Study of data collection, intrusion and detection using fuzzy logic

[1]Nabonarayan Jha, [2]Jaynarayan Jha, [3]K. B. Singh

[1]Department of Mathematics, Patan Multiple Campus, Patan Dhoka, Lalitpur,Tribhuvan University, Kathmandu.
[2]Department of Mathemathics, Padmakanya Multiple Campus, Tribhuvan University, Kathmandu.
[3]Samastipur College, Samastippur, L. N. M. University, Darbhanga.

***Abstract*: The design of a Network Intrusion Detection System (NIDS) is a delicate process which requires the successful completion of numerous design stages. The feature selection stage is one of the first steps that needs to be addressed, and can be considered among the top most important ones. If this step is not carefully considered the overall performance of the NIDS will greatly suffer, regardless of the detection technique, or any other algorithms that the NIDS is using. The most common approach for selecting the network features is to use expert knowledge to reason about the selection process. However, this approach is not deterministic, thus, in most cases researchers end-up with completely different sets of important features for the detection process. Furthermore, the lack of a generally accepted feature classification schema forces different researchers to use different names for the same (subsets of) features, or the same name for completely different ones. In this paper we present about the data collection, intrusion and detection using fuzzy system network.**

Keywords: NIDS, IDSs, AIIDS, HTML, ASP, PHP, JVM, OS, API, ABIDS, HIDS

I. INTRODUCTION

Computer security is one of the top priorities of our modern society. The Internet growth, information sharing, and technology improvement are some of the factors that humans become dependent on. Illegal access to private and confidential data has become a new way of crime. Tools for malicious purposes are freely available for download from the Internet. Hackers all over the world no longer need to have a strong IT background in order to perform attacks. There are several avenues that help regular users or professional administrators to better protect their data. The first line of defence against intruders is to adopt preventive measures such as physical access control, logical access control (i.e., passwords and encryption) or network access control (i.e., firewalls, VPNs) that dramatically decrease the chances of an intruder to compromise the data. Another preventive measure is to keep the software updated with the latest security patches that are released to avoid software-bug exploits. Another common way not only to prevent but also to combat unwanted activity is the use of Intrusion Detection Systems (IDS) that can detect and also actively or passively respond to intrusions.

II. RESULTS AND DISCUSSION

Data acquisition is one of the biggest challenges that a network security system must undertake. The decision on the amount of data, and the type and place of the data capturing process dramatically influences not only the performance of the system, but also its trustworthiness and detection scope. Based on the type of data that is collected, the IDSs can be classified into five main categories as follows:

a. Application-integrated IDSs: are those IDSs that collect the data out of a single application. They have an embedded sensor inside the application itself that collects and sends the extracted features to the IDS for processing.

b. Application-based IDSs: are those IDSs that monitor only one application by transparently collecting the necessary data. This is done by the use of external sensors that detect and capture the data exchanged between the monitored application and those third party entities (e.g., applications, hosts) that it interacts with.

c. Host-based IDSs: evaluate and keep track of the wellness of a host as an entity by monitoring its applications.

d. Network-based IDSs: are those IDSs that are meant to protect the network itself. The network is pictured as a collection of hosts that interact by exchanging messages that are transferred through wires (i.e., in the case of a wired network) or radio waves (i.e., in the case of a wireless network). The sensor of the IDS collects data by sniffing it directly from the network traffic in a transparent way.

A. Application-integrated IDSs

An application-integrated IDS (AIIDS) is designed to monitor a particular application. The sensor of the IDS is built as a tightly coupled module inside the application itself to extract the desired information. The main advantages of this technique is the in-depth knowledge that the IDS has with respect to the monitored application. Thus, all AIIDS implement a specification-based detection designed to detect any deviation of the monitored application from the desired behavior. This is done by studying the internal states that the application goes through at running time. Moreover, due to the tight-coupling technique that is used, the resynchronizations between the AIIDS and monitored application is eliminated, allowing for prompt response actions. Applications that provide web services, are among the most appealing targets of attackers. Apache is a commonly used web server application that supports different kinds of languages (e.g., HTML, ASP, PHP, Perl, to name a few) for providing web services to the users. One way to protect this application is to mine the communication between the Apache server and its users [1]. This can be done by treating each request to the server as a sequence of stages that have to be executed. At the end of each stage the IDS has the ability to send feedback to Apache that might influence the current and future operations that the server is performing for accomplishing the user request.

B. Limitations and solutions

The main problem with the application-integrated approach is that it makes the IDS dependent on one particular application and commercially unattractive. A solution that partially solves this issue is to enable one IDS to communicate with multiple applications at the same time through a standardized application user interface (API) [3]. In this case the sensor is not integrated into the application anymore, but the application itself sends data to the IDS by calling the appropriate methods from the interface. The advantage of this particular implementation is that it accommodates a bidirectional feedback between the IDS and the monitored applications. Three types of data that the IDS is expecting to receive from applications are defined as follows: data related to the identity of application (i.e, *Credentials*), data dependent on the domain that the application is running in (i.e., *Domain Specific Information*), and data that remains semantically unchanged across multiple domains (i.e., *Domain Independent Information*). However, despite its advantages this new approach also introduces further issues related to the system authentication and communication trustworthiness.

C. Application-based IDSs

One alternative to the previously described technique is transforming the IDS into an Application-Based IDS (ABIDS) that will function as a transparent entity to the monitored application. This gains flexibility for the developing phase of the application, making its implementation an independent task from the one of the ABIDS. In order to monitor the application, the IDS must take advantage of the third party mechanisms of the OS (e.g., logging) that will allow it to collect the targeted data without any direct interaction with the monitored application itself. This method, also known as the *interception* technique [3], entails decoding and monitoring of the data being exchanged between the application and third party entities in a transparent manner for the applications. Anomaly detection methods are also used as underling implementation mechanisms for the ABIDSs. For instance, for detecting web-based attacks against web-servers, this method uses as primary feature the GET requests since they are the most critical commands that a web server has to fulfill. The ABIDS may also analyze the parameters that are passed along with the request by monitoring their length, structure, order of arguments, presence or absence of arguments, character distribution for string arguments, and set of values for numerical arguments [4;5;6].

D. Limitations and solutions

Despite the degree of details and quality of data that an IDS can explore once implemented as an application-based system, it still remains limited to a single type of application. Limitations like applicability, resource consumption as well as maintenance are only some of the disadvantages of this method. Moreover, using this approach, the protection of the whole host can be achieved if for each of the existing applications in the current host an ABIDS is deployed. This is not practically feasible since in the first place there are not too many applications that actually have specialized ABIDSs, and even if we assume they do, the load introduced by this extra layer of protection will rapidly exhaust the system resources.

E. Host-based IDSs

The host-based approach is one of the most common approaches in intrusion detection. An IDS using host-based data is basically a software residing on the protected host, which constantly searches for abnormalities of the data. These types of IDSs mine different sources of data such as memory contents, process behavior, system calls, file system accesses, or log files. Thus, the detection coverage of a Host-based IDS (HIDS) is physically limited to the protected host.

F. Audit Logs

The OS auditing feature has been primarily adopted as a means to provide feedback to the OS developers. Later on, this functionality also proved to be a good source of information for intrusion detection. The OS logs a considerable amount of data that is not necessarily useful for intrusion detection. Thus, the HIDSs that use audit logs filter that data and monitor only a small set of selected tasks that their abusive execution might compromise the system. For instance, for Unix OS some of those programs are admintool, eject, fdformat, mount, passwd, ps, su, traceroute, whodo, to name a few. Once the set of critical applications to trace is decided, all detection techniques such as anomaly-, specification-, and signature-based can be used to discriminate between normal and abusive executions. For example, in the case of anomaly detection, techniques that involved the use of Recurrent Neural Networks [7], Support Vector Machines [8], Finite State Machines [7] proved to successfully discriminate between the normal and abnormal executions. In these cases, the HIDS learns the normal behavior of each monitored application and signals any kind of deviation from the created profile.

G. Limitations and solutions of Audit Logs

Using audit logs for security related purposes is a challenging task due to incompatibility reasons (i.e., different OSs have different formats for the log files, and save different types of data), transforming the HIDS into a platform dependent one. Furthermore, since the log functionality was not primarily designed for reporting security related events, extracting the desired data out of a log file is a difficult and time consuming task [10]. Furthermore, in most cases the extracted data proves to be insufficient and to have a poor quality [10]. One solution to solve this problem is to have an auditing language that formalizes the description of a security related audit log [10]. However, since the implementation of such a solution involves multiple HIDS and OS vendors it remains very challenging in practice.

H. System Call Sequences

Each OS provides a set of unique basic operations that any application must invoke in order to accomplish its tasks. These atomic operations are refereed to as System Calls (SC). The SC facilitate the manipulation of all resources that the OS has access to, such as the file system, memory, external devices, to name a few. Thus, any application can be indirectly monitored by studying the set of SC that it uses and the time relations between consecutive SCs. The expectation is that once an application is compromised its behavior will change in such a way that an IDS will be able to detect it by only looking at SCs. One way to use the SC for anomaly detection is to create normal profiles for each individual application by monitoring short SC sequences that it uses [13; 14]. The created profiles proved to differ from application to application, and be consistent throughout different runs of the same application. Furthermore, a different approach is to study the arguments of the SC invocations [6]. This approach completely ignores the relationship between consecutive SC, the normal profiles being built based on features extracted from the arguments of each individual SC. The arguments are pictured as touples of *n* values. Each touple can be studied versus four main characteristics such as *string length*, *string character distribution*, *structural interface*, and *tokens* that it contains. One property of the SC arguments is that most of the times they are composed of human readable characters and their length does not usually exceed 100 characters. Moreover, the characters in a string occur with different distributions that can be learned. The *structural interface* of an argument can also be learned since most of them represent system paths to files that the application uses. The specification-based approach can also be successfully used when using SCs [15; 16]. In this case a set of rules can be defined for each individual application with respect to the allowed actions that it can perform. In other words, the HIDS detects the attack each time when the application does not respect its specifications. The allowed profiles can be built for instance by defining rules for the client-server communication, file system access, registry access, to name a few. The SC approach has the advantage of making the implementation of HIDS independent of any application design and implementation that is being monitored.

I. Limitations and solutions of system call sequences

One of the disadvantages of using SC is the OS dependency, and as a consequence, the same application running on two different OSs (e.g. the same Java program) will end up having two different profiles. Moreover, due to the diversity of applications, some of the HIDSs concentrate only upon those applications considered to be critical, or susceptible to attacks, and this automatically restricts the detection scope of the HIDS. Next, a HIDS can negatively influence the performance of the host machine through the constant set of computations it needs to execute. Finally, another disadvantage of this technique that in fact applies to all host-based IDSs, is that they cannot detect network related attacks, such as worm spread, distributed denial of service attacks (DDoS), and probing. Furthermore, to gain protection of all hosts in an enterprise network, an HIDS needs to be installed, and maintained on all the hosts of the network, which can prove to be a difficult and time consuming task.

J. Network-based IDSs

Network-based IDSs (NIDS) capture and analyze data directly from the network, aiming to identify intrusions that exhibit abnormality at the network level, such as worms, DoS, DDoS, probing attacks, as well as application-based attacks through packet payload inspection. The data is usually sniffed from the network by the use of data collectors. There are two main types of NIDSs such as Inline and Online NIDSs. The former one captures, analyzes, and resends the packets back in the network, while the latter one only captures and analyzes the packets through a mirror port. The coverage of a NIDS depends on the place that the sniffing is done. If the sniffing is done on the main link to the Internet, then the NIDS sees all the communication coming and going between the Internet and local hosts of the network, but misses the traffic within the protected network. A NIDS can be also placed on the subnet level, in which case it is limited to analyze all the traffic regarding a particular set of hosts within the internal network. A usual deployment of a NIDS consists of a distributed solution that includes several sensors strategically placed in the network and at least one correlation box that combines the data received from the sensors and provides a centralize interface which the administrator of the network can use to manipulate the collected data.

K. SNMP

The Simple Network Management Protocol (SNMP) is a widely used network monitoring and control protocol that was introduced by the Internet Engineering Task Force (IETF). It describes the way and format of data that is passed between the SNMP agents and the Network Management System (NMS) that supervises the network. The management information is stored into a hierarchical (i.e. tree structured) database using Management Information Based (MIB) objects as defined in RFC1066 [17]. MIB variables are defined for almost any protocol and are used to monitor the performance of network devices. The SNMP agents are hardware or software processes that report the performance activity of a device, and are usually implemented in each network device such as a hub, router, switch or firewall. There is no connection between the NIDSs and NMSs systems, even though a lot of information collected by a NMS is useful for NIDS and viceversa. Several methods have been explored by different researchers to make these two types of devices collaborate [18; 19]. This will make the NMSs more robust security wise, and NIDSs more aware of what happens in the network. Furthermore, some of the features computed by the NMSs could be directly used by the NIDSs for attack detection. For instance, by using the MIB *up In Errors* variable an NIDS might detect an UDP flooding attack without analyzing the content of the packets that are exchanged in the network. One NIDS example that successfully uses the information stored in MIB objects for detecting intrusions is the RIPPER system [18]. It detects DoS and probing attacks by creating rules that monitor the values of MIB objects in time. Statistical techniques are also used for detecting any abrupt changes in the values of MIB variables for anomaly detection [19].

L. Limitations of the SNMP approach

Although promising, this approach is not applicable in practice due to multiple reasons. In the first place, it requires the NIDS to rely on third party generated data. Most of the system administrators actually disable the SNMP functionality since the data may be also misused by attackers to learn about valuable information about the current network. Next, the method also introduces unnecessary delays between the two devices. Finally, the privacy and authenticity of the data can also be questionable.

M. Packets

The fundamental unit of information carriage in the network is represented by a packet. Thus, by capturing packets a NIDS is theoretically able to analyze all the data that is exchanged between the hosts, making it the best approach for an enterprise intrusion detection solution. However, due to the massive amount of traffic generated in a network, the diversity of protocols, and some privacy and confidentiality issues, most of the IDSs analyze only some part of the captured data. Once the packets are captured, they are sent to the processing and detection engines, where they are grouped into flows (also known as connections) for further processing. The most common solution for the deployment of an NIDS is the centralized approach. This approach uses only one sensor for collecting the packets from the network, and is adopted by most of the commercially available IDSs and Intrusion Prevention Systems (IPS). The advantage of having a single data collector is that it can be implemented on the same device where the processing and detection engine resides. This has a huge impact on the communication speed between the capturing and detection engine, improving its overall performance. The main disadvantage of centralized IDSs is the lack of adaptability to different network topologies, its common deployment being on the main link that connects the protected network to the Internet. The second solution is a distributed-NIDS, where multiple collectors are scattered through the network in key places and a centralized point receives and processes their output. The advantage of such a technique is the high adaptability to any kind of network topologies. In the distributed architecture, all of the devices that belong to the NIDS will need to communicate among themselves in a secure and timely fashion. To avoid attack exposure, in almost all cases an extra network is dedicated only for this type of communication.

N. Limitations and solutions for the data collection from packets

Despite all of the advantages that NIDS provide, this technique faces several challenges due to ever increasing data volume that has to be processed, diversity of protocols, huge number of features that can be computed for each packet, and speed requirements. A NIDS is essentially a black box that stays inline or online and constantly probes the data for potential attackers. Inline devices are known as possible botlenecks of the network [31], that may not be able to keep up all the time with the network speed and have the potential to produce more damage than the actual attack. Thus, situations when the packets are dropped or accidental latencies are introduced need to be avoided. This goal is very hard to achieve since a NIDS needs to pass the data through several processing stages (e.g., data collection, feature extraction, detection, planning, and response) in a very short and limited time. This task becomes even more critical and hard to accomplish during an actual attack such as DoS [32]. A solution to handle the high demand of processing power is to use a set of distributed sensors that will each process a much smaller portion of the data, each component being responsible for a certain part of the network. Besides the advantages that this approach introduces, it is still susceptible to delays due to the communications between different parts of the system. Moreover, most of the problems arise because the communication protocol is usually implemented as a client-server architecture. Helmer et al. [33] explain some of the problems that this paradigm is deemed to have. Firstly, the need for a stable connection between the client and server, which is unlikely to happen in the case of a DoS attack, as well as in the case of poor quality links. Secondly, once a communication protocol is de**fi**ned between the client and server it is very hard to be dynamically changed at the execution time, the only choice remaining the recoding of the whole protocol.

III. CONCLUSION

Finally, we find that such deployment produces an additional unwanted load in the network in order to keep alive the connection between the different parts. Some of these disadvantages can be easily solved if a separate dedicated network or channel is implemented only for critical data communication. The most difficult challenge to solve is the tradeoff between the amount and type of data that the NIDS considers for mining purposes, and the number of intrusion types that it targets. Thus, due to constraints such as, large computational time, diversity of protocols and applications that exist, and memory consumption, many of the implementations do resume to only detect a small subset of the intrusions types.
In the recent years, despite the challenges, due to the unlimited amounts of data that can be explored in a networked environment, NIDS that extract data from network packets become a new standard in intrusion detection.

REFERENCES

[1] Magnus Almgren and Ulf Lindqvist, *Application-integrated data collection for security monitoring*, Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium, (RAID 2001) (Davis, CA, USA) (W, L. M Lee, and A. Wespi, eds.), Lecture Notes in Computer Science, Springer-Verlag Heidelberg, October 2001, pp. 22{36.
[2] S. Soman, C. Krintz, and G. Vigna, *Detecting malicious java code using virtual machine auditing*, Proceedings of 12*th* USENIX Security Symposium (Washington, DC) (V. Paxson, ed.), USENIX, August 2003, pp. 153{167.Marc Welz and Andrew Hutchison, *Interfacing trusted applications with intr*

[3] *usion detection systems*, Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium, (RAID 2001) (Davis, CA, USA) (W, L. M Lee, and A. Wespi, eds.), Lecture Notes in Computer Science, Springer-Verlag Heidelberg, October 2001, pp. 37{53.

[4] Christopher Kruegel and Giovanni Vigna, *Anomaly detection of web-based attacks*, Proceedings of the 10th ACM conference on Computer and communication security (Washington D.C., USA), ACM Press, October 2003, pp. 251{261.

[5] K. Das, *The development of stealthy attacks to evaluate intrusion detection systems*, Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 2000.

[6] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna, *On the detection of anomalous system call arguments*, Proceedings of the 8*th* European Symposium on Research in Computer Security (ESORICS '03) (Gjovik, Norway), LNCS, Springer-Verlag, October 2003, pp. 326{343.

[7] Anup K. Ghosh, Christoph Michael, and Michael Schatz, *A real-time intrusion detection system based on learning program behavior*, Proceedings of Recent Advances in Intrusion Detection, 3rd International Symposium, (RAID 2000)(Toulouse, France) (H. Debar, L. M, and S.F. Wu, eds.), Lecture Notes in Computer Science, Springer-Verlag Heidelberg, October 2000, pp. 93{109.

[8] Zonghua Zhang and Hong Shen, *Online training of svms for real-time intrusion detection*, Proceedings of the 18th International Conference on Advanced Information Networking and Applications, AINA'04, vol. 1, March 2004, pp. 568{573.

[9] Josu Kuri, Gonzalo Navarro, Ludovic M, and Laurent Heye, *A pattern matching based filter for audit reduction and fast detection of potential intrusions*, Proceedings of Recent Advances in Intrusion Detection, 3rd International Symposium, (RAID 2000) (Toulouse, France) (H. Debar, L. M, and S.F. Wu, eds.), Lecture Notes in Computer Science, Springer-Verlag Heidelberg, October 2000, pp. 17{27.

[10] Chapman Flack and Mikhail J. Atallah, *Better logging through formality applying formal specification techniques to improve audit logs and log consumers*, Proceedings of Recent Advances in Intrusion Detection, 3rd International Symposium, (RAID 2000) (Toulouse, France) (H. Debar, L. M, and S.F. Wu, eds.), Lecture Notes in Computer Science, Springer-Verlag Heidelberg, October 2000, pp. 1{16.

[11] Joachim Biskup and Ulrich Flegel, *Transaction-based pseudonyms in audit data for privacy respecting intrusion detection*, Proceedings of Recent Advances in Intrusion Detection, 3rd International Symposium, (RAID 2000)(Toulouse, France) (H. Debar, L. M, and S.F. Wu, eds.), Lecture Notes in Computer Science, Springer-Verlag Heidelberg, October 2000, pp. 28{48.

[12] Bruce Schneier and John Kelsey, *Secure audit logs to support computer forensics*, ACM Transactions on Information and System Security (TISSEC) 2 (1999), no. 2, 159{176.

[13] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, *A sense of self for unix processes*, Proceedings of the 1996 IEEE Symposium on Security and Privacy (Los Alamitos, CA), IEEE Computer Society Press, 1996, p. 120128.

[14] Li Zhuowei, A. Das, and S. Nandi, *Utilizing statistical characteristics of ngrams for intrusion detection*, Proceedings of the International Conference on Cyberworlds, December 2003, pp. 486{493.

[15] C. Ko, *System health and intrusion monitoring (shim): project summary*, Proceedings of The DARPA Information Survivability Conference and Exposition II, DISCEX'03, vol. 2, April 2003, pp. 202{207.

[16] OKENA, *Stormsystem*, August 2002, Cisco acquired Okena
in 2003.

[17] K. McCloghrie and M. Rose, *Management information base for network management of tcp/ip-based internets*, Request For Comments: 1066, August 1988.

[18] Y. Rui, T. Huang, and S. Chang, *Image retrieval: current techniques, promising directions and open issues*, Journal of Visual Communication and Image Representation 10 (1999), no. 4, 39{62.

[19] Marina Thottan and Chuanyi Ji, *Anomaly detection in ip networks*, IEEE Transactions on Signal Processing 51 (2003), no. 8, 148{166.

[20] M. V. Mahoney and P. K. Chan, *PHAD: Packet header anomaly detection for identifying hostile network traffic*, Tech. Report CS-2001-4, Florida Tech, 2001.

[21] DARPA, *Darpa intrusion detection and evaluation dataset 1999*, http://www.ll.mit.edu, Website, Last accessed February 2006.

[22] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P. Tan, *Data mining for network intrusion detection*, Proceedings of NSF Workshop on Next Generation Data Mining (Baltimore, MD), November 2002, pp. 21{30.

[23] L. Ertoz, E. Eilertson, A. Lazarevic, P.N. Tan, P. Dokas, V. Kumar, and J. Srivastava, *Detection of novel network attacks using data mining*, In ICDM Workshop on Data Mining for Computer Security (DMSEC) (Melbourne, FL), Nov. 19 2003, pp. 30{39.

[24] S. J. Stolfo W. Lee and K. W. Mok, *Mining in a data-flow environment: Experience in network intrusion detection*, Proceedings of the 5 International Conference on Knowledge Discovery and Data Mining, 1999, pp. 114{124.

[25] S. Staniford, J. Hoagland, and J. McAlerney, *Practical automated detection of stealthy portscans*, Journal of Computer Security 10 (2002), no. 1 and 2,105{126.

[26] D. Zhang H.Wang and K. G. Shin, *Detecting syn °ooding attacks*, Proceedings of the INFOCOM Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (New York, NY), June 2002, pp. 1530{ 1539.

[27] K. Choi S. Noh, C. Lee and G. Jung, *Detecting distributed denial of service (ddos) attacks through inductive learning*, Proceedings of the Intelligent Data Engineering and Automated Learning, 4th International Conference, IDEAL 2003 (Hong Kong, China), Lecture Notes in Computer Science, vol. 2690, Springer, March 21-23 2003, pp. 287{295.
[28] T. Toth and C. Kruegel, *Connection-history based anomaly detection*, Proceedings of IEEE Workshop on Information Assurance and Security (West Point, NY), June 2002, pp. 1{6.
[29] V. Berk, G. Bakos, and R. Morris, *Designing a framework for active worm detection on global networks*, Proceedings of the IEEE InternationalWorkshop on Information Assurance (Darmstadt, Germany), March 2003, pp. 13{23.
[30] X. Wu V. A. Mahadik and Douglas S. Reeves, *A summary of detection of denial-of-QoS attacks on DiffServ networks*, Proceedings of the DARPA Information Survivability Conference and Exposition, April 2003, pp. 277{282.
[31] Greg Taleck, *Ambiguity resolution via passive os fingerprinting*, Proceedings of Recent Advances in Intrusion Detection, 6th International Symposium, (RAID 2003) (Pittsburgh, PA, USA) (G. Vigna, E. Jonsson, and C. Kruegel, eds.), Lecture Notes in Computer Science, Springer-Verlag Heidelberg, September 2003, pp. 192{206.
[32] T.H. Ong, C.P. Tan, Y.T Tan, and Christopher Ting, *Snms - shadow network management system*, Proceedings of Recent Advances in Intrusion Detection, 2nd International Symposium, (RAID 1999) (Purdue, IN, USA), September 1999, http://www.raid-symposium.org/raid99/PAPERS/ChungPheng.pdf.
[33] G. Helmer, J. S. K. Wong, V. Honavar, L. Miller, and Y. Wong, *Lightweight agents for intrusion detection*, Journal of Systems and Software 67 (2003),no. 2, 109{122.
[34] R. Basu, R.K. Cunningham, S. E. Webster, and R. P. Lippmann, *Detecting low-profile probes and novel denial-of-service attacks*, Proceedings of the workshop on Information Assurance and Security, United States Military Academy (IEEE, ed.), June 2001, pp. 5{10.
[35] T. Peng, C. Leckie, and R. Kotagiri, *Proactively detecting ddos attack using source ip address monitoring*, Proceedings of the Third International IFIPTC6 Networking Conference (Networking 2004) (Athens, Greece), May 2004, pp. 771{782.