and intrusion prevention in high speed network, introduce related technology and our research results.

1.1 The information system and system security

Information system is an integrated set of components for collecting, storing, processing, and communicating information. Information systems are more than just computer programs. Though information and communications technologies are playing an increasing role in meeting organisations' information needs, an information system is a much more general concept. It refers to the wider systems of people, data and activities, both computer-based and manual, that effectively gather, process, store and disseminate organisations' information. Of course, system security is essential for information system. In another words, security is the most reliable foundation for information system.

1.2 The actual condition of information system security

With the development of Internet, the world economy has been deeply communed together. The nation is just like a huge network computer, and computer network has been the foundation and life vein of a nation's economy. As the entire society increasingly relies on network infrastructures, network security also changes for the worse seriously. It is very difficult for traditional security policies or mechanisms (such as authentication, cryptography and firewall) to prevent network attacks. The whole society needs new technology to solve those problems.

The openness of the system network, the security hole of the network protocol, the defects of the software…Those drawbacks make the network security worse than worse. According to the recently research and report, people found the details and data of network attack easily. The high occurrence probability makes the problem urgent (Allen et al., 2000).

Cases are known, the network security is the most reliable foundation for network applications. Every country, for commercial or military purposes, spared a lot to study network security. Research on this issue

Although there are various measures to protect safety, they are not the keys to all kinds of attack. For instance:

1. A perfect design of software safety is impossible.
2. Encryption technology itself has some problems, and those shortcomings may lead to keylogger activities. Moreover, people may misunderstand the arithmetic.
3. The security hole of the network protocol.
4. The contradiction between the availability and the safety is always one of those contradictions running through the long developing process of computer technology.
5. The complex security system is usually difficult to configure. The blander of wrong configuration will leave some hidden danger.
6. The system log and the audit have massive data. They need automatic mode to work with those information.
7. Staff members may abuse the safety system.

This chapter presents the corresponding research work on the intrusion detection and intrusion prevention in large-scale high-speed network environment and is organized as follows: firstly, a distributed extensible intrusion prevention system is provided, then various packet selection models for intrusion detection systems based-on sampling are illustrated, again the design and implementation of a high-speed traffic collection platform based-on sampling on FPGAs and the research of trusted communication protocol for intrusion prevention system are presented, last we draw concclusions.

2. DXIPS: A distributed extensible intrusion prevention system

2.1 Related technology

2.1.1 Snort_inline

The Snort_inline IPS is a modified version of the famous Snort IDS. It receives packets sent from the Netfilter firewall with the help of the lipipq library[1], compares them with Snort signature rules and tags them as drop if they match a rule, then finally sends them back to Netfilter where the Snort_Inline tagged packets are dropped.

There are 5 available default actions in Snort, alert, log, pass, activate , and dynamic:

1. alert-generate an alert using the selected alert method, and then log the packet
2. log-log the packet
3. pass - ignore the packet
4. activate - alert and then turn on another dynamic rule
5. dynamic - remain idle until activated by an activate rule , then act as a log rule

There are three rule options more than Snort's:

1. Drop – The drop rule tells iptables to drop the packet and log it via usual snort means
2. Sdrop – The sdrop rule tells iptables to drop the packet. Nothing is logged.

[1] Libipq library is a development library for iptables userspace packet queuing

3. Reject – The reject rule type tells iptables to drop the packet; log it via usual snort means; and send a TCP reset if the protocol is TCP or an ICMP port unreachable if the protocol is UDP.

2.1.2 Netfilter

Along with the development of the technology used in Linux firewall, Netfilter comes into being. It is well known that Netfilter is a framework that provides hook handling within the Linux kernel for intercepting and manipulating network packets.

Netfilter is made up of five hook functions towards IPV4, IPV6, and IPX. The identifiers of all hooks for each supported protocol are defined in the protocol-specific header file. The following five hooks are defined for IP Version 4 in <linux/netfilter_ipv4.h>:

NF_IP_PRE_ROUTING (default value is 0): incoming packets pass this hook in the ip_rcv() (linux/net/ipv4/ip_input.c) function before they are processed by the routing code

NF_IP_LOCAL_IN (default value is 1): all incoming packets addressed to the local computer pass this hook in the function ip_local_deliver()

NF_IP_FORWARD (default value is 2): all incoming packets not addressed to the local computer pass this hook in the function ip_forward()

NF_IP_LOCAL_OUT (default value is 3): all outgoing packets created in the local computer pass this hook in the function ip_build_and_send_pkt()

NF_IP_POST_ROUTING (default value is 4): this hook in the ip_finish_output() function represents the last chance to access all outgoing (forwarded or locally created) packets before they leave the computer over a network device

There are five return values in the hook functions:

NF_DROP (default value is 0): The active rules list processing is stopped, and the packet is dropped

NF_ACCEPT (default value is 1): The packet is passed to the next packet filter function in the rules list. Once the end of the list has been reached, the packet is released by okfn() for further processing

NF_STOLEN (default value is 2): The packet filter function withholds the packet for further processing, so that the active rules list processing is stopped. In contrast to NF_DROP, however, the packet does not have to be explicitly dropped

NF_QUEUE (default value is 3): The function nf_queue() (net/core/netfilter.c) puts the packet in a queue from which it can be removed and processed (e.g., by a user space program). Subsequently, nf_reinject() has to be invoked to return the packet to the Linux kernel for further processing by netfilter

NF_REPEAT (default value is 4): In contrast to NF_ACCEPT, rather than a continuation of processing at the next packet-filter function, the current filter function is invoked again

2.1.3 Iptables

Iptables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

A firewall rule specifies criteria for a packet, and a target. If the packet does not match, the next rule in the chain is the examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain or one of the special values ACCEPT, DROP, QUEUE, or RETURN.

ACCEPT means to let the packet through. DROP means to drop the packet on the floor. QUEUE means to pass the packet to userspace. (How the packet can be received by a userspace process differs by the particular queue handler. 2.4.x and 2.6.x kernels up to 2.6.13 include the ip_queue queue handler. Kernels 2.6.14 and later additionally include the nfnetlink_queue queue handler. Packets with a target of QUEUE will be sent to queue number '0' in this case. Please also see the NFQUEUE target as described later in this man page.) RETURN means stop traversing this chain and resume at the next rule in the previous (calling) chain. If the end of a built-in chain is reached or a rule in a built-in chain with target RETURN is matched, the target specified by the chain policy determines the fate of the packet.

2.2 Architecture of DXIPS

DXIPS is a distributed network IPS based on the combination of Snort_inline and Netfilter firewall configured by IPtables, and it provides the ability to detect malicious network traffic, drop or reject attack packets, and perform intrusion detection and prevention on 4-7 layers of network protocol.

Hierarchical structure is applied to the architecture of DXIPS, which consists of three layers:

Intrusion prevention layer: monitor the network traffic passing by and perform intrusion detection and prevention.
Server layer: collect log data and save to readable format.
Control layer: analysis console and perform data display.

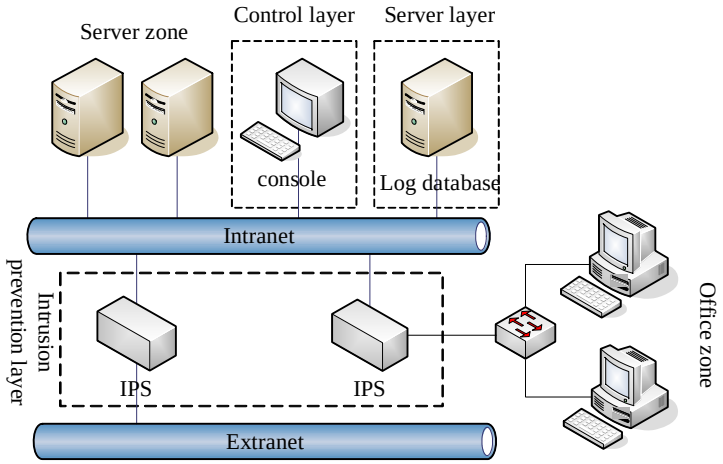


Fig. 1. Architecture of DXIPS

2.3 Construction and implementation of DXIPS

DXIPS is composed of four modules: intrusion prevention module, log record module, central control module and communication module. These four modules coordinate with each other and perform intrusion prevention in distributed network environment.

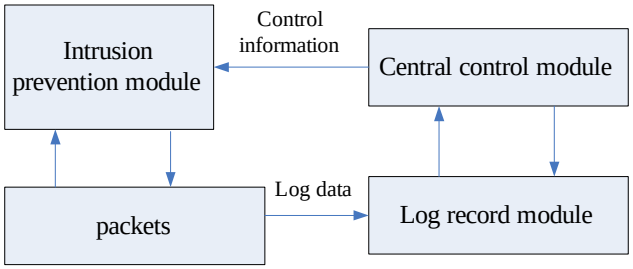


Fig. 2. Construction of DXIPS

Intrusion prevention module runs on intrusion prevention layer, and it is responsible for capturing packets, intrusion detection and prevention. This module is to be deployed at the crucial position of network, such as the network link between intranet and extranet, thus all the packets are to be monitored. This module is based on the combination of Snort_inline and Netfilter firewall configured by IPtables, and consists of three submodules such as packets capture, packets detection and response.

Log record module runs on server layer and aims for log collection and formatting. The collected logs include intrusion detection log generated by Snort_inline and firewall log by the configuration of IPtables.

Central control module is the core of the whole system and runs on control layer. It is in charge of coordinating other modules and performing management operations such as the configuration of IPS, management of log server, data analysis and load balancing.

Communication module has the ability to provide secure and reliable communication channels between modules.

2.3.1 Intrusion prevention module

Intrusion prevention module is to be deployed at the crucial position of network and needs the ability of routing, that is, packets with correct route information should be delivered from source address to destination address. Thus normal communication between intranet and extranet is to be done. The part of function of routing in DXIPS makes use of the default route module in Linux system, and the command is as follows:

"echo 1>/proc/sys/net/ipv4/ip_ forward"

When packets are captured by intrusion prevention module, the function of IP forward is active, and kernel will deliver packets based on address information of packets and routing table information.

1. Packets capture

Packets capture procedure is to pass packets from kernel space to user space, and includes event generator, Netfilter hook program, IPtables, ip_queue kernel module and netlink interface.

During the procedure of packets capture, intrusion prevention module looks on the filtered packets by Netfilter firewall configured by IPtables as data source. It may reduce the amount of packets to be detected by performing intrusion detection on the specific traffic with regard to security policy. IPtables, which is a packet filter firewall running on data link layer and network layer, may firstly filter packets according to security policy, then pass the filtered packets to Snort_inline and perform intrusion detection.

Specific types of packets are to be accepted by Snort_inline according to the configuration of IPtables. For example, SMTP traffic is to be monitored by Snort_inline according to the configuration of IPtables. The following parts present the commands:

"iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p tcp --dport 25 -j QUEUE"

"iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT"

"iptables -A FORWARD -p tcp --dport 25 -m state --state NEW -j QUEUE"

2. Packets detection

Packets detection procedure is to perform intrusion detection corresponding with rules set, and includes event analyzer, libipq library, Snort_inline and rules set.

Intrusion prevention module captures packets according to security policy and parses the structure of packet based on protocol specifications, then executes data formatting and performs string matching on packets with rules set.

Fig.3 illustrates the packets detection procedure. Snort_inline executes initialization based on command line parameters, and working mode is specified by parameters. Then two dimensional linked lists are constructed based on rules by parsing rules library. Again, Snort_inline performs intrusion detection on packets by calling the function "ProcessPacket( )" after packets are accessed cyclically from structure "ip_queue". Function "ProcessPacket( )" firstly executes protocol parsing and set structure "Packet", then calls detection function "Detect( )" to perform detection according to rule linked lists in certain turn, lastly returns detection results.

Corresponding response actions based on packet detection results may be executed. There are 5 available default actions in Snort, alert, log, pass, activate, and dynamic. In addition, there are additional options which include drop, sdrop and reject in Snort_inline. Action "drop" is to make IPtables drop the packet and log the packet; action "sdrop" is to make IPtables drop the packet but does not log it; action "reject" is to make IPtables drop the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.

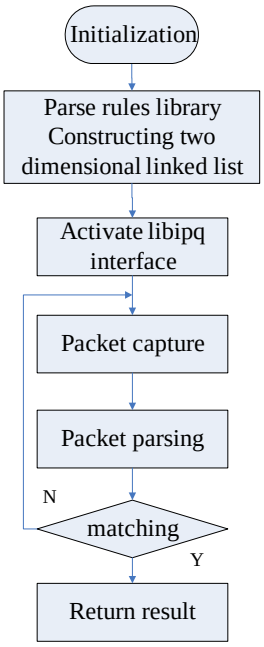


Fig. 3. Flow diagram of packets detection

2.3.2 Log record module

Log record module is applied to record and audit the time when event happened and corresponding information of subjects and objects, and it aims for providing sufficient information for detailed analysis of security events later.

Log record module collects log of intrusion prevention module, saves it to log server, and provides analysis data for security policy specified by central control module. The collected logs include intrusion detection log generated by Snort_inline and firewall log by the configuration of IPtables.

Snort_inline will generate corresponding log based on specific parameter of rule action such as alert, log, drop and reject. The default directory of log files is "/var/log/snort".

The Netfilter firewall log may be accessed with LOG target of IPtables. To make use of LOG target, ipt_LOG module is needed to be loaded as follows:

"/sbin/modprobe ipt_LOG"

For instance, all the connection information passing by may be recorded as follows:

"#iptables -A FORWARD -j LOG"

LOG target is dedicated for recording detailed packet information such as IP header and other useful information. The default directory of log of IPtables is "/var/log/message", which is done by the "syslogd"daemon.

There are three manners for log server to collect intrusion detection log of Snort_inline and firewall log of Netfilter configured by IPtables:

All the nodes of intrusion prevention module directly interact with log server

This can be done by outputting log records directly to log database with Snort_inline output plugin and LOG target of IPtables. The defect of this manner is to improve the system overhead in that intrusion prevention module needs to execute log operations as well as perform intrusion detection and prevention. Additionally, this manner compromises the extensibility of the whole system for the duplicated implementation of log recording.

Log server accesses logs saved in every intrusion prevention nodes at regular time

Though this manner decreases the overhead of intrusion prevention module, it compromises real time performance of the whole system. Log server can't get log information in time, thus central control module is unable to monitor the protected network in real time.

Apply specialized log collection daemon

These daemons are responsible for receiving and dispatching logs of various intrusion prevention nodes and saving to log server in distributed network environment. Compared to the preceding manners, this manner shortens handling time, reduces system overhead, and provides better extensibility.

2.3.3 Central control module

Central control module is the core of the whole system, in charge of coordinating various modules, and conducts centralized management. It locates at central position of the whole system, and connects with intrusion prevention module and log record module. The main functions of central control module include:

Customized policy: central control module is able to establish corresponding policies based on intrusion detection and prevention logs. Log management: central control module can analyze intrusion prevention logs and firewall logs saved in log server in real time and provide corresponding statistical information.

System management: central control module has the ability to manage various intrusion prevention nodes by console, such as dynamical management of intrusion detection rules and firewall rules, management of log server, network traffic statistics and load balancing.

2.3.4 Communication module

It is an important component for secure and reliable communication among various modules of DXIPS. This module mainly refers to the communication among central control module, intrusion prevention module and log record module.

To achieve the interoperation among modules, there is a need of general communication protocol, which contributes to simplifying the management of DXIPS. For example, if central control module is to disable the intrusion prevention function of certain node, a control message is sent to this node by console, and then this node returns an acknowledgement message after it quits normally.

The general communication protocol is derived from standard TCP/IP protocol with farther encapsulation, and it is an application layer protocol.

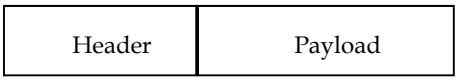


Fig. 4. Customized protocol unit

Customized protocol header is composed of version, type and total length. Type field specifies the type of message such as control message, log message or acknowledgement message according to specific functions. Meanwhile, type field specifies the format of data.

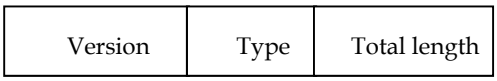


Fig. 5. Format of customized protocol header

The format of payload field varies according to type field. The format of a sample log message is as follows:

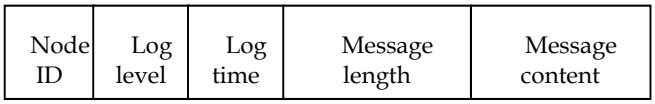


Fig. 6. Format of payload field of a sample log message

Customized communication protocol provides the better extensibility for later new function to be updated or new communication needs. Also it supports encryption manners for secure and reliable communication.

2.4 Deployment of DXIPS

DXIPS provides flexible deployment strategies and better extensibility. It can be expediently deployed in distributed network environment corresponding with various securities prevention needs, and presents a comprehensive protection.

2.4.1 Border prevention deployment

With the rapid development of Internet, great deals of business applications rely on Internet. However, there are various security threats, such as worms, computer viruses, spywares and DDoS attacks, towards the entrance to the Internet of an intranet due to the openability of Internet.

To minimize external network security risks, intrusion prevention module may sit on the entrance to Internet, examine traffic, and block malicious or suspect code in real time. As shown in figure 7.

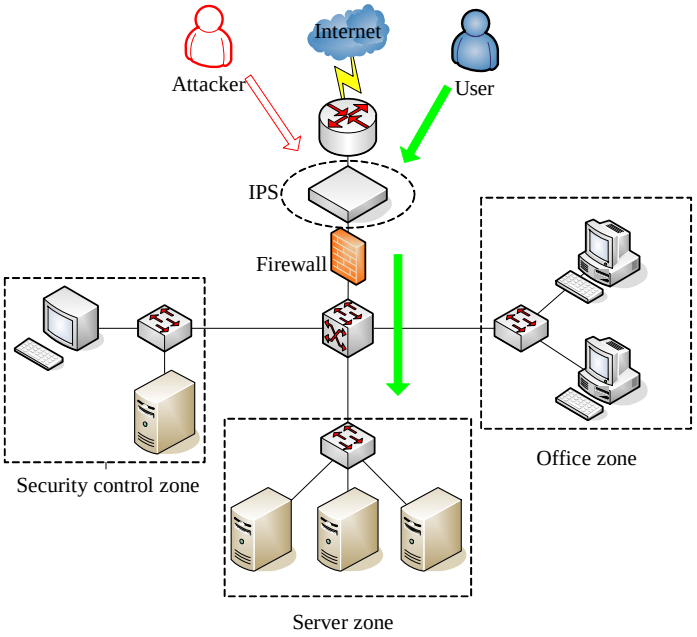


Fig. 7. Border prevention deployment

2.4.2 Key zones prevention deployment

External security threats, such as worms, computer viruses, spywares, may be introduced into intranet by inconsciently users. To minimize internal network security risks, intrusion prevention module may sit on the entrance to office zone and key server zone, block malicious traffic in real time, filter worms, computer viruses and spywares from office zone, and protect the key network server. As shown in Fig. 8.

2.4.3 Hybrid prevention deployment

To minimize external and internal network security risks simultaneously, intrusion prevention module may sit on the entrance to key network link, block malicious traffic in real time, and protect the key network resources.

Meanwhile, intrusion prevention module may sit on the entrance to key network link in bypass mode, which can be treated as intrusion detection system, and performs analysis and detections on intranet. As shown in figure 9.

2.5 Related work

There are various research and commercial work on the design and implementation of IPS. Tan and Weinsberg attempt to improve string-matching algorithms for intrusion detection and prevention on large-scale high-speed network traffic; Drinic (Drinic & Kirovski, 2004) and Weaver (Weaver et al., 2007) present the hardware implementation of IPS based on field programmable gate arrays; Green uses a generic and reliable model to anticipate future

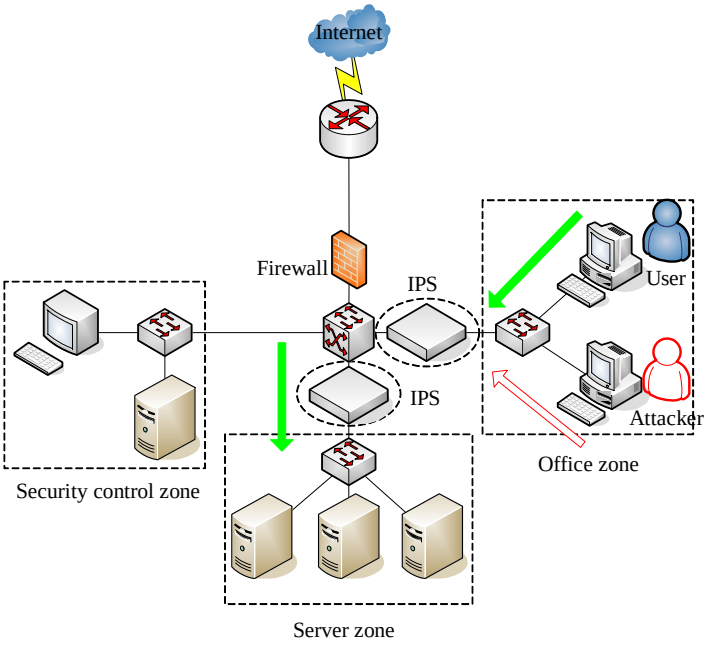


Fig. 8. Key zone prevention deployment

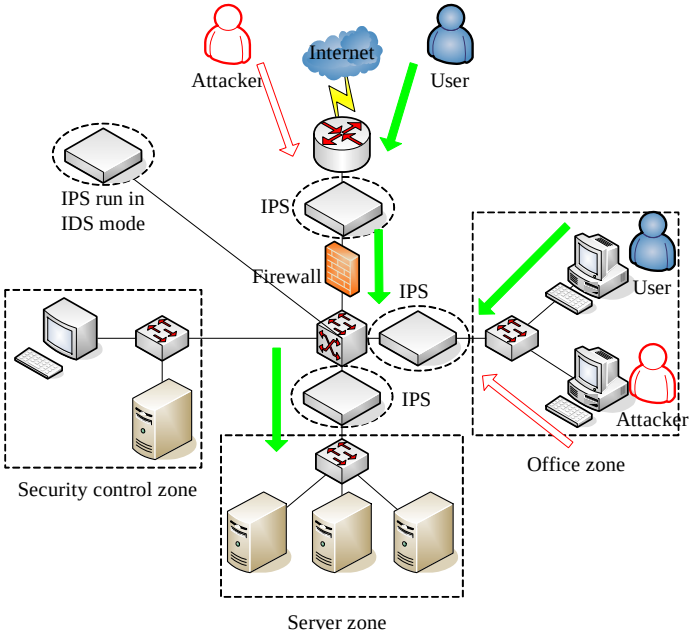


Fig. 9. Hybrid prevention deployment

attack scenarios; Uppuluri provides a practical approach to detect and prevent race condition attacks (Uppuluri et al., 2005). What's more, there are many commercial IPS products available such as TippingPoint IPS, ISS IPS, Cisco IPS and NetKeeper IPS, and these representative products are online, network-based solution, designed to accurately identify, classify, and stop malicious traffic, including worms, spyware/adware, network viruses, and application abuse.

2.6 Summary

This subsection presents the design and implementation of DXIPS, a distributed extensible intrusion prevention system, which is composed of intrusion prevention module, central control module, log record module and communication module. And DXIPS provides an extensible architecture of intrusion detection and prevention in distributed network environment.

3. Packet selection model for intrusion detection system based-on sampling

3.1 Packet selection model

There are many sampling model of statistics, while there are few applied to computer network especially high speed network. For real time demands of network packets processing, packet selection model have to conform to high precision and simple applications. In this paper, systematic sampling, Poisson sampling and stratified sampling methods are applied to network packets selection.

3.1.1 Systematic sampling

Systematic sampling describes the process of selecting the starting points and the duration of the selection intervals according to a deterministic function. This can be for instance the periodic selection of every n-th element of a trace but also the selection of all packets that arrive at pre-defined points in time. Even if the selection process does not follow a periodic function (e.g. if the time between the sampling intervals varies over time) we consider this as systematic sampling as long as the selection is deterministic.

The use of systematic sampling always involves the risk of biasing the results. If the systematics in the sampling process resembles systematics in the observed stochastic process (occurrence of the characteristic of interest in the network), there is a high probability that the estimation will be biased. Systematics (e.g. periodic repetition of an event) in the observed process might not be known in advance.

3.1.2 Poisson sampling

Poisson sampling is an example of random additive sampling. In this sampling, samples are separated by independent, randomly generated intervals that have a common statistical distribution. In general, Poisson sampling avoids synchronization effects and yields an unbiased estimate of the property being sampled.

If sampling function obeys an exponential distribution with rate λ, that is $F(t) = 1 - e^{-\lambda t}$, then the arrival of new samples cannot be predicted (and, again, the sampling is unbiased).

Furthermore, the sampling is asymptotically unbiased even if the act of sampling affects the network's state. Such sampling is referred to as Poisson sampling. The sample size during interval T obeys Poisson distribution with rate λ at which the singleton measurements will on average be made, that is $P_k(T) = (\lambda T)k^k e^{-\lambda T} / k!$.

To generate Poisson sampling intervals, one first determines the rate λ at which the singleton measurements will on average be made (e.g., for an average sampling interval of 30 seconds, we have λ= 1/30, if the units of time are seconds). One then generates a series of exponentially-distributed (pseudo) random numbers E_1, E_2,…, E_n The first measurement is made at time E_1, the next at time E_1+E_2, and so on.

3.1.3 Stratified sampline

Before sampling, the whole population is first divided into mutually exclusive subgroups, called stratum. Let N be the number of population unit, L be the number of strata and N_1,N_2,…,N_L represent the size of each stratum, then $N = \sum_{h=1}^{L} N_h$. If the sample is taken randomly from each stratum, the procedure is known as stratified random sampling.

For stratified sampling, there are some factors supposed to determine such as stratified characteristics, stratum number, stratum border, sample size allocation and variance within strata. The concrete discussions of these factors are as follows:

Stratified characteristics are the base of stratified sampling. Network packets may be stratified by protocol type, TTL or total length field of IP header. The selection of stratified characteristics relates to the type of intrusions. As a example of ICMP sweep attack, the characteristics of this attack is the generation of lots of ping packets with light payload suddenly, then the proportion of ICMP packets ascends and the average length of packets decreases. If the packet length is selected for stratification, sound detection results are to be achieved.

From [12], the gain of stratified sampling increases with the more stratum number; when the stratum number increases to 3 from 2, the gain won't improve too much; the gain will appear to decrease with the increase of stratum number; when the stratum number is beyond 4, the gain of stratified sampling tend to be stable.

The determination of stratum border is based on the stratified strategy. Sample with similar characteristics may be classified into certain stratum in stratified sampling, which leads to the less variance in certain stratum. The simplest way to determine stratum border is based on the type of packets such as TCP, UDP and ICMP, and stratum border is naturally determined by protocol type of IP header of packet. If packet is classified into certain stratum according to the total length field, the stratum border is determined by the variable interval of the actual value L_i ($L_{\min} \le L_i \le L_{\max}$). If every stratum is based on L_i, it is obvious that this is impractical. In this paper, there are hundreds kinds of packets related to total length field. If every stratum is based on total length, the efficiency of this implementation is too bad. So we make use of the optimum stratified method based on the cumulate square root of stratified variable distribution [13], shown as Table 1.

Total length of packet	Number	$\sqrt{f}$	Cumulate $\sqrt{f}$
L_1-L_2	M_i	$\sqrt{M_1}$	$\sqrt{M_1}$
L_2-L_3	M_2	$\sqrt{M_2}$	$\sqrt{M_1}+\sqrt{M_2}$
…	…	…	…
…	…	…	…
L_{n-2}-L_{n-1}	M_{n-1}	$\sqrt{M_{n-1}}$	$\sqrt{M_1}+\sqrt{M_2}+\ldots+\sqrt{M_{i-1}}$
L_{n-1}-L_n	M_n	$\sqrt{M_n}$	$\sum_{i=1}^{n}\sqrt{M_i}$

Table 1. Cumulate square root table

Let $R=\sum_{i=1}^{n}\sqrt{M_i}$, if the stratum number is 5, then a new stratum is generated at intervals of *R/5*, and so the four stratum border point is to make the value of cumulate $\sqrt{f}$ closed to the following value: *R/5*, *2R/5*, *3R/5*, *4R/5*, and these four values are stratum border points of stratified sampling.

For stratified sampling, it is supposed to solve the problem of sample size allocation in strata given fixed population size. Because the precision of stratified random sampling relates to sample size allocation and variance within strata, the stratification or sample size allocation would affect the efficiency of stratified sampling directly. Generally, proportional allocation is the better if the means between strata vary greatly, while optimum allocation is the better if the standard deviations between strata vary greatly. In practical sample procedure, we tend to select the proportional allocation method because optimum allocation is only toward to single target variable. In fact, there are usually more target variables. The optimum allocation of single variable may be not proper to other variables. Proportional allocation refers to allocate the size within strata based on each stratum weight.

What's more, sampling strategy in a stratum is also to be determined. In practice, both systematic sampling and random sampling are to be applied.

3.2 Experiments and discussion

To test the performance of different sampling strategies, we conduct experiments in our campus network. The network traffic is captured by a host linked to the router of a C class subnet. Experimental results are shown in Figure 10 and Table 2 as follows.

	Population	Systematic sampling	Poisson sampling	Stratified sampling
UDP	82.47%	82.14%	82.85%	82.5%
TCP	11.87%	11.92%	11.94%	11.73%
ICMP	5.66%	5.95%	5.21%	5.77%

Table 2. Packet type proportion

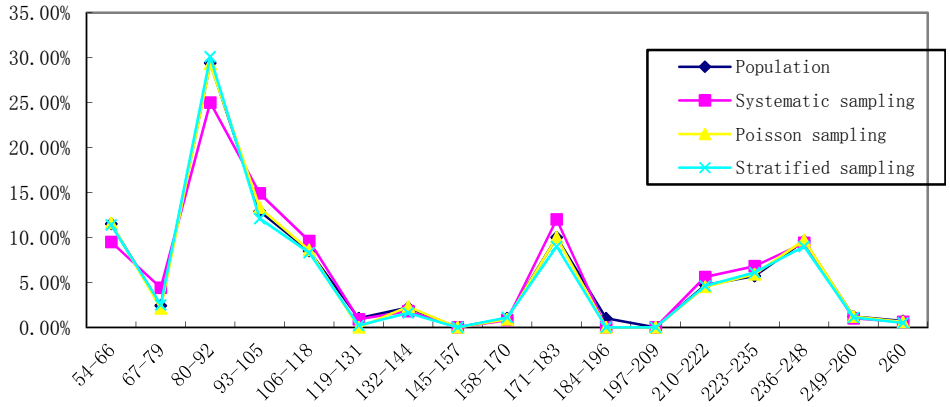


Fig. 10. Packet length distribution curve

Figure 10 is the packet length distribution curve, and table 2 shows the proportion of different type of packets in population and separate sampling strategies. From figure 1, it can be seen that the distribution of packet length in Poisson sampling and stratified sampling conforms to the distribution of packet length in population. Though there is a little diversity between the distribution of packet length in systematic sampling and the distribution of population, this diversity is also in the acceptive scope of precision. Table 2 shows the proportion of different kind of packet with diverse sampling strategies accords with population and these sampling strategies are applicable.

In practice, sampling may also be applied to intrusion detection by estimating the value of certain measure in population. The following parts briefly present the relative efforts with the average length of packets as the measure.

Assume the mean of population X is μ, variance is σ^2. Let X_1, X_2, X_3, …, Xn are the sample of population X, then $E(\overline{X}) = \mu, D(\overline{X}) = \sigma^2 / n$, and $X \sim N(\mu, \sigma^2)$, so $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ also obeys normal distribution, that is $\overline{X} \sim N(\mu, \sigma^2 / n)$ [14].

For given confidence $1-\alpha$, let X_1, X_2, X_3, …, Xn are the sample of population $N(\mu, \sigma^2)$, $\overline{X}$ and S^2 are the corresponding mean and variance of sample, then

$$\frac{\overline{X} - \mu}{S / \sqrt{n}} \sim t(n-1) \tag{1}$$

and right side distribution *t(n-1)* doesn't rely on any other parameters, then

$$P\{-t_{\alpha/2}(n-1) < \frac{\overline{X} - \mu}{S / \sqrt{n}} < t_{\alpha/2}(n-1)\} = 1 - \alpha \tag{2}$$

that is

$$p\{\overline{X} - \frac{S}{\sqrt{n}} t_{\alpha/2}(n-1) < \mu < \overline{X} + \frac{S}{\sqrt{n}} t_{\alpha/2}(n-1)\} = 1 - \alpha$$

so the confidence interval of μ with given confidence $1-\alpha$ is

$$(\overline{X} \pm \frac{S}{\sqrt{n}} t_{\alpha/2}(n-1)) \tag{3}$$

The measured value of average length of packet with systematic sampling is 134.0631, standard variance is 73.607 and total number of packet n is 92647. From the distribution table of t, the value of $t(n-1)$ appears to be a constant when n run to infinite.

We estimate the average packet length of population by systematic sampling and ensure the confidence of this length is 95%. From (3), we get the confidence interval of average packet length of population with confidence 95% is (134.0634 ± 0.47). If the error is no more than 0.94, the confidence of this error is 95% considering any value in this interval as the estimate of packet average length in population. From preceding experimental results, the value of average length of packet in population is 134.4601, which is in the interval of (134.0634 ± 0.47).

It can be seen from experiments that the average length of packet in normal traffic tends to be stable. When ICMP sweep attack appears, there are lots of packets with short length, and average length of packet is obviously various [15]. So the average length may be considered as the measure to detect intrusions.

With the change of user behavior and network topology, the characteristic of network may also vary. So the data related to the behavior of certain network during some time may be chosen to characterize the normal characteristics rather than all the passed data. Assume the granularity of time is T, L_i is the average length of packet in the *ith* time interval. Anomaly actions are to be detected by comparing current sampled value and the value of preceding *i-1th* time interval. That is, if current average length of packet is in the scope of normal value calculated by preceding sampled data, there is no intrusion, and then the preceding data can be updated. Otherwise, there appears intrusion, and current data can't be updated.

3.3 Summary

With the rapid development of network technology, there are more severe challenges to information security, and IDS has been an indispensable part of computer security. However, there appears packet drop for IDS especially in a high speed network environment. In this chaper, we apply packet selection model based on sampling methods of statistics to the procedure of data collection of IDS. Experiment results show that selected sample (packets) can be applied to detection and analysis for IDS in the scope of certain precision. In short, our method has the following advantages: firstly, this method exceedingly strengthens the processing performance of IDS by the means of replacing dropping packets passively with sampling packets actively especially in the large-scale high-speed network; secondly, this method has better expansibility, and various sampling strategies may be applied corresponding to different implementation.

4. STAMP -A high-speed traffic collection platform based on sampling on FPGAs

4.1 Design and implementation of STAMP

In this paper, we describe the design and implementation of a uniform high-speed traffic collection platform for intrusion detection/prevention based on sampling on FPGAs. To achieve this goal, HSTCP's architecture integrates elephant flow identification and adaptive elephant flow sampling into a FPGA prototyping board, which is a gigabit Ethernet network interface card with open hardware and software specifications.

A flow is a sequence of packets that share certain common properties (called flow specification) and have some temporal locality as observed at a given measurement point. Depending on the application and measurement objectives, flows may be defined in various manners such as source/destination IP addresses, port numbers, protocols, or combinations thereof. They can be further grouped and aggregated into various granularity levels such as network prefixes or autonomous systems. In this paper, we present flow statistics and experimental results using flows of 5 tuple (source/destination IP addresses, port numbers, and the protocol number) with a 60-s timeout value as our basic flow definition.

As many measurement-based studies have revealed, flow statistics exhibit strong heavy-tail behaviors in various networks (including the Internet). This characteristic is often referred to as the elephant and mice phenomenon (aka the vital few and trivial many rule), i.e., most flows (mice flows) only have a small number of packets, while a very few flows (elephant flows) have a large number of packets. A noticeable attribute of elephant flows is that they contribute a large portion of the total traffic volume despite being relatively few in the number of flows. In this paper, we define an elephant flow as a flow that contributes more than 0.1% of all unsampled packets.

The elephant flow identification module maintains an array of counters for every flow. Counters at certain index would contain the total number of packets belonging to all of the flows colliding into this index.

At intervals of certain time (60 s), the adaptive elephant flow sampling module would adjust the sampled rate according to the traffic load changes in the identified elephant flow. The sampled rate is based on the packet count. An AR model is used for predicting the number of packets of a certain elephant flow in the next time interval.

HSTCP is built on the Avnet Virtex-II Pro Development Board shown in Figure 11. This FPGA prototyping board includes all of the components necessary for a gigabit Ethernet network interface with embedded processors and on-board memory.

Fig. 11. HSTCP PCI card

4.1.1 Elephant flow identification

Identifying elephant flows is very important in developing effective and efficient traffic engineering schemes. In addition, obtaining the statistics of these flows is also very useful for network operation and management. On the other hand, with the rapid growth of the link speed in recent years, packet sampling has become a very attractive and scalable means to measure flow statistics.

To identify elephant flows, traditionally we have to collect all packets in the concerned network, and then extract their flow statistics. As many previous studies have indicated, however, such an approach lacks scalability. For very high speed links (say, OC-192+), directly measuring all flows is beyond the capability of measurement equipments (i.e., the requirements for CPU power, memory/storage capacity, and access speed are overwhelming).

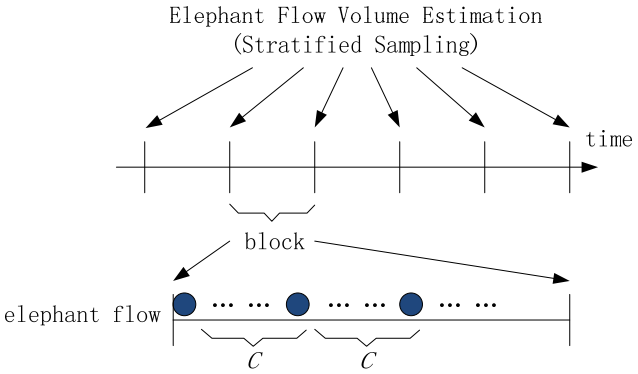


Fig. 12. Elephant flow volume estimation.

Because flows are dynamic in their arrival time and active duration, it is very hard to define a sampling interval that is valid for all elephant flows, while allowing us to adjust the sampling rate in accordance with the changing traffic condition to ensure estimation accuracy. We tackle this problem by using stratified sampling. Sequenced predetermined, nonoverlapping time blocks are called strata. In each block, systematic count - based sampling is applied, that is every Cth packet of the parent process is deterministically selected for sampling, starting from some starting sampling point, and other packets will be directly dropped. At the end of each block, flow statistics are estimated. Then, naturally, a flow's volume is summarized into a single estimation record at the end of the last time block enclosing the flow. Notice that from each flow's point of view, its duration is divided or stratified in a fixed time. The predetermined time blocks enable us to estimate the flow volume without knowing dynamic flow arrival times and their durations while adjusting the sampling rate according to dynamical traffic changes.

The elephant flow identification module maintains an array of counters. Upon the arrival of a packet, its flow specification is hashed to generate an index into this array, and the counter at this index is incremented by 1. Collisions due to hashing might cause two or more flow labels to be hashed to same indices. Counters at such an index would contain the total

number of packets belonging to all of the flows colliding into this index. We do not have any explicit mechanisms to handle collisions as any such mechanism would impose additional processing and storage overheads that are unsustainable at high speeds. This makes the encoding process very simple and fast. Efficient implementations of hash functions allow the online streaming module to operate at speeds as high as OC-768 (40 Gbps) without missing any packets.

Internet traffic is known to have the property that a few flows can be very large, while most other flows are small. Thus, the counters in our array need to be large enough to accommodate the largest flow size . On the other hand, the counter size needs to be made as small as possible to save precious SRAM. Recent work on efficient implementation of statistical counters provides an ideal mechanism to balance these two conflicting requirements, which we will leverage in our scheme. For each counter in the array, say 32 bits wide, this mechanism uses 32 bits of slow memory (DRAM) to store a large counter and maintains a smaller counter, say 7 bits wide, in fast memory (SRAM). As the counters in SRAM exceed a certain threshold value (say 64) due to increments, it increments the value of the corresponding counter in DRAM by 64 and resets the counter in SRAM to 0. There is a 2-bit per counter overhead that covers the cost of keeping track of counters above the threshold, bringing the total number of bits per counter in SRAM to 9. For suitable choices of parameters, this scheme allows an efficient implementation of wide counters using a small amount of SRAM. This technique can be applied seamlessly to implementing the array of counters required in our data streaming module. In our algorithm 6, the size of each counter in SRAM is 9 bits and in DRAM is 32. Also, since the scheme in [23] incurs very little extra computational and memory access overhead, our streaming algorithm running on top of it can still achieve high speeds such as OC-768.

4.1.2 Adaptive elephant flow sampling

Traffic measurement and monitoring serves as the basis for a wide range of IP network operations and engineering tasks such as troubleshooting, accounting and usage profiling, routing weight configuration, load balancing, capacity planning, etc. Traditionally, traffic measurement and monitoring is done by capturing every packet traversing a router interface or a link. With today's high-speed (e.g., gigabit or terabit) links, such an approach is no longer feasible due to the excessive overheads it incurs on line-cards or routers. As a result, packet sampling has been suggested as a scalable alternative to address this problem. In this paper, we have investigated two sampling techniques, namely, simple random packet sampling and adaptive weighted packet sampling.

Given the dynamic nature of network traffic, static sampling does not always ensure the accuracy of estimation, and tends to oversample at peak periods when efficiency and timeliness are most critical. More generally, static random sampling techniques do not take traffic dynamics into account; thus they cannot guarantee that the sampling error in each block falls within a prescribed error tolerance level.

In other words, under some traffic loads, static count-based sampling may be poorly suited to the monitoring task. During periods of idle activity or low network loads, a big sampling count provides sufficient accuracy at a minimal overhead. However, bursts of high activity require a small sampling count to accurately measure the network status at the expense of

increased sampling overhead. To address this issue, adaptive sampling techniques can be employed to dynamically adjust the sampling count and optimize accuracy and overhead.

In this paper, we investigated adaptive sampling techniques to intelligently sample the incoming elephant flows. A key element in adaptive sampling is the prediction of future behavior based on the observed samples. The AR model described in this section predicts the packet count of the next sampling interval based on the past samples. Inaccurate predictions indicate a change in the elephant flow load and require an increased/decreased sampling count to determine the new value.

In any case, we cannot accurately choose the sampling count when the population size (total packet count of the observation time block) is unknown. We can compute the sampling probability at the beginning of a block by predicting the total packet count of a certain elephant flow. We employ an AR model for predicting the total packet count m_h^f of the hth block of elephant flow f, as compared to other time series models, since it is easier to understand and computationally more efficient. In particular, using the AR model, the model parameters can be obtained by solving a set of simple linear equations, making it suitable for online implementation.

We will now briefly describe how the total packet count m_h^f of the hth block of elephant flow f can be estimated, based on the past packet counts using the AR(u) model, where u is the lag length. Using the AR(u) model, m_h^f can be expressed as

$$m_h^f = \sum_{i=1}^{u} a_i m_{h-i}^f + e_h$$

where a_i, i=1, …, u, are the model parameters, and e_h is the uncorrelated error (which we refer to as the prediction error).

The model parameters a_i, i=1, …, u, can be determined by solving a set of linear equations in terms of v past values of m_i^f's, where $v \geq 1$ is a configurable parameter independent of u, and is typically referred to as the memory size.

Let $\hat{m}_h^f$ denote the predicted packet count of the hth block of elephant flow f. Using the AR(u) prediction model, we have

$$\hat{m}_h^f = \sum_{i=1}^{u} a_i m_{h-i}^f \ .$$

Using the AR prediction model, at the end of each block, the model parameters (a_i) are computed. The complexity of the AR prediction model parameter computation is only $O(v)$ where v is the memory size.

The predicted $\hat{m}_h^f$ is then compared with the actual value of the sample m_h^f. A set of rules is applied to adjust the current sampling count, $\Delta C_{\text{curr.}}=c(h)-c(h-1)$, to a new value, ΔC_{new}, which is used to adjust the sampling count to compare the rate of change in the predicted sample value, $\hat{m}_h^f - m_{h-1}^f$, to the actual rate of change, $m_h^f - m_{h-1}^f$. The ratio between the two rates is defined as R, where

$$R = \left| \frac{\hat{m}_h^f - m_{h-1}^f}{m_h^f - m_{h-1}^f} \right| .$$

The value of R will be equal to 1 when the predicted behavior is same as the observed behavior. We define a range of values $R_{MIN} \le 1 \le R_{MAX}$, such that

if $R < R_{MIN}$, that is $\Delta C_{New} < \Delta C_{Curr.}$, then

$$\Delta C_{New} = \lfloor (R) \times \Delta C_{Curr.} \rfloor .$$

If $R_{MIN} < R < R_{MAX}$, then

$$\Delta C_{New} = 2 \times \Delta C_{Curr.}$$

If $R > R_{MAX}$, that is $\Delta C_{New} < \Delta C_{Curr.}$, then

$$\Delta C_{New} = \lceil (1 + R) \times \Delta C_{Curr.} \rceil .$$

Otherwise,

$$R_{undefined} \Rightarrow \Delta C_{New} = 2 \times \Delta C_{Curr.}$$

4.1.3 FPGA implementation

HSTCP is built on the Avnet Virtex-II Pro Development Board, and its architecture satisfies the constraints of the Avnet board while efficiently integrating the components of a gigabit Ethernet network interface. The MAC unit, DMA unit, inter-FPGA bridge, hardware event management unit, and PCI interface were custom designed for HSTCP. The remaining components were provided by Xilinx and used with little or no modification. Both the MAC unit and the PCI interface are built around low-level interfaces provided by Xilinx; however, those units are still mostly custom logic to integrate them into the rest of the system and to provide flexible software control over the hardware functionality.

The Xilinx Virtex-II Pro FPGA on the Avnet development board contains most of the NIC logic, including the PowerPC processors, on-chip memories, MAC controller, DMA unit front-end, and DDR memory controller. The smaller Spartan-IIE FPGA contains the PCI controller, the back-end DMA controller, and a SRAM memory controller. The SDRAM, although connected to a shared data bus between the Spartan and Virtex FPGAs, was not used because the entire bus bandwidth was needed to efficiently use the PCI interface.

To save development time, prebuilt Xilinx cores were used for several of the hardware modules, including the PCI interface, DDR controller, and a low-level MAC. However, these cores cannot be connected directly to form a working NIC. For example, although Xilinx provides a PCI core, it must be wrapped within a custom DMA unit to allow the PowerPC to initiate and manage high-performance burst transfers to/from the host system across the FPGA bridge. Similarly, although Xilinx provides a low-level MAC core, it must be outfitted with an advanced descriptor-based control system, data buffers, and a DMA unit to transfer

packet data between NIC memory and the PHY. Finally, the DDR controller required modifications to function in this specific development board with its unique wiring.

The processors, memories, and hardware units are interconnected on the Virtex FPGA by a processor local bus (PLB). The PLB is a high-performance memory-mapped 100-MHz 64-bit-wide split-transaction bus that can provide a maximum theoretical bandwidth of 12.5 Gbits/s in a full duplex mode. The PLB allows for burst transmissions of up to 128 bytes in a single operation, which is used by the DMA and MAC units to improve memory access efficiency.

Also attached to the PLB is a small memory-mapped control module used to route descriptors to or from the MAC and DMA hardware assist units. This module also provides a central location for hardware counters, event thresholds, and other low-level control functions. By attaching this central control unit to the PLB, either PowerPC processor can manipulate these important NIC functions. The control unit takes 18 PowerPC cycles to read and 12 cycles to write a 32-bit word, primarily due to bus arbitration delays.

4.2 Experiment

We evaluated HSTCP using a synthetic dataset that was generated by combining the data from the 1999 DARPA intrusion detection project and 2000 DARPA "Scenario Datasets" that have been crafted to provide examples of multiple component attack scenarios instead of the atomic attacks as found in past evaluations [24] and Münchner Wissenschaftsnetz (MWN), Germany. The MWN provides Internet connectivity to two major universities and a number of research institutes. Overall, the network contains about 50,000 individual hosts and 65,000 registered users. The trace "mwn-cs-full" that we analyzed is a 2-h trace including the full payload of all packets to/from one of the CS departments in MWN, with some high-volume servers excluded.

The 1999 DARPA dataset that we used consisted of 5 weeks of TCPdump data. Weeks 1 and 3 have normal attack-free network traffic. Week 2 consists of network traffic with labeled attacks, while weeks 4 and 5 contain 201 instances of 58 different attacks, 177 of which are visible in the TCPdump data. The 2000 DARPA dataset includes two recently created scenario datasets that address the needs of mid-level correlation systems. Each includes several hours of background traffic and a complete attack scenario. Attacks and background traffic were run on the same testbed used in the 1999 evaluation, but with the addition of a commercial off-the-shelf firewall and demilitarized zone (DMZ) network separating the Internal and Internet networks and a Solaris 2.7 victim host.

The experimental evaluation of HSTCP has been divided into three steps. In section 4.1, we evaluate the performance of the sampling algorithm and compare its performance with the stratified random sampling algorithm. Then in section 4.2 we evaluate the performance of HSTCP for intrusion detection/prevention.

4.2.1 Evaluation of the sampling algorithm

Experiments were conducted to compare and evaluate the performance of the proposed adaptive sampling algorithm with the stratified random sampling algorithm.

In 1994, Leland *et al*. showed that the Ethernet traffic consisted of slowly decaying packet count bursts across all time scales. Time series that consist of such a pattern are said to exhibit the property of long-range dependence and are termed as "self-similar." Similar self-similar behavior has also been observed in wide-area Internet traffic by other researchers . One important characteristic of a self-similar process is that its degree of self-similarity can be expressed with a single parameter, namely the Hurst parameter which can be derived from the rescaled adjusted range (R/S) statistic. It is defined as follows.

For a given set of observations X1, X2, …, Xn with a sample mean $\bar{X}(n)$ and sample variance $S^2(n)$, the rescaled adjusted range or the R/S statistic is given by

$R(n)/S(n) = 1/S(n)\left[\max(0, W_1, W_2, ..., W_n) - \min(0, W_1, W_2, ..., W_n)\right]$, with

$W_k = (X_1 + X_2 + \cdots X_k) - k\bar{X}(n)$, k=1, 2, …, n. Hurst (1955) found that many naturally occurring time series appear to be well represented by the relation $E\left[R(n)/S(n)\right] \approx \alpha_4 n^H$, as $n \to \infty$, with Hurst Parameter H "typically" about 0.73. On the other hand, if the observations Xk come from a short-range dependent model, then Mandelbrot and Van Ness (1968) showed that $E\left[R(n)/S(n)\right] \approx \alpha_5 n^{0.5}$, as $n \to \infty$. This discrepancy is generally referred to as the Hurst effect or Hurst phenomenon.

It is very important whether the traffic data sampled by the proposed sampling scheme retains the self-similar property for various anomaly detection techniques, which may directly affect the accuracy and efficiency of detection. So we verify this based on two different parameters: the mean of the packet count and the Hurst parameter. The peak-to-mean ratio (PMR) can be used as an indicator of traffic burstiness. The PMR is calculated by comparing the peak value of the measure entity with the average value from the population. However, this statistic is heavily dependent on the size of the intervals, and therefore may or may not represent the actual traffic characteristic. A more accurate indicator of traffic burstiness is given by the Hurst parameter. The Hurst parameter (*H*) is a measure of the degree of self-similarity. In this paper we use the R/S statistical test to obtain an estimate for the Hurst parameter. We run the test on both the original and the sampled data.

In our sampling scheme, simple random sampling is conducted in every time block (strata) and this refers to stratified random sampling.

In Figures 3 and 4, we show the average sampling error for the Hurst parameter and the sample mean, respectively. As one can see in Figure 13, the stratified random sampling algorithm resulted in a higher average percent error for the Hurst parameter when compared to adaptive sampling. This could be the result of missing data spread out over a number of sampling intervals. In Figure 14, the average percentage error for the mean statistic was marginally lower for our sampling algorithm when compared with the stratified random sampling algorithm, albeit the difference was insignificant. One possible reason for this marginal difference is the inherent randomness nature of stratified random sampling algorithm—i.e., the weighted mean packets are nicely sampled randomly, which results in good estimation of mean.

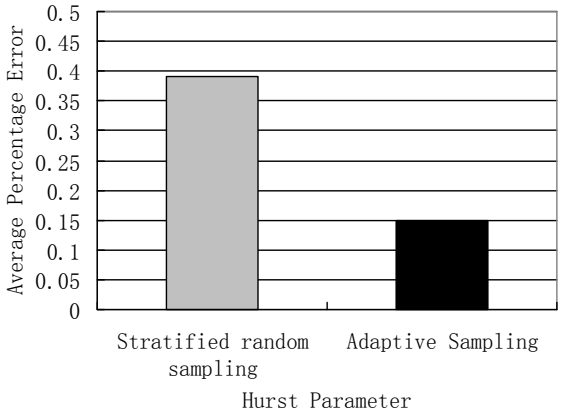


Fig. 13. Average percentage error for the Hurst parameter.

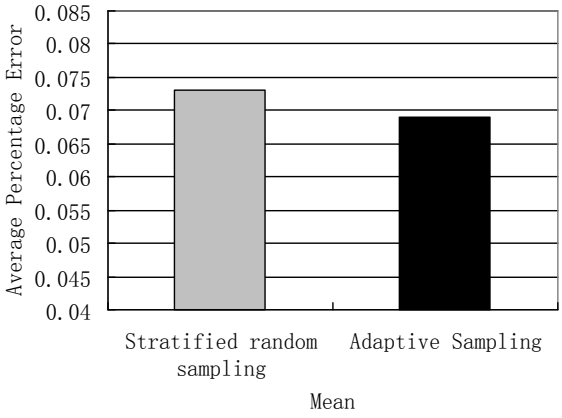


Fig. 14. Average percentage error for the mean statistic.

4.2.2 Evaluation of the platform for intrusion detection/prevention

The IXIA1600T (a special network test facility of IXIA Crop) is used to transmit the synthetic dataset in 1000M Ethernet. The ISS RealSecure gigabit network is selected to detect attacks (intrusions), and HSTCP is used for collecting network traffic for it. As the first commercial IDS, RealSecure has been playing an important role in this field, and it provides network intrusion detection and response capabilities that monitor the gigabit network.

The ROC (receiver operating characteristic) technique is used to evaluate the detection effect of ISS RealSecure. The ROC approach analyzes the tradeoff between false alarm and detection rates for detection systems. It was originally developed in the field of signal detection. More recently, it has become the standard approach to evaluate IDSs. In this paper, we mainly take into account the detection rates of RealSecure when its false alarm rate is under 0.2. Too many false alarms will make administrators consume unnecessary time and energy analyzing these alarms, which compromises the usability and validity.

In Figure 15, we can see that HSTCP was able to cope with the 1-Gbps network traffic, and elephant flow sampling was not initiated. The detection rates of RealSecure remarkably increased during initial phases, and then tended to be stable.

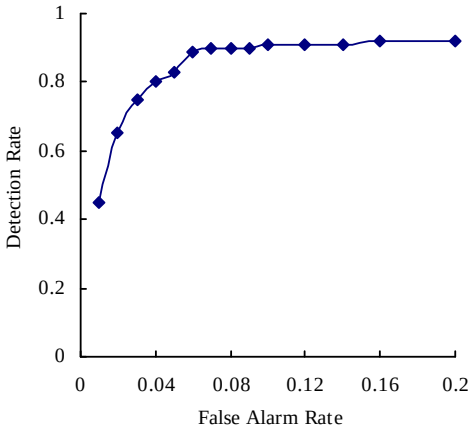


Fig. 15. The ROC curves at 1-Gbps speed.

To evaluate the performance of HSTCP for intrusion detection in the high-speed network, we used IXIA1600T to transmit the synthetic dataset at a 10-Gbps speed. Under this circumstance, HSTCP could not capture the whole traffic and tended to drop packets. In Figure16, we can see that the detection rates sharply decline without sampling. While HSTCP initiated elephant flow sampling methods to cope with the network traffic, RealSecure could still make response to the high-speed traffic. With the increase in false alarm rates, detection rates remained high and stable.

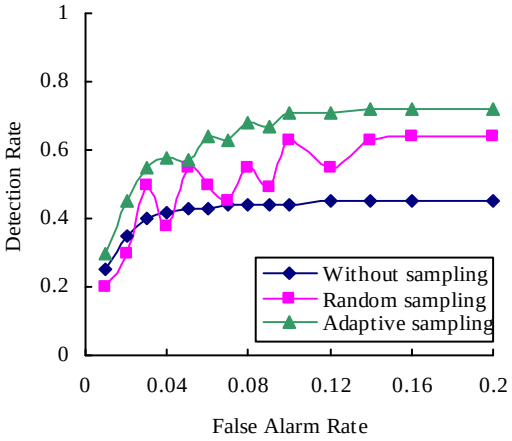


Fig. 16. The ROC curves at a 10-Gbps speed.

We compared the performance of the adaptive sampling technique and the random sampling technique. As expected, the adaptive sampling technique showed superior performance.

4.3 Summary

In this subsection, we have presented a uniform high-speed traffic collection platform for intrusion detection/prevention based on sampling on FPGAs—called HSTCP—that has the ability to cope with very high speed network traffic (even Tbps). By employing complete mice flows' capture and adaptive elephant flow sampling, HSTCP effectively reduces the volume of network traffic for intrusion detection/prevention without losing its intrinsic characteristics. In addition, HSTCP provides a flexible and scalable platform for network IDSs/IPSs faced to the challenge of future high-speed networks.

5. The research of trusted communication protocol for intrustion prevention system

5.1 IPS trusted communication mechanism

A trusted communication mechanism proposed in this chapter applied to the correlation between firewall and IDS in a distributed intrusion prevention system. It is mainly based on middleware technology and security Protocol standard techniques. And the middleware technology is CORBA. If CORBA underlies network layer, it may encapsulate the underlying unit; which may make the application transparent to the up-layer. In this paper, TLS is applied to the trusted data transmission between firewall and IDS.

5.2 The design of IPS trusted communication protocol

The design of the mechanism of trusted data communication is illustrated by figure 17:

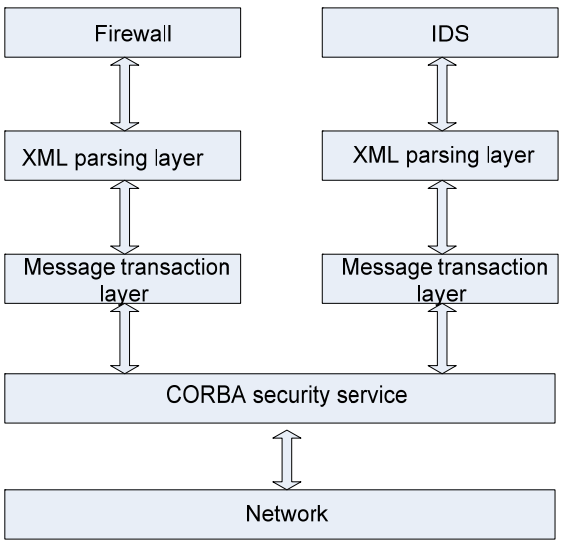


Fig. 17. The mechanism of trusted communication for IPS

Application Layer

The application layer mainly refers to Client and Server; in this paper it represents firewall and IDS. During the transmission, both of the firewall and IDS can be client or server. Firewall and IDS system are all in the network layer. Firewall provides the data source and a place to process the final data while IDS is responsible for receiving data request from firewall and analyzing them, and then return the processed results to the firewall.

XML parsing layer

This layer primarily encapsulates and analyzes the communicated data.

Message transaction layer

In this paper, a security protocol called TLS (Transport Layer Security) applied to supporting the security and reliability for data communicate among each other. It also may protect the privacy of the applications and users for network communication. When server and client are communicated, TLS could make sure important messages won't be sniffered or stolen by the third party. It is a successor protocol followed by SSL.

CORBA security service

CORBA security service (CORBASec) is an important public object service in CORBA. It constructs secure language environment between client objects and service objects, and also provides better security service [10].

5.3 Design of data exchange format based on XML

In a distributed intrusion prevention system, data filter module is composed of a firewall and other components. Network data processing module generally refers to IDS. In this paper, data transmitted between firewall and IDS are classified into four categories: event data, rule data, analysis result data and actions response data, and it is referred to the function units in CIDF framework [11].

The relationship among above data: Data generated by the firewall with network packets filtered called event data. Whether the event described is an intrusion event, it depends on the match or analysis of IDS based rule data. If it was a real intrusion event, then generated the analysis result data. The firewall will make a response to the analysis result data based on the corresponding strategies, and generates action response data.

Design of event data

Data filtered original network packets by firewall according to security strategy is event data. Therefore, this kind of data must contain the complete description of network original data that IDS can detect or analyze by those matching information in rules. In addition, it needs to contain the firewall name and the time of event happened. The reason of including firewall name is that more firewalls may be deployed in network. When detecting the network data, more than one firewall will find the same event, so we need to distinguish and analyze. In some cases, IDS need to detect what happened during some phases and then ensure whether any intrusions had happened, so the data type also contains the time of event happened. At last, to support data expansion, we need additional data applied to describing some additional description in event data, and it can be used as a reserved interface. The illustration of event data is as follows:

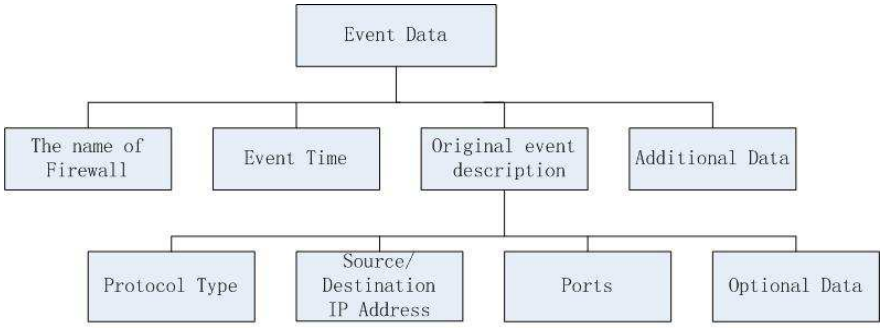


Fig. 18. The design of event data

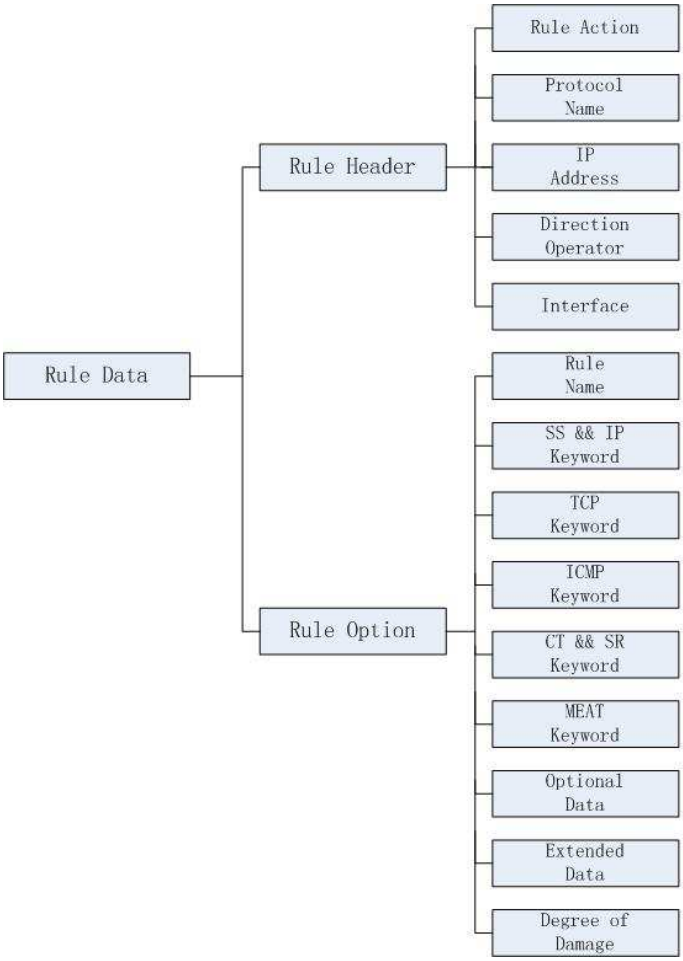


Fig. 19. The design of rule data

The XML DTD of event data is provided in details as follows, but other types are ignored.

Design of rule data

According to snort rules, the rule data divided into two logic parts: Rule header and rule options. The rule header contains rule actions, protocols, source and destination IP Address and mask, source or destination ports and some direction operators. The rule option contains some alert messages and the main parts of checked packets; it includes the characteristics and the priority. And the part of characteristics should be described by one or more keywords. Same as the event data, rule data also need contain the reserved interface applied to the expansion of rule data, and then put them into rule options. The design of rule data is shown in Figure.3.

Design of analysis results data

To analyze the data delivered to IDS and the rule data of IDS then get the analysis results data. After the detection analysis by IDS, the event can be classified into three categories: a malicious attacked event, a security event being suspected but not confirmed and normal network traffic. To the malicious attacks event, we use the obstructive response methods and record this event; it needs some alert response methods to those being suspected but not confirmed security events. Just writing down the key information to the normal network traffic is enough. However, none of these will be directly solved in IDS. IDS will put the analysis results data through the central control module and match it, then feedback to firewall if it was allowed, and firewall will tackle them according to the response actions. So the analysis results data should describe the event type, event processing action, time of the event, which IDS dealt with the event and so on. If it was the malicious or suspected event, we also need to describe the name of event, the source of event, the target of event, when attack happened and how it did affect the system. The same as the first two data structure, we also need to give the analysis results data a reserved interface which is called expansion data. The design of analysis results data is shown in Figure.20.

Design of action response data

According to response actions, firewall deal with the results data detected by IDS and gets the action response data. During this, the normal network traffic don't need to be recorded, whereas put the malicious or suspected events and corresponding processing actions into the record, thus the action response data only defined to IDS. So the action response data should present which firewall has been responded to the analysis data? What is attack event called? What is the source from? What is the target? When has it happened? And how does it compromise the system? All of these relate to the corresponding attack events in the analysis result events. These records used to keep the respond data integrity. The action response data should also record the final processing results analyzed by firewall to the results data, which is called response action. In addition, the receiver should be provided either. At last, including the extended data and the implementation of data extended. The design of action response data is shown in Figure.21.

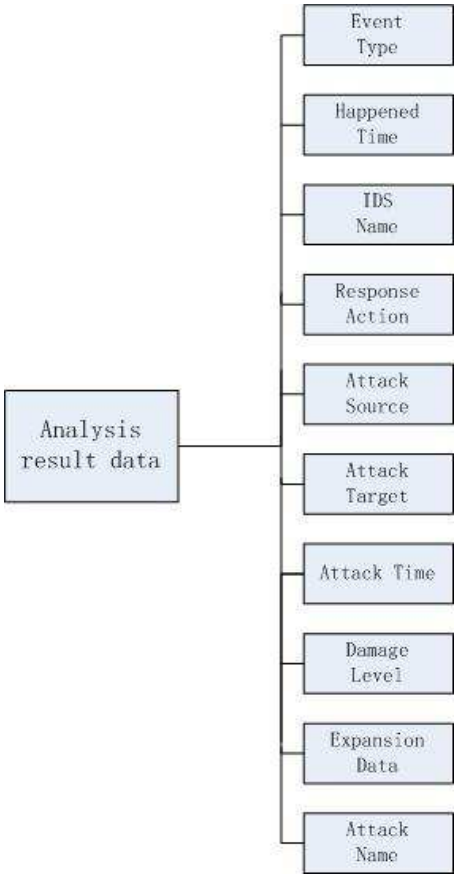


Fig. 20. The design of analysis results data

5.4 Related works

IDXP (Intrusion Detection Exchange Protocol) is an application layer protocol, which is applied to the data exchange among intrusion detection entities (Feinstein & Matthews, 2007), and it may support the transformation of the IDMEF (Intrusion Detection Message Exchange Format) message, unstructured text and binary dataset (Debar et al., 2007). Again, it has the security characteristics of bidirectional authentication, integrity and confidentiality based on connection-oriented protocols. However, IDXP and IDMEF are simply data exchange protocols, and they aren't adaptive to other correlation schemes.

IAP (Intrusion Alert Protocol)[9] is a transport protocol applied to the alert data of intrusion detection, which is based on TCP protocol and TLS protocol used for the secure data transmission (Gupta et al., 2001). Alert data is described in XML, and it conforms to the format specification of IDMEF.

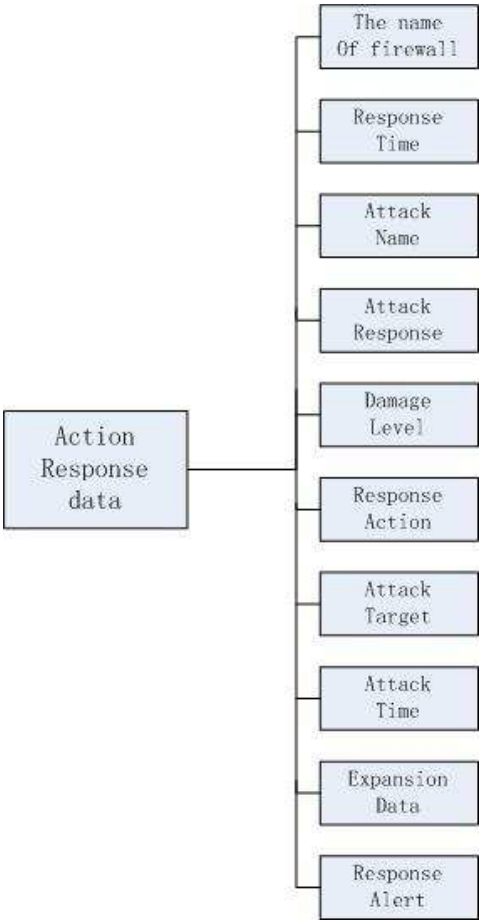


Fig. 21. The design drawing of action response data

IPIEP (Intrusion Protection Interaction Exchange Protocol) is an application layer protocol applied to the definition of exchange rules for correlation messages, while IPIMEF (Intrusion Protection Interaction Message Exchange Format) provides the definition of data format for message, and they don't have corresponding dependent relationship.

Above all, all of these related works don't provide the support to the trusted communication transmission between data filter module and network data process module in a distributed intrusion prevention system.

5.5 Summary

The main contribution of this subsection is to provide a trusted communication protocol between data filter module and network data processing module in a distributed IPS. The transmit data can be classified into four categories with XML technology defined respectively, and XML DTD description language is applied to the definition of these four

data formats, and then a data trusted communication mechanism is provided, which divided into three layers: application layer, XML parsing layer and message transaction layer. CORBA technology is applied to solving the heterogeneous platform and load-balancing problems. TLS security protocol standard is to ensure the integrity and security during data transmission. The trusted communicated protocol also can be adaptive to network security products and network management equipments, and it contributes to security data fusion and detecting sophisticated distributed network attacks.

6. Conclusion

With the rapid development and comprehensive application of network technology, Internet significantly contributes to the development of the human society. Meanwhile, network security problems gradually appear serious. Traditional firewall technologies can't provide sufficient security protection against various attacks and intrusions, while intrusion detection systems (IDS) are faced with compromise between false alarms and false positives, so intrusion prevention system (IPS) come into being. IPS may block malicious attack traffic before corresponding intrusions cause more severe damage other than simply generate intrusion alarms.

In this chaper, we investigate intrusion detection and intrusion prevention in high speed network and the main research work is as follows:

1. Based on the investigation on the recent trends of network security techniques, such as firewall and IDS, we propose a intrusion prevention scheme based on the correlation between IDS and firewall. This scheme complements the fundamental flaws of IDS and firewall, and it may provide real-time, active prevention and attempts to stop attacks, which contributes to normal transmission of legal network traffic. In this paper, we present the design and implementation of a prototype system of network IPS —— DXIPS, based on the correlation between Snort_inline and Netfilter configured by IPtables. The hierarchical architecture of this system includes intrusion prevention layer, server layer and control layer, in which intrusion prevention layer monitors the traversing traffic and conducts intrusion detection and prevention; server layer collects log data and translate them into readable formats; control layer is administrational console and perform data display. The system is design with modularization, which includes intrusion prevention module, log recording module, central control module and communication module, and the concrete implementations of these modules are presented. The deployment policies are discussed according to various applications environment. Netfilter is a built-in firewall in the kernel of Linux, which belongs to the latest fifth generation firewall. It has the capability to directly filter malicious packets in the TCP/IP stack in kernel, which improves the response performance. What's more, DXIPS provides better scalability according to various applications environment.
2. Data collection mechanism is a key factor that affects the performance of IDS/IPS. The most current products execute per-packet detection. However, with the development and widespread of high speed networking technique, the application of IDS/IPS has been faced with serious challenges. In this paper, the sampling technique in statistics is introduced into the procedure of data collection for IDS/IPS, and the new data collection module based on sampling is proposed. Three typical sampling strategies, such as systematic sampling, Poisson sampling and stratified sampling, are applied to

network traffic collection. The packet length and type serve as the measure of anomaly detection, and simulation results show that the sample traffic is still characterized as the whole network traffic, and it may provide efficient data source for anomaly detection with the lower overhead. In a short, this method exceedingly strengthens the processing performance of IDS/IPS by the means of replacing dropping packets passively with sampling packets actively with the minor degradation of detection rates, and may improve resistant to Denial of Service attacks.

3. With the ever increasing deployment and usage of gigabit networks, traditional networks Intrusion Detection/Prevention Systems (IDS/IPS) have not scaled accordingly. More recently, researchers have been looking at hardware based solutions that use FPGA's to assist network IDSs/IPSs, and some proposed systems have been developed that can be scaled to achieve a high speed over 10Gbps. However, these solutions available have inherent limitations and unable to be applied to future high speed network (Tbps). In this paper, we present a scalable traffic sampling platform for intrusion detection/prevention on FPGA, called STAMP. The methodology is when the proposed platform is unable to capture the whole network traffic; it will initiate elephant flow sampling other than merely randomly dropping packets. Meanwhile, sampling rate is adaptive to the traffic load of elephant flow. All the captured packets are forward from STAMP to IDS via PCI bus. The noteworthy features of STAMP include: it takes the self similarity of network traffic into account with the attempts to collect malicious traffic, and improve the efficiency of network traffic sampling for IDS/IPS; it employs adaptive elephant flow sampling (AEFS) to retain inherent characteristics of network traffic, which contributes to anomaly detection; it provides a flexible and scalable platform for network IDSs/IPSs that will be faced the challenge of future high-speed network.
4. To achieve the secure and reliable transmission for the interactive data between IDS and firewall, the concept of trusted communication is introduced in this paper. We give the design and implementation of a trusted communication protocol based on XML. The design and implementation of trusted communication mechanism between firewall and IDS is presented considering each functional unit of common intrusion detection framework. The CORBA middleware is applied to data transmission, and TLS secure protocol is applied to trusted transmission between IDS and firewall. The hierarchical architecture of this protocol includes application layer, XML resolution layer and message transaction layer, in which application layer consists of client and server used to capture and analyze packets; XML resolution layer translates the data into uniform XML format and provide the base for data exchange; message transaction layer employs TLS security protocol to achieve secure and trusted communication. The data type between IDS and firewall of the proposed prototype system is composed of event data, rule data, analysis result data and response action data, and the concrete descriptions of these data based on XML DTD are also provided. The proposed trusted communication protocol has the scalability to support various network security products (such as firewall, IDS, IPS, etc.) and management facilities, and may contribute to the data fusion of these facilities and detect sophisticated distributed network attacks.

7. Acknowledgment

The authors would like to thank the editors and anonymous reviewers for their valuable comments. This work was supported in part by the National Grand Fundamental Research

973 Program of China under Grant No. 2009CB320706, the National High Technology Research and Development Program of China under Grant No. 2011AA010101, the National Natural Science Foundation of China under Grant No. 61103197 and 61073009, Program of New Century Excellent Talents in University of Ministry of Education of China under Grant No. NCET-06-0300, the Youth Foundation of Jilin Province of China under Grant No. 201101035, and the Fundamental Research Funds for the Central Universities of China under Grant NO.200903179.

8. References

Anderson, J.P. (1980). Computer Security Threat Monitoring and Surveillance, Technical report, James P Anderson Corporation: Fort Washington, Pennsylvania

Allen, J.; Christie, W.; Fithen, et al. (2000). State of the practice of intrusion detection technologies, S Technical report: CMU/ SEI2992TR2028, Software Engineering Institute, Carnegie Mellon University

Denning, D. E. (1987). An Intrusion Detection Model. *IEEE Transactions on Software Engineering*, Vol.13, No.2, (1987), pp. 222-232, ISSN: 0098-5589

Drinic, M.; & Kirovski, D. (2004). A hardware-software platform for intrusion prevention, *Proceedings of 37th annual IEEE/ACM international symposium on microarchitecture*, pp: 233-242, ISBN 0-7695-2126-6, Oregon, Portland, USA, 2004,

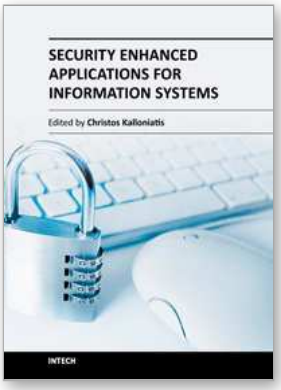
Weaver, N.; Paxson V.; & Gonzalez, J.M. (2007). The shunt: an FPGA-based accelerator for network iintrusion prevention, *Proceedings of 2007 ACM/SIGDA 15th international symposium on field programmable gate arrays*, pp:199-206, ISBN 978-1-59593-600-4, Monterey, California, USA, 2007

Uppuluri, P.; Joshi U.; & Ray, A. (2005). Preventing race condition attacks on file-systems, *Proceedings of 2005 ACM symposium on applied computing*, pp:346-353, ISBN 978-1-58113-964-8, Santa Fe, New Mexico, USA, 2005

Feinstein, B. & Matthews, G. (2007). The Intrusion Detection Exchange Protocol (IDXP), IETF RFC 4767, 24.10.2011, Available from http://www.ietf.org/rfc/rfc4767.txt, 2007

Debar, H.; Curry, D. & Feinstein, B. (2007). The Intrusion Detection Message Exchange Format (IDMEF), IETF RFC 4765, 24.10.2011, Available from http://www.ietf.org/rfc/rfc4765.txt, 2007

Gupta, D.; Buchheim, T.; Matthews, G. et al. (2001). IAP: Intrusion Alert Protocol IETF IDWG Draft, 24.10.2011, Available from http://tools.ietf.org/html/draft-ietf-idwg-iap-05, 2001.



Security Enhanced Applications for Information Systems

Edited by Dr. Christos Kalloniatis

ISBN 978-953-51-0643-2
Hard cover, 224 pages
Publisher InTech
Published online 30, May, 2012
Published in print edition May, 2012

Every day, more users access services and electronically transmit information which is usually disseminated over insecure networks and processed by websites and databases, which lack proper security protection mechanisms and tools. This may have an impact on both the users' trust as well as the reputation of the system's stakeholders. Designing and implementing security enhanced systems is of vital importance. Therefore, this book aims to present a number of innovative security enhanced applications. It is titled "Security Enhanced Applications for Information Systems" and includes 11 chapters. This book is a quality guide for teaching purposes as well as for young researchers since it presents leading innovative contributions on security enhanced applications on various Information Systems. It involves cases based on the standalone, network and Cloud environments.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821