Kim, Myung-Sup, et al. "A flow-based method for abnormal network traffic detection." 2004 IEEE/IFIP network operations and management symposium (IEEE Cat. No. 04CH37507). Vol. 1. IEEE, 2004.

Today, the number of Internet users is continuously increasing, along with new network services. As the internet grows, network security attack threats have become more serious.
One recent trend in network security attacks is an increasing number of indirect attacks which influence network traffic negatively, instead of directly entering a system and damaging it. In future, damages from this type of attack are expected to become more serious. The attacks on a famous website, such as Yahoo, E-bay, and E*trade, are good examples [1, 2]. In addition, the bandwidth consumption by these attacks influences the entire network performance.
Even on highly over-provisioned links, malicious traffic causes an increase in the average DNS latency by 230% and an increase in the average web latency by 30% [6]. Flooding attacks transmit many spurious packets to the target system, and waste CPU, memory, and network resources. In case of TCP SYN flood [14], the victim receives packets that exceed buffer of the data structure limit and cripple its service
Also, ICMP, TCP, and UDP flooding attacks [16] overwhelm bandwidth by sending useless traffic to the victim.

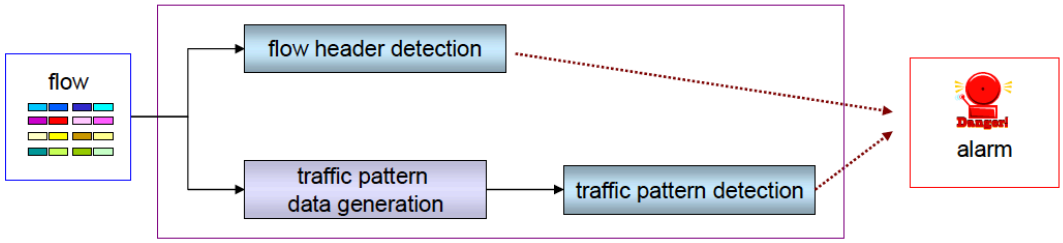


Figure 1. Overall Detection Process

To detect the attacks described above, network-based Intrusion Detection Systems (NIDS), such as snort [19], in a packet header or payload. Signature-based detecting systems require a huge database that contains information on every attack.
It causes much system overhead to compare every packet with the signatures in the database. Therefore, these systems are not appropriate in a high-speed network. Thus, if a new or mutant attack appears, the signature detecting method cannot catch it. In addition, packet information may be insufficient because some types of attack can be detected only by using from a series of packets information. Other types of detecting methods have been suggested. These approaches monitor the volume of traffic which every single host has received [20, 21] or the number of new source IP addresses [22]. These methods identify attacks by checking the volume of traffic or the number of new source IP addresses. These methods may have low overhead, but can result false alarms. To reduce false alarms, it

is necessary to use all the possible traffic parameters.

This paper focuses on detecting abnormal network traffic which includes those generated by internet worms[6], DoS/DDoS and scanning[24], scanning which include both port and network scanning. We present a detecting method and a system prototype. We analyze network traffic based on flows, which is defined as collections of packets that travel between the same end points [8]. By aggregating packets that belong to the identical flow, we can reduce processing overhead in the system. We suggest a detecting algorithm using changes in traffic patterns that appear during attacks.
The detection module receives flow information from monitoring systems or routers. the overall process consists of two parts: the flow header detection and the traffic pattern detection. The flow header detection takes part in checking the fields of the flow headers.
By validating these values, this part mainly detects logic attacks.

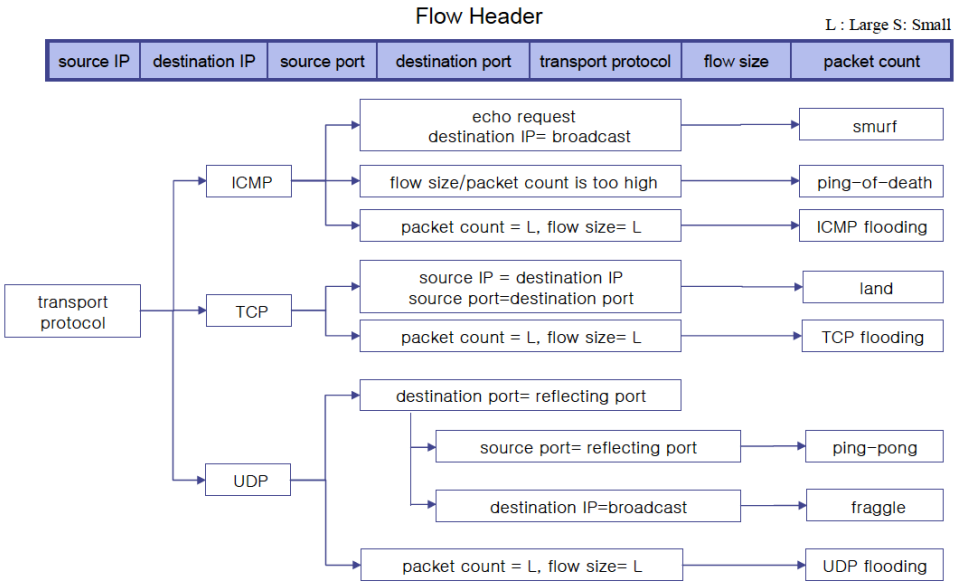


Figure 2. Flow Header Detection Sequence

The flow header detection part checks the field values of a flow header. The diagram in Figure 2 classifies attacks by the field values of the flow header. This part can detect logic attacks or other attacks with a specific header such as broadcast destination or specific port numbers. We define flow as a collection of packets with the same 5-tuple: source IP address, destination IP address, source port, destination port, and protocol number.
The flow size and packet count, refer to the total bytes and the number of packets that belongs to the flow, respectively.

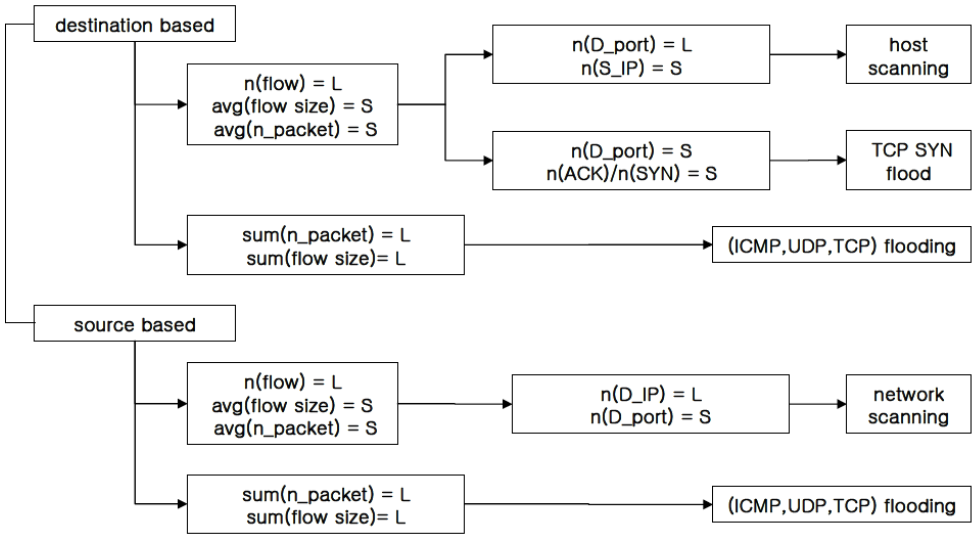


Figure 4. Traffic Pattern Detection Sequence

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

TCP attacks
In the case of a TCP transport protocol; this part certifies if the pair of source IP, source port is identical with the pair of destination IP, destination port for the purpose of detecting a Land attack.
The TCP SYN flood induces a lot of flow activities, because it sends many packets to a specific port of the victim. Also, the packet count and total packet length in each flow are small, as this attack sends small SYN packets. However, the total bandwidth and total packet count vary according to the number of transmitted packets.

UDP attacks:
In UDP flows, Fraggle and Ping-Pong attacks use UDP reflecting services, such as echo (port 7), chargen (port 19), daytime (port13), and qotd (port 17). Therefore, the port numbers of source and destination port are validated. If both destination and source ports are reflecting port numbers, then this flow is used for the Ping-Pong attack. Also, if the destination port is a reflecting port and the destination IP is a broadcast address, the flow is supposed to be a Fraggle attack, similar to the Smurf attack.

Scanning attacks:
During scanning, the attacker makes many connection attempts. Consequently, many flows are generated and the packet count in each flow is small, when a scanning occurs. In addition, the packet size is mostly small (about 40 bytes), because the attacker sends small packets and observes responses from these packets.

All layers:
In each transport protocol, the flow header detection part searches flows with a large packet count and flow size in order to identify flooding flows. For example, if a large number of ICMP packets and its flow size is large, then it is determined to be an ICMP flooding. To determine whether the flow size and packet count of a flow are large or not, currently we are using a percentile threshold value to total flow size and total packet count.

L : Large S : Small

Attacks / Property	scanning		flooding				
	host	network	TCP SYN	smurf	fraggle	ping-pong	general (ICMP,UDP,TCP) flooding
flow count	L	L	L	L	L	S	L or S
flow size/flow	S	S	S	L or S	L or S	L	L or S
packet count/flow	S	S	S	L or S	L or S	L	L or S
packet size	S	S	S	L or S	L or S	L or S	L or S
total bandwidth	L or S	L or S	L or S	L	L	L	L
total packet count	L or S	L or S	L or S	L	L	L	L
property	1 destination	1 port		ICMP broadcast	UDP broadcast	reflecting port	

Table 1. Characterization of Attack Traffic Patterns

When discovering the traffic patterns described in Table 1, some attacks are not possible to detect only with the flow information. Therefore, we generate traffic pattern data by aggregating related flows. In order to check traffic characteristics, we generate traffic information that are sent and received from a certain host.
See table 2 for the Network Parameters in the anomaly detection and figure 4 for the chain of detection.
References
[1] CNN, Immense network assault takes down yahoo, February 2000, http:// www.cnn.com/2000/TECH/computing/02/08/yahoo.assault.idg/index.ht ml.
[2] CNN, Cyber-attacks batter web heavyweights, February 2000, http://www.cnn.com/ 2000/TECH/computing/02/09/cyber.attacks.01/.
[6] Kun-chan Lan, Alefiya Hussain, and Debojyoti Dutta, “Effect of Malicious Traffic on the Network,” Proc. of PAM 2003, San Diego, California, April 2003.
[8] Siegfried Lifler, “Using Flows for Analysis and Measurement of Internet Traffic,”

Diploma Thesis, Istitute of Communication Network and Computer Engineering, University of Stuttgart, Germany, 1997.
[14] Common Vulnerabilities and Exposures (CVE), “SYN flood (CVE-1999-0116),” http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0116.
[16] Common Vulnerabilities and Exposures (CVE), “UDP packet storm (CVE-1999-0103),” http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE- 1999-0103.
[19] M. Roesch, “Snort - lightweight intrusion detection for networks,” http://www.snort.org.
[20] Rudolf B. Blazek, Hongjoong Kim, Boris Rozovskii, and Alexander Tartakovsky, “A novel approach to detection of denial-of-service attacks via adaptive sequential and batch sequential change-point detection methods,” Pro. of IEEE Systems, Man and Cybernetics Information Assurance Workshop, New York, June 2001.
[21] David K. Y. Yau, John C. S. Lui, and Feng Lian, “Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles,” Proc. of IEEE International Workshop on Quality of Service (IWQoS), Miami Beach, Florida, May 2002.
[22] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao, “Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring,” http://www.ee.mu.oz.au/pgrad/taop/research/detection.pdf.
[24] Urupoj Kanlayasiri, Surasak Sanguanpong, and Wipa Jaratmanachot, “A Rule-based Approach for Port Scanning Detection,” Proc. of the 23nd Electrical Engineering Conference (EECON-23), Chiangmai, Thailand, November 2000.

==========

Bhuyan, Monowar H., Dhruba Kumar Bhattacharyya, and Jugal K. Kalita. "Network anomaly detection: methods, systems and tools." Ieee communications surveys & tutorials 16.1 (2013): 303-336.

In this paper, we provide a structured and comprehensive overview of various facets of network anomaly detection so that a researcher can become quickly familiar with every aspect of network anomaly detection. We present attacks normally encountered by network intrusion detection systems. We categorize existing network anomaly detection methods and systems based on the underlying computational techniques used. Within this framework, we briefly describe and compare a large number of network anomaly detection methods and systems.

CLASSES OF COMPUTER ATTACKS: CHARACTERISTICS AND EXAMPLE

Attack name	*Characteristics*	*Example*
Virus	(i) A self replicating program that infects the system without any knowledge or permission from the user. (ii) Increases the infection rate of a network file system if the system is accessed by another computer.	Trivial.88.D, Polyboot.B, Tuareg
Worm	(i) A self replicating program that propagates through network services on computer systems without user intervention. (ii) Can highly harm network by consuming network bandwidth.	SQL Slammer, Mydoom, CodeRed Nimda
Trojan	(i) A malicious program that cannot replicate itself but can cause serious security problems in the computer system. (ii) Appears as a useful program but in reality it has a secret code that can create a backdoor to the system, allowing it to do anything on the system easily, and can be called as the hacker gets control on the system without user permission.	Example-Mail Bomb, phishing attack
Denial of service (DoS)	(i) Attempts to block access to system or network resources. (ii) The loss of service is the inability of a particular network or a host service, such as e-mail to function. (iii) It is implemented by either forcing the targeted computer(s) to reset, or consuming resources. (iv) Intended users can no longer communicate adequately due to non-availability of service or because of obstructed communication media.	Buffer overflow, ping of death(PoD), TCP SYN, smurf, teardrop
Network Attack	(i) Any process used to maliciously attempt to compromise the security of the network ranging from the data link layer to the application layer by various means such as manipulation of network protocols. (ii) Illegally using user accounts and privileges, performing actions to delete network resources and bandwidth, performing actions that prevent legitimate authorized users from accessing network services and resources.	Packet injection, SYN flood
Physical Attack	An attempt to damage the physical components of networks or computers.	Cold boot, evil maid
Password Attack	Aims to gain a password within a short period of time, and is usually indicated by a series of login failures.	Dictionary attack, SQL injection attack
Information Gathering Attack	Gathers information or finds known vulnerabilities by scanning or probing computers or networks.	SYS scan, FIN scan, XMAS scan
User to Root (U2R) attack	(i) It is able to exploit vulnerabilities to gain privileges of superuser of the system while starting as a normal user on the system. (ii) Vulnerabilities include sniffing passwords, dictionary attack, or social engineering.	Rootkit, loadmodule, perl
Remote to Local (R2L) attack	(i) Ability to send packets to a remote system over a network without having any account on that system, gain access either as a user or as a root to the system and do harmful operations. (ii) Performs attack against public services (such as HTTP and FTP) or during the connection of protected services (such as POP and IMAP).	Warezclient, warezmaster, imap, ftp_write, multihop, phf, spy
Probe	(i) Scans the networks to identify valid IP addresses and to collect information about host (e.g., what services they offer, operating system used). (ii) Provides information to an attacker with the list of potential vulnerabilities that can later be used to launch an attack against selected systems and services.	IPsweep, portsweep

According to Anderson [2], an intrusion attempt or a threat is a deliberate and unauthorized attempt to (i) access information, (ii) manipulate information, or (iii) render a system unreliable or unusable.
The term anomaly-based intrusion detection in networks refers to the problem of finding exceptional patterns in network traffic that do not conform to the expected normal behavior. These nonconforming patterns are often referred to as anomalies, outliers, exceptions, aberrations, surprises,

Prior research:
An extensive survey of anomaly detection techniques developed in machine learning and statistics has been provided by [8], [9]. Agyemang et al. [10] present a broad review of anomaly detection techniques for numeric as well as symbolic data.
Patcha and Park [6] and Snyder [12] present surveys of anomaly detection techniques used specifically for cyber intrusion detection. Callado et al. [19] report major techniques and problems identified in IP traffic analysis, with an emphasis on application detection. Zhang
et al. [20] present a survey on anomaly detection methods in networks. A review of flow-based intrusion detection is presented by Sperotto et al. [21], who explain the concepts of flow and classified attacks, and provide a detailed discussion of detection techniques for scans, worms, Botnets and DoS attacks.
Wu and Banzhaf [29] present an overview of applications of computational intelligence methods to the problem of intrusion detection

To provide an appropriate solution in network anomaly detection, we need the concept of normality. Consequently, an event or an object is detected as anomalous if its degree of deviation with respect to the profile or behavior of the system, specified by the

normality model,
is high enough.

Network intrusion detection has been studied for almost 20 years. Generally, an intruder's behavior is noticeably different from that of a legitimate user and hence can be detected [41]. IDSs can also be classified based on their deployment in real time.
1) Host-based IDS (HIDS): A HIDSmonitors and analyzes the internals of a computing system rather than its external interfaces [42].
2) Network-based IDS (NIDS): An NIDS deals with detecting intrusions in network data. Intrusions typically occur as anomalous patterns though certain techniques model the data in a sequential fashion and detect anomalous subsequences [42].

Anomaly detection has extensive applications in areas such as fraud detection for credit cards, intrusion detection for cyber security, and military surveillance for enemy activities.
Anomaly detection attempts to find patterns in data, which do not conform to expected normal behavior. The importance of anomaly detection is due to the fact that anomalies in data translate to significant (and often critical) actionable information in a wide variety of application domains [45].
As stated in [35], there are two broad categories of network anomalies: (a) performance related anomalies and (b) security related anomalies. Various examples of performance related anomalies are: broadcast storms, transient congestion, babbling node, paging across the network, and file server failure.
Security related network anomalies may be due to malicious activity of intruder(s) who intentionally flood the network with unnecessary traffic to hijack the bandwidth so that legitimate users are unable to receive service(s).

A. Generic Architecture of ANIDS
1) Anomaly detection engine: This is the heart of any network intrusion detection system. It attempts to detect occurrence of any intrusion either online or offline. However, before sending any network traffic to the detection engine, it needs preprocessing.
Matching mechanism: It entails looking for a particular pattern or profile in network traffic that can be built by continuous monitoring of network behavior including known exploits or vulnerabilities.
2) Reference data: The reference data stores information about known intrusion signatures or profiles of normal behavior. Reference data needs to be stored in an efficient manner. Possible types of reference data used in the generic architecture of a NIDS are: profile, signature and rule. In case of an ANIDS, it is mostly profiles.
3) Configuration data: This corresponds to intermediate results, e.g., partially created intrusion signatures.
4) Alarm: This component of the architecture is responsible for generation of alarm based on the indication received from the detection engine.
5) Human analyst: A human analyst is responsible for analysis, interpretation and for taking necessary action based on the alarm information provided by the detection engine.

6) Post-processing: This is an important module in a NIDS for post-processing of the generated alarms for diagnosis of actual attacks.
7) Capturing traffic: Traffic capturing is an important module in a NIDS. The raw traffic data is captured at both packet and flow levels.
8) Security manager: Stored intrusion signatures are updated by the Security Manager (SM) as and when new intrusions become known.

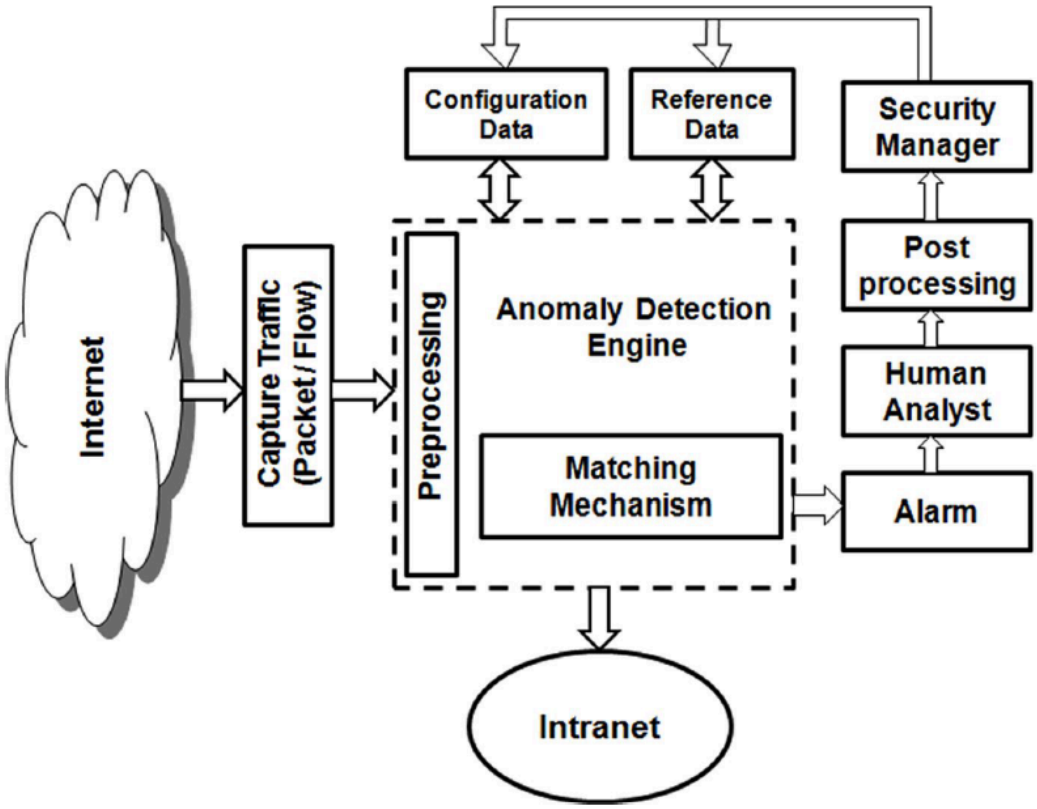


Fig. 1. A generic architecture of ANIDS

Intrusion detection techniques:
Technique Characteristics
Misuse based (i) Detection is based on a set of rules or signatures for known attacks. (ii) Can detect all known attack patterns based on the reference data. (iii) How to write a signature that encompasses all possible variations of the pertinent attack is a challenging task.
Anomaly based (i) Principal assumption: All intrusive activities are necessarily anomalous. (ii) Such a method builds a normal activity profile and checks whether the system state varies from the established profile by a statistically significant amount to report intrusion attempts. (iii) Anomalous
activities that are not intrusive may be flagged as intrusive. These are false positives. (iv) One should select threshold levels so that neither of the above two problems is unreasonably magnified nor the selection of features to monitor is optimized. (v)

Computationally expensive
because of overhead and possibly updating several system profile matrices.
Hybrid (i) Exploits benefits of both misuse and anomaly-based detection techniques. (ii) Attempts to detect known as well as unknown attacks.

==========
==========
==========

Barford, Paul, and David Plonka. "Characteristics of network traffic flow anomalies." Internet Measurement Workshop. 2001.

Recognizing and identifying anomalous behavior is often based on ad hoc methods developed from years of experience in managing networks. A variety of commercial and open source tools have been developed
to assist in this process, however these require policies and/or or thresholds to be defined by the user in order to trigger alerts. The better the description of the anomalous behavior, the more effective these tools become.

The first step in our project is to gather passive measurements of network traffic at the IP flow level. IP flow level data as defined in [1] is a unidirectional series of IP packets of a given protocol traveling between a source and a destination IP/port pair within a certain period of time.
While flow level data is certainly not as precise as passive measurements of packet level data, we demonstrate that it is sufficient for exposing many different types of aberrant network traffic behavior in close to real time. It also has the benefit of generating much smaller data sets than
packet level measurements which can become a significant issue in large, heavily used networks.
Our anomaly analysis process will be focused on precisely identifying both similarities and differences within each anomaly group. Our goal is not simply to cluster anomalies with similar statistical features but actually to characterize the features of each anomaly group rigorously.
Our analysis approach will employ a variety of tools including simple statistics, time series analysis and wavelet analysis to characterize anomaly features.

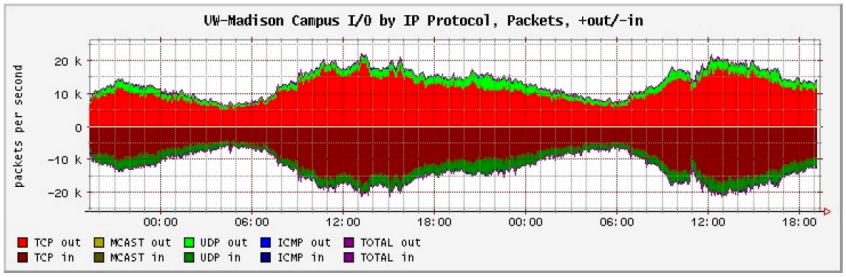


Fig. 2. Example of FlowScan output: Packet count per second broken down by protocol for a typical 48 hour period

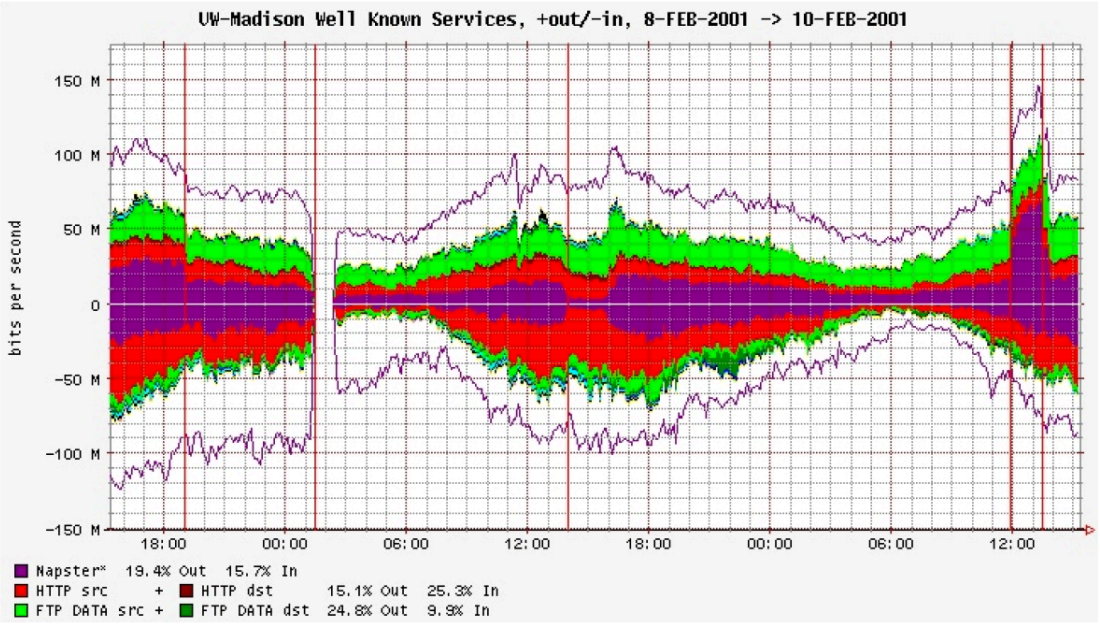


Fig. 3. Example of implementation of rate limits on Napster traffic

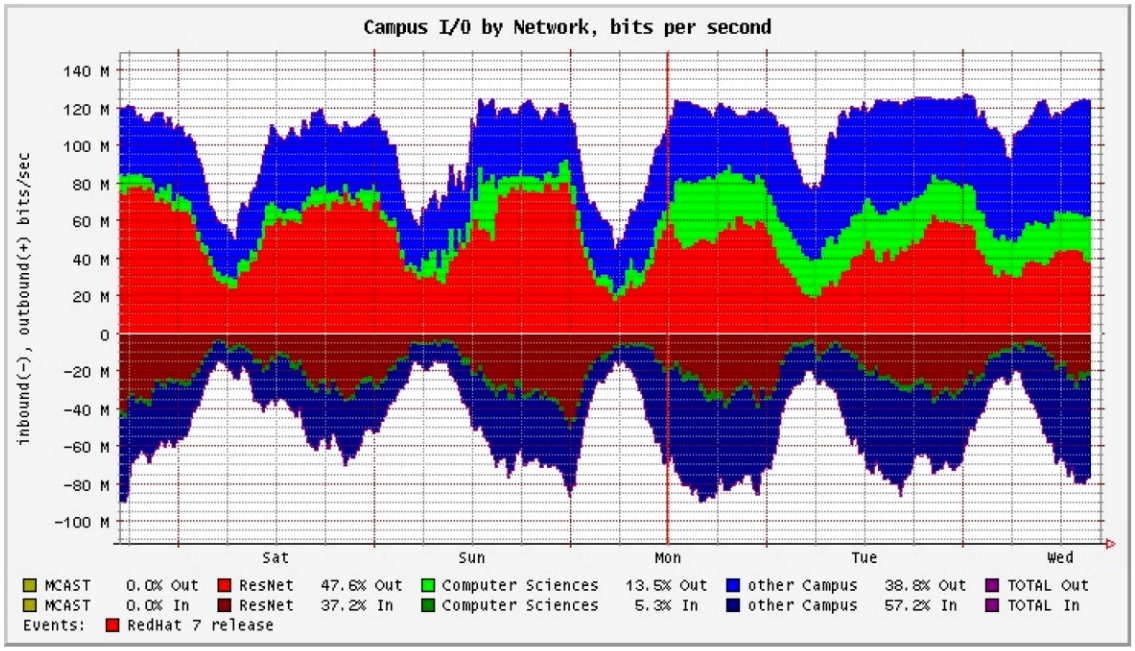


Fig. 4. The release of Linux Redhat 7.0: an example of flash crowd behavior

Network traffic properties have been intensely studied for quite some time. Examples of analysis of typical traffic behavior can be found in [4], [5]. More detailed characterizations and models of network traffic including the identification of self-similar properties can be found in [6], [7].

Visual analysis of traffic flow anomalies has lead to grouping anomalies into three general categories. These categories are useful for describing general anomaly characteristics however, they may or may not continue to be useful after we complete our characterization work.

Network Operation Anomalies: These include network device outages, significant differences in network behavior caused by configuration changes (e.g. adding new equipment or imposing rate limits) and plateau behavior caused by traffic reaching environmental limits. Anomalies in this category are distinguished visually by steep, nearly instantaneous changes in bit rate followed by bit rates which are stable but at a different level over a time period. An example of a network operation anomalies can be seen in Figure 3.

Flash Crowd Anomalies: In our environment, anomalies in this category are typically due to either a software release (e.g. UW is a RedHat Linux mirror site) or external interest in a Web site due to some kind of national publicity. Flash crowd behavior is distinguished by a rapid rise in traffic flows of a particular type (eg. FTP flows) or to a well known destination with a gradual drop off over time.

Network Abuse Anomalies: Two types of network abuse that can be identified using flows are DoS flood attacks and port scans. Network abuse anomalies are distinct from network operation and flash crowd anomalies in that they are not always readily apparent in bit or packet rate measurements. However, flow count measurements clearly indicate abuse activity with many distinct source address/port pairs since each

connection appears as a separate flow.

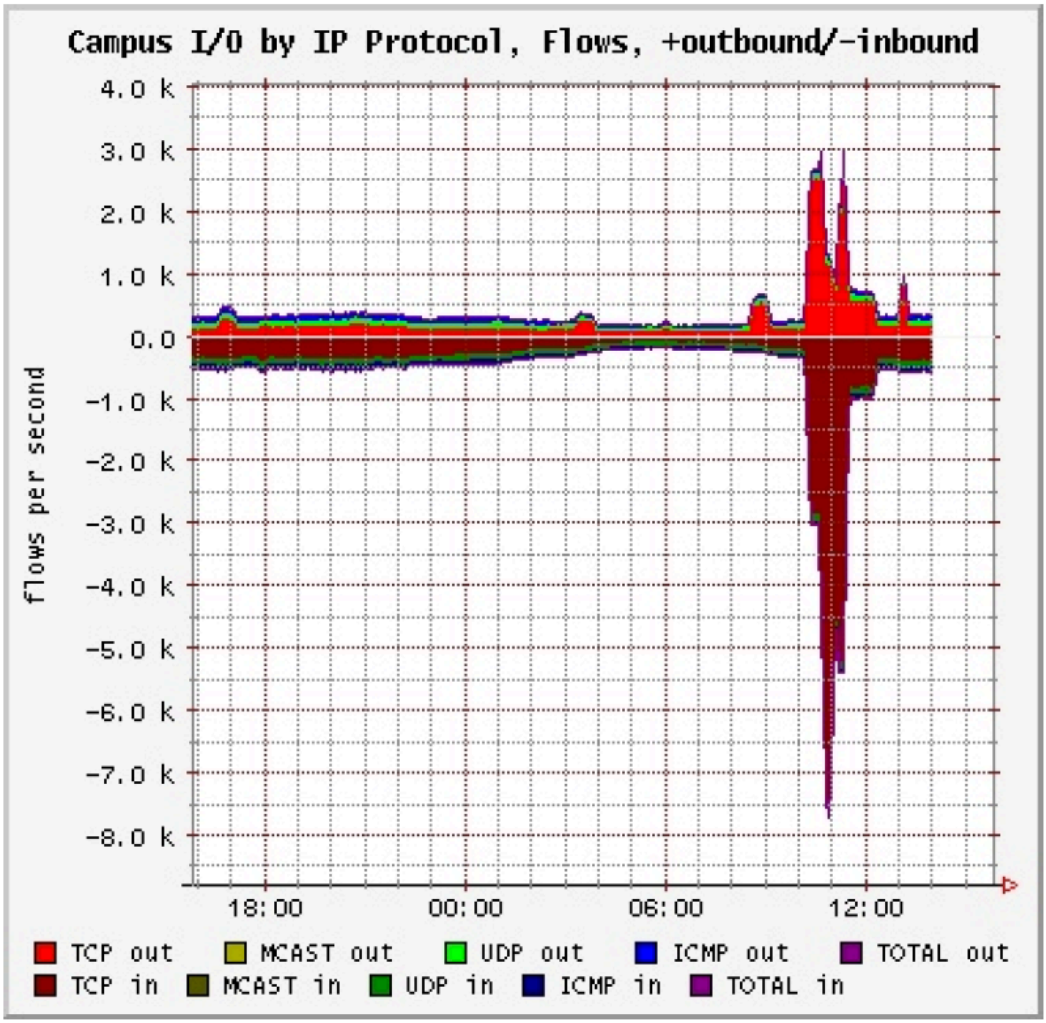


Fig. 5. An example of detecting a denial of service attack

[1] K. Claffy, G. Polyzos, and H.-W. Braun, “Internet traffic flow profiling,” Tech. Rep. TR-CS93-328, University of California San Diego, November 1989.
[4] R. C´aceres, “Measurements of wide-area Internet traffic,” Tech. Rep. UCB/CSD 89/550, Computer Science Department, University of California, Berkeley, 1989.
[5] V. Paxson, Measurements and Analysis of End-to-End Internet Dynamics, Ph.D. thesis, University of California Berkeley, 1997.
[6] V. Paxson and S. Floyd, “Wide-area traffic: The failure of poisson modeling,” IEEE/ACM Transactions on Networking, vol. 3(3), pp. 226–244, June 1995.
[7] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, “Self similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level,” IEEE/ACM Transactions on Networking, vol. 5, no. 1, pp. 71–86, February 1997.

==========
==========
==========
Jiang, Jun, and Symeon Papavassiliou. "Detecting network attacks in the internet via statistical network traffic normality prediction." Journal of Network and Systems Management 12.1 (2004): 51-72.

In this dissertation we present a network anomaly detection methodology, which relies on the analysis of network traffic and the characterization of the dynamic statistical properties of traffic normality, in order to accurately and timely detect network anomalies.
Anomaly detection is based on the concept that perturbations of normal behavior suggest the presence of anomalies, faults, attacks etc. This methodology can be uniformly applied in order to detect network attacks, especially in cases where novel attacks are present and the nature of the intrusion is unknown.
Specifically, in order to provide an accurate identification of the normal network traffic behavior, we first develop an anomaly-tolerant non-stationary traffic prediction technique, which is capable of removing both pulse and continuous anomalies.

In [19] the authors introduced a Resource-Allocating Network that allocates a new computational unit whenever an unusual pattern is presented to the network. This algorithm is suitable for sequential learning and is based on the idea that the number of hidden units
should correspond to the complexity of the underlying function as reflected in the observed signals. RAN [19] was developed to overcome the problem of NP-completeness in learning with fixed size network. I
In this section we introduce the RAN algorithm and describe in detail an anomaly-tolerant non-stationary traffic prediction algorithm based on the RAN algorithm.

RAN can be simplified as a single hidden layer network whose output response to sequential inputs is a linear combination of the hidden unit responses (figure 2.1).
Anomaly Violation Conditions and Detection Since network anomalies may present different characteristics that vary both in amplitude as well as time duration, in the following we define the anomaly detection based on the
above thresholds, as a combined function of both magnitude and duration. The reason of utilizing such a combination of both magnitude and duration, is to highlight the occurrence of anomalies that present a potential degradation in the network performance and behavior,
while minimize the flagging of temporary anomalies that are "non-threatening" for the network's performance.

When the traffic crosses the dynamic threshold, the violation magnitude and the corresponding time will be recorded and an anomaly detection alarm will be sent out if relation (2.17) is satisfied. This indicates that the network is under "anomaly" status and all time frames within the anomaly violation sliding window that are judged as being in anomaly status will be marked.

Attack/Anomaly Models
Two different attacks, namely the mail-bombing attack (scenario 1) and UDP flooding attack (scenario 2) were modeled and used throughout our evaluation. Internet "mailbombing" is the act of sending an extraordinarily large number of duplicate or random
messages via Internet electronic mail to a target whose account typically resides on a system other than the one the messages were originated from.
An UDP flooding attack repeatedly transmits a UDP packet to a specified port at the target machine for a specified period of time. The characteristics of the anomalous injected traffic for the scenario 2 attack ("UDP flooding" attack) are as follows: the repetition interval follows lognormal distribution with parameters (0.37,2.25) and the packet length is exponentially distributed with mean 1000 bits.

==========
==========
==========
Navaz, A. S., V. Sangeetha, and C. Prabhadevi. "Entropy based anomaly detection system to prevent DDoS attacks in cloud." arXiv preprint arXiv:1308.6745 (2013)
Cloud Computing is a recent computing model; provides consistent access to wide area distributed resources. It revolutionized the IT world with its services provision infrastructure, less maintenance cost, data and service availability assurance, rapid accessibility and scalability

Signature based IDS will perform poor capturing in large volume of anomalies. Another problem is that Cloud Service Provider (CSP) hides the attack that is caused by intruder, due to distributed nature; cloud environment has high possibility for vulnerable resources. By impersonating legitimate users, the intruders can use a service's abundant resources maliciously.
In Proposed System we combine few concepts which are available with new intrusion detection techniques. Here to merge Entropy based System with Anomaly detection System for providing multilevel Distributed Denial of Service (DDoS).

Entropy or Shannon-Wiener index is an important concept of information theory, which is a measure of the uncertainty or randomness associated with a random variable or in this case data coming over the network. If it was more random it contains more entropy. The value of sample entropy lies in range [0, logn]. The rate of entropy is lesser when the class distribution is pure i.e.it belongs to one class. The rate of entropy is larger when the class distribution is impure i.e. class distribution belongs to many class.
Hence comparing the rate of entropy of some sample of packet header fields to that of another sample of packet header fields provides a mechanism for detecting changes in the randomness. The entropy shows its minimum value 0 when all the items (IP address or port) are same and its maximum value logn when all the items are different. Use

change of entropy of traffic distributions (IP address, port) for DDoS detection.
you are interested in measuring the entropy of packets over unique source or destination address then maximum value of n is 232 for ipv4 address. If you want to calculate entropy over various applications port then n is the maximum number of ports.

The entropy H (X) of a random variable X with possible values {x1, x2…, xn} and distribution of probabilities P = {p1, p2, . . . , pn} with n elements, where 0 <= pi <= 1 and
Here p (xi) where xi belongs to X is the probability that X takes the value xi. Suppose it randomly observe X for a fixed time window w, then p (xi) = mi/m, where mi is the frequency or number of times it observe X taking the value xi

When to calculate probability of any source (destination) address then,
P(x) = Number of pkts with x as src (dst) address / Total number of packets
mi = number of packets with xi as source (Destination) address ,m = total number of packets

Here total number of packets is the number of packets seen for a time window T.
Similarly it can calculate probability for each source (destination) port as
P(x) = Number of pkts with X as src (dst) port / Total number of pkts

When to calculate with flow size
P(x) = Number of flows with flow size / Total number of flows.

Normalized entropy calculates the over all probability distribution in the captured flow for the time window T. Normalized entropy = (H / log n0)
If NE<th1, th1 is threshold value1, Mark flow as suspected and raise an alert Here nois the number of distinct xi values in the given time window

In a DDoS attack from the captured traffic in time window T, the attack flow dominates the whole traffic, as a result the normalized entropy of the traffic decreased in a detectable manner.

Projected Entropy: According to for a stochastic processes the entropy rate H (x) of two random processes are same:

$$H(\chi) = \lim_{n \to \infty} \frac{1}{n} H(x_1, x_2 \ldots x_n)$$

If H(x) <=th2, th2 is the threshold value2,Mark the flow as attacked, raise a final alert, discard the attack flow

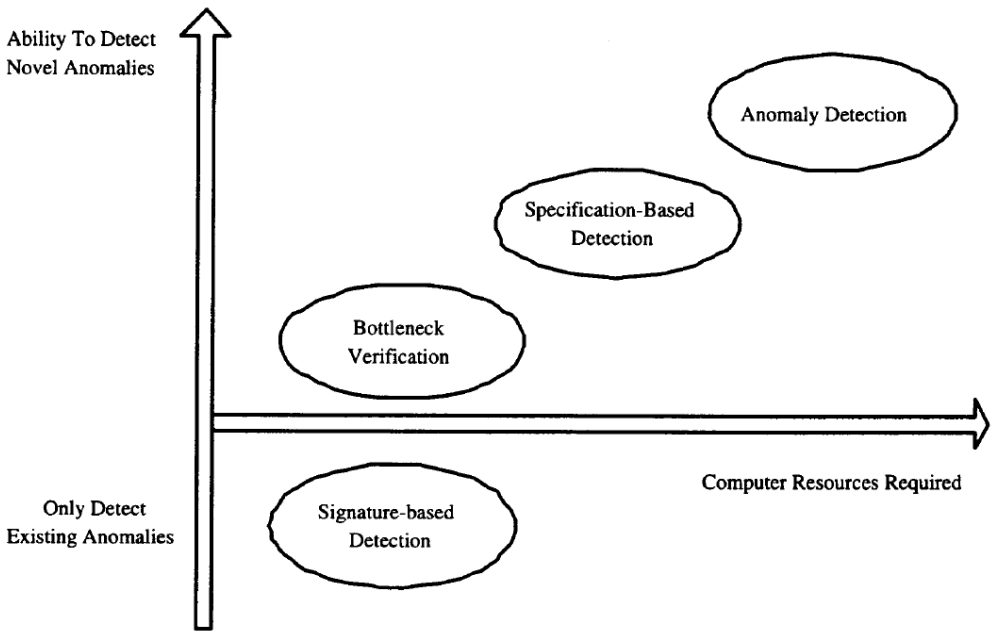
If H(x) <=th2, th2 is the threshold value2,Mark the flow as attacked, raise a final alert, discard the attack flow

To handle a large number of data packets flow in such an environment a multi-threaded IDS approach has been proposed in this paper. The multi-threaded IDS would be able to process large amount of data and could reduce the packet loss.
After an efficient processing the proposed IDS would pass the monitored alerts to a third party monitoring service, who would in turn directly inform the cloud user about their system under attack.

==========
==========
==========
Galtsev, Aleksey A., and Andrei M. Sukhov. "Network attack detection at flow level." Smart Spaces and Next Generation Wired/Wireless Networking. Springer,Berlin, Heidelberg, 2011. 326-334.
In this paper, we propose a new method for detecting unauthorized network intrusions, based on a traffic flow model and Cisco NetFlow protocol application. The method developed allows us not only to detect the most common types of network attack (DDoS and port scanning), but also to make a list of trespassers' IP-addresses.

Currently, Internet information resources are actively growing, penetrating many spheres of social life. Information technologies are being introduced not only into private enterprises, but also in the provision of public services. With each passing day, more and more confidential transactions are carried out via the Internet. In connection with these trends, the question of computer networks security is starkly raised. Attackers have developed and actively use many types of network intrusion [1,2,3,4], most of which can be prevented by standard methods of protection.
In recent years the rate of end-user connections to the Internet has increased sharply, which has given rise to an increase in the number and intensity of attacks such as DDoS.
Port scanning is used by hackers to conduct "network intelligence". In this article we

would like to propose a new method of detecting DDoS attacks and port scanning, based on the Cisco NetFlow protocol [7,8].



A network flow has been defined in many ways. The traditional Cisco definition is to use a 7-tuple key, where a flow is defined as a unidirectional sequence of packets sharing all of the following 7 values:

{ Source IP address
{ Destination IP address
{ Source port for UDP or TCP, 0 for other protocols
{ Destination port for UDP or TCP, type and code for ICMP, or 0 for other protocols
{ IP protocol
{ Ingress interface (SNMP ifIndex)
{ IP Type of Service

The proposed method for detecting network attacks based on the traffic flow model is described in [10]. Traffic models shows that two parameters, the load of the channel and the number of active flows in it, must be used for a full representation of the network state.

Two variables, the number of active flows and the utilization of the channel, best describe the current network state. Analysis of all data on the network, represented by individual points on the plane with axes, which are plotted the number of active flows and utilization of the network, allows to the definition of three areas that correspond to qualitatively
different states of the network.

The straight line corresponds to the part of the network that is characterized by a minimal loss of IP-packets, less than a half of one percent. The bent part of the curve corresponds to an overloaded network, and is characterized by a significant packet loss of up to 5%, which reduces the effective size of the
transferred segment of the TCP/IP. The third, nearly horizontal, portion of the curve corresponds to a completely unusable network with significant packet loss of over 5%. The distribution of total load tends to a normal (Gaussian) distribution, since the total load of the studied channel is the result of multiplexing a large number of flows that are independent of each other. The theoretical model allows us to estimate the confidence interval for the working area of the curve:
$B(t) = b(N + kA(exp)sqrt(N)$

Here A(exp) is the normal quantile function. Equation 1 indicates that the real state of the network, described by the number of active flows N and the flow data rate B(t), will be outside these limits in only 100% x exp of the total observation
time. The traffic model presented here allows the formulation of a simple criterion for finding anomalous network states: if several consecutive measurements go beyond the confidence interval of exp=0.05, we can confidently consider problems on the network. If we collect the data every 5 minutes, then the statistics of a few hours will make it possible to determine all the parameters of equation 1
with reasonable accuracy. Presumably, the network state will be out of the confidence interval during the progress of a network attack. During port scanning, the number of active

Flows will increase with a nearly constant load, as the data transmitted is only limited to establish the connection and to close it. The channel load as well as the flow number should sharply increase during the progress of DDoS attacks.
In order to prove these hypotheses it was decided to conduct two experiments with network scanning and with a DDoS attack.

Our studies have revealed patterns, based on the NetFlow data, that allow the IP addresses of the computers with which conducted DDoS attacks and port scans to be determined. Based on these patterns we developed an algorithm for attack detection. Before formulating the algorithm we will specify the format for
recording flow data:
{ Date and time of flow
{ Duration of flow (in seconds, up to thousandths)
{ Transfer protocol
{ IP address and source port
{ IP address and destination port
{ The number of transmitted packets
{ The number of bytes transferred
The algorithm developed for the detection of attacks such as DDoS and port scanning is:
1. Find IP-addresses of sources that generate a large number of flows.

(a) If the size of these flows is very short, up to 50 bytes, it is most likely port scanning.

(b) If the duration of the flows is greater then this IP-address might be carrying out DoS attacks.

2. Find IP-addresses of sources that generate very long streams (lasting more than 5 minutes). The IP-address assignment can be carried out DoS attack in this case.

If many IP-addresses from which a potential DoS attack are found, we may classify this attack as DDoS. Suspicious IP addresses are entered into the database and all trafic from those addresses are blocked by an iptables firewall [16] for 5 minutes. Iptables is installed on the protected server, i.e. only the server is protected, not the whole network.

Reference:

1. N. Paulauskas, E. Garsva, Computer System Attack Classi
cation, Electronics and Electrical Engineering, 2(66), 2006.
2. J. Mirkovic, P. Reiher, A taxonomy of DDoS attack and DDoS defense mechanisms, ACM SIGCOMM Comput. Commun., 34(2), 2004, 3953.
3. A. Hussain, J. Heidemann, C. Paradopoulos, A Framework for Classifying Denial- of-Service Attacks, Karlsruhe, Germany, 2003, 99110.
4. C. Douligeris, A. Mitrokotsa, DDoS Attacks and Defense Mechanisms: Classification and State-of-the-art, Comp. Networks, 44, 2004, 64366.
5. V. Paxson, An Analysis of Using Reectors for Distributed Denial-of-Service Attacks, CCR, 31(3), July 2001.
6. R.K.C. Chang, Defending against Flooding-based Distributed Denial of Service Attacks: A tutorial, IEEE Communications Magazine 40(10) (2002), 4251.
7. Cisco IOS NetFlow site, Cisco Systems, Available at: www.cisco.com/go/netflow.
8. B. Claise, NetFlow Services Export Version 9, RFC 3954, 2004.

==========

==========

==========

Vykopal, Jan, Tomas Plesnik, and Pavel Minarik. "Network-based dictionary attack detection." 2009 international conference on future networks. IEEE, 2009.

This paper describes the novel network-based approach to a dictionary attack detection with the ability to recognize successful attack. We analyzed SSH break-in attempts at a flow level and determined a dictionary attack pattern. This pattern was verified and compared to common SSH traffic
to prevent false positives. The SSH dictionary attack pattern was implemented using decision tree technique. The evaluation was performed in a large high-speed university network with promising results.

When we received a report that one of our computer was participating in a phishing scam, we started to analyze the compromised host. Finally, we found out someone broke in the host via secure shell (SSH). The administrator set up
a weak password1, thus a dictionary attack was successful. We also inspected other host logs in our network and concluded such dictionary attacks are very common. Moreover, we deployed new SSH server with a public IP address. Consequently,

we encounter new attacks almost every day. As this real example shows, although there are many advanced authentication schemes such as biometric or hardware tokens, a knowledge-based authentication is still widespread. Unfortunately, human chosen passwords are
inherently insecure since a large fraction of the users chooses passwords that come from a small domain. This enables adversaries to attempt to login to accounts by trying all possible passwords, until they find the correct one. This attack is known as a dictionary attack [6].

We analyzed real attacks in the following way. First, we inspected logs on attacked hosts and then identified appropriate traffic in NetFlow data collected at the border of a university network or its subnets. A flow is defined as a unidirectional sequence of packets with some common
properties that pass through a network device. Network flows are highly granular; for example, flow records include details such as IP addresses, packet and byte counts, timestamps, Type of Service (ToS), application ports, input and output interfaces, etc. [4] First of all, we searched for reconnaissance activities such as TCP SYN and FIN scanning that are invisible in access logs. Only one IP address that performed TCP SYN scanning before the dictionary attack was found. So we focused on this host and discovered other 93 592 TCP
SYN probes to 64 251 hosts in our class B network. In a nutshell, TCP scanning is not often preceding to SSH dictionary attacks.

Next, we focused on IP addresses of attackers in relevant
time windows known from host based analysis. We derived
the following dictionary attack pattern at NetFlow level:
- TCP port of the victim is 224, TCP port of the attacker is generally random and greater than 1024,
- many flows (tens or hundreds) from the attacker to the victim in a short time window (5 minutes),
- the flows from the attacker are small: from 10 to 30 packets and from 1 400 to 5 000 bytes,
- victims' responses are small too (typically the same number of packets and bytes),
- flow duration is up to 5 seconds,
- the last flow is different in case of successful attempt.

We also analyzed network traffic of various applications that utilize the SSH protocol to eliminate false positives. We perform the following scenarios in both Linux andWindows operating system: To sum it up, we did not find any traffic that fully corresponds
to the attack pattern derived from real dictionary attacks. That means there was observed no flow that meets all requirements above.

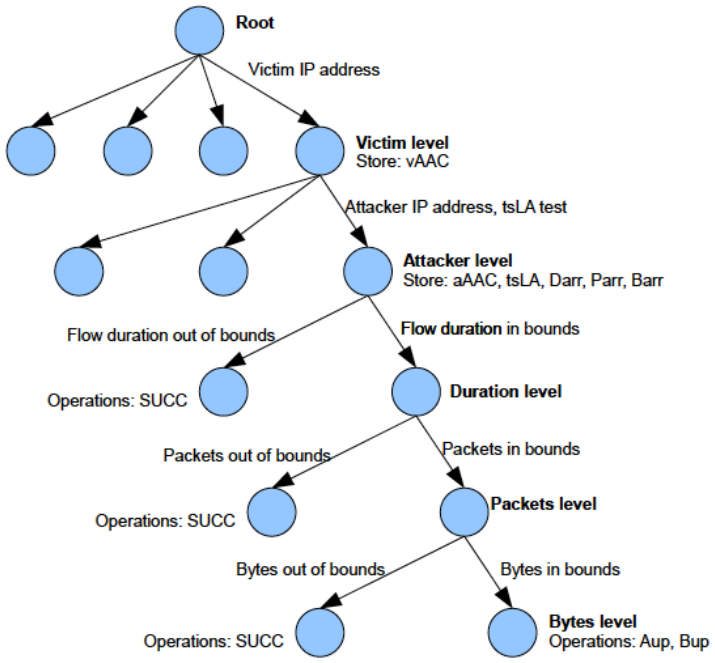


Figure 1. *SSH-D-tree* **diagram.**

Our SSH detection decision tree (SSH-D-tree) is presented in Figure 1. SSH-D-tree uses a subset of attributes of NetFlow record corresponding to SSH server response as the input data. This subset is formally a n-tuple: source ip address, destination ip address,
flow start, duration, packets, bytes. Each component of this n-tuple except of flow start forms a level of the SSH-D-tree. To avoid false positives, especially scans, we consider only responses having number of packets greater then certain constant.
Another false positives may raise from secured data uploading; therefore we use bytes per packet indicator in requests corresponding to given replies.
SSH-D-tree starts with a set of input n-tuples ordered by flow start and processes individual flows in sequence. The tree also starts with generic bounds for flow duration, number of packets and number of bytes transferred that fit to all attacks. For each pair (attacker; victim) arrays
of attack indicators are build (duration array, packets array and bytes array) until singleAttackAttempts threshold is reached. New bounds for given attacker and victim are calculated using toleration parameters afterwards.

SSH-D-tree uses two types of parameters. First type of parameters is static. These parameters specify namely the sensitivity of the detection procedure. The second type of parameters is called dynamic.

Static (initial)
– arraySize – length of attack indicator history influencing bounds for duration, packets, bytes
– singleAttackAttempts – number of attempts from single source to alert attack (length of learning process of the attack indicators between attacker and its victim)
– distributedAttackAttempts – number of attempts from all sources to alert attack
– certaintyReductionFactor – reduction factor of the success of attack
– tsDelta – maximum time difference between two attempts of attack from single source
– initialDurationBounds, initialPacketsBounds, initialBytesBounds – initial bounds of duration, numbers of packets and bytes in SSH server response
– deltaDuration, deltaPackets, deltaBytes – tolerance factors to calculate dynamically changing bounds of expected flow duration, number of packets and number of bytes transferred

Dynamic (calculated)
– vAAC, aAAC – victim/attacker attack counter.
– tsLA – last attack attempt time stamp.
– Daar, Paar, Baar – duration array, packets array, bytes array computed continuously during the attack for each pair (victim; attacker).

This section summarizes operations performed during the detection process except bounds check operations which are obvious:
- tsLA test (Last attack attempt time stamp test)
  – flow start > tsLA + tsDelta) drop whole subtree from the given attacker level and create a new one.
- SUCC (Successful attack report)
  – aAAC > singleAttackAttempts) mark attacker as successful and victim as compromised.
- Aup (Attack attempt update)
  – aAAC = aAAC + 1, vAAC = vAAC + 1.
  – aAAC > singleAttackAttempts) mark attacker as unsuccessful and victim as resisting or divide certainty of successful attacker and compromised victim by certaintyReductionFactor.
- Bup (Bounds update)
  – update arrays Daar, Paar, Baar according to actual values of flow duration, number of packets and number of bytes transferred. Calculate new bounds as the arithmetic mean of values in the array reduced by or increased by tolerance factor respectively.

From the performance point of view, our SSH-D-tree is able to process approximately 2 500 flows per second on COTS (cost of-the-shelf) hardware.

[4] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), Oct. 2004.
[6] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In CCS ’02:

Proceedings of the 9th ACM conference on Computer and communications security, pages 161–170, New York, NY, USA, 2002. ACM

==========

==========

==========

Munz, Gerhard, and Georg Carle. "Real-time analysis of flow data for network attack detection." 2007 10th IFIP/IEEE International Symposium on Integrated Network Management. IEEE, 2007

The operation of a computer network is a challenging task, especially if the network is connected to or part of the Internet. One reason is the increasing complexity of the Internet itself, which is characterized by the interconnection of lots of different devices and device types operated in multiple network domains and controlled by separate network management authorities.
As an important prerequisite for time-critical network operation tasks, many modern network devices support monitoring functions that offer monitoring data with low latency. The export of ow data is such a monitoring function that allows retrieving information about the traffic currently observed at a monitoring device, which may be a router or a stand-alone network monitor. A ow is dened as a unidirectional stream of IP packets that share a set of common properties; typically, the IP-five-tuple of protocol, source and destination IP addresses, source and destination ports is used.

While flow data is currently mainly evaluated for accounting purposes, it also carries valuable information for traffic analysis. It allows examining the traffic composition and drawing conclusions concerning the originating applications and services. It also enables discovering periodical changes and temporal trends. Furthermore, it can be used to detect and analyze traffic anomalies caused by network problems or illegitimate attack traffic. Especially in the latter case, it is important that the examination of the ow data occurs in a timely manner, enabling the network administrator to evaluate the detection results and to decide on appropriate countermeasures.

For this purpose, we designed and implemented a system called TOPAS (Traffic Ow and Packet Analysis System). TOPAS works as collector for ow data exported via Cisco Netow and/or IPFIX and provides a framework for user defined detection modules that simultaneously analyze the received data in real-time. Raw ow data usually requires some normalization and preprocessing in order to become appropriate input for detection algorithms. First of all, flows can be measured with different ow keys. If flow records with different ow keys are to be analyzed jointly, special care is necessary in order not to bias the analysis results. Another normalization issue concerns ow timestamps. Flow statistics such as octet and packet counters are typically

exported in conjunction with the timestamps of the first and the last counted packet. However, these timestamps do not necessarily indicate the actual beginning and end of a packet stream. They rather depend on the ow expiration and export policy deployed by the monitoring device. Generally, a flow is considered to be terminated if no more packets have been observed for a given timeout.

A common solution to achieve temporal normalization of flow data is to calculate octet and packet counters for predefined equally spaced time intervals. The start and end timestamps of a flow record do not necessarily fall into the same time interval used to
calculate a time-series value. More sophisticated algorithms rely on entropy measures calculated for specific feature value distributions such as port numbers and/or IP addresses [10], [11].

The main contributions of this paper are as follows:

- We describe the generic procedure of analyzing flow data for real-time anomaly and attack detection by decomposing it into three principal steps. These steps represent a generic model for all kinds of detection mechanisms, yet their actual implementations have to be adapted to the deployed detection algorithm (Section II).
- We present the functional properties of TOPAS and describe its system design. Performance measurements show that TOPAS is able to cope with high amounts of flow data.
- We explain how TOPAS is deployed in Diadem Firewall to detect DoS and DDoS attacks and identify the entry points of attack packets with spoofed source addresses

Different flow-based anomaly and attack detection approaches have been presented in literature. We can classify the most important ones into four categories:

- Threshold-based detection algorithms that rely on a prior knowledge such as predefined or adaptive thresholds for specic measures (e.g. [7], [12]).
- Principal component classifiers (PCC) that detect anomalies in multivariate time-series [8], [10], [11].
- Outlier detection algorithms that determine the similarity of a sample vector to the learned normal behavior [13].
- Rule learning algorithms that learn classification rules from training data containing labeled normal and attack data (e.g. [14]).

The computational complexity varies from very simple decisions (thresholds,

rules) to more complex distance calculations (outlier detection) and linear transformations (PCC). Nevertheless, all of them are practicable solutions for real-time analysis since we trigger the algorithms with the periodicity of the incoming time-series only.

Architecture and Implementation

A collector process receives the ow data from the monitoring devices supporting both Cisco Netow.v9 and IPFIX protocol. The received data is stored to a ring buffer located in a memory block that is shared by the collector and the detection modules.

A module manager running in a second thread is responsible for synchronization between the collector and the detection modules. It notifies the detection modules about newly arrived data and grants read access for a predefined period of time. After timeout or after all modules have signalized that they have read the data, the module manager frees the corresponding buffer segment and proceeds with the next block of data.

The detection modules are running in individual processes with two threads. The first thread (container) listens for notifications from the module manager, reads new ow data from the ring buffer, normalizes and preprocesses it according to the input requirements of the detection algorithm, and buffers the result.

The second thread empties the container at regular intervals and triggers the detection algorithm with the new input data.

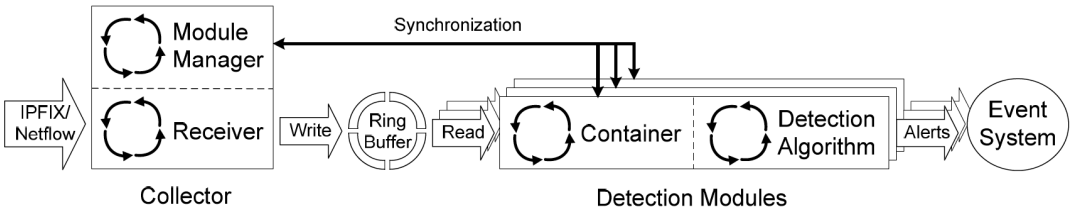


Fig. 2. TOPAS architecture

Realizing the detection module as separate processes has some performance drawbacks compared to a single process solution with multiple threads, but it also has clear advantages with respect to robustness and stability. As an example, it is possible to dynamically start and stop individual detection modules on demand during operation, and misbehaving detection modules can be easily stopped by killing the corresponding processes.

SYN Flood Detection Module: During a SYN flood attack, attackers are sending a

large number of TCP connection requests (SYN packets) to a single open port at the victim host. These packets have spoofed source addresses, so that the returned SYN/ACK packets do not reach any valid connection endpoint and thus remain unanswered in most cases. As a result, the victim host has to cope with many half-open TCP connections that are only released after a timeout. Hosts without specific protection mechanisms can only handle a limited number of half-open TCP connections, and if this limit is reached, new TCP connection requests cannot be answered any more.

In Diadem Firewall, SYN floods are detected with the SYNdog mechanism by Wang et al. [23], which is implemented in a detection module. SYN-dog compares the numbers of SYN and SYN/ACK packets that a host receives and returns respectively. Under normal conditions, the two numbers should be balanced since every SYN packet is answered by a SYN/ACK packet, apart from some SYN packets that are accidentally directed to a closed port. Consequently, a high number of unanswered SYN packets is an indication of an ongoing SYN flood attack.

Web Server Overloading Detection Module: Within Diadem Firewall, Olivier Paul developed a method to infer the nature of HTTP requests from passive traffic measurements based on probabilistic models [25]. The resulting information is used to trace the user behavior in an HTTP session and detect anomalous or aggressive request patterns using EWMA (Exponentially Weighted Moving Average) change detection algorithm. The inference mechanism as well as the detection algorithm have been implemented in a third detection module [26]. Evaluations showed that the detection method is fast and accurate in case of focused DoS attacks requesting a small number of objects on the web server [27].

A system for near real-time analysis of ow data is MINDS (Minnesota Intrusion Detection System) [9]. A collector receives Netow data and saves it in files covering consecutive time windows of 10 minutes each. An analyzer module processes the saved files and performs known attack detection and anomaly detection algorithms. Because of the batch mode processing of the files, it may last up to several minutes until an attack or anomaly is detected, which is too late for fast reaction.

References:
[7] C. Kotsokalis, D. Kalogeras, and B. Maglaris, .Router-based Detection of DoS and DDoS Attacks,. in Proc. of HP Openview University Association (HP-OVUA) 8th Annual Workshop, Berlin, Germany, June 2001.
[8] A. Lakhina, M. Crovella, and C. Diot, .Characterization of Network-Wide Anomalies in Trafc Flows,. in Proc. of 4th ACM SIGCOMM Conference on Internet Measurement. Taormina, Sicily, Italy: ACM Press, Oct. 2004, pp. 201.206.

[9] L. Ert¨oz, E. Eilertson, A. Lazarevic, P.-N. Tan, P. Dokas, V. Kumar, and J. Srivastava, .Detection of novel network attacks using data mining,. in Proc. of Workshop on Data Mining for Computer Security, to beheld in conjuction with IEEE International Conference on Data Mining, Melbourne, FL, USA, Nov. 2003.
[10] A. Lakhina, M. Crovella, and C. Diot, .Mining Anomalies Using Traffic Feature Distributions,. in Proc. of ACM SIGCOMM Conference, Philadelphia, PA, USA, Aug. 2005, pp. 217.228.
[11] J. Terrell, K. Jeffay, F. D. Smith, L. Zhang, H. Shen, Z. Zhu, and A. Nobel, .Multivariate SVD Analyses For Network Anomaly Detection,. in Proc. of ACM SIGCOMM Conference, Poster Session, Philadelphia, PA, USA, Aug. 2005.
[12] G. Androulidakis, V. Chatzigiannakis, M. Grammatikou, and F. Stamatelopoulos,.Network Flow-Based Anomaly Detection of DDoS Attacks, . in Proc. of Trans-European Research and Education Networking Association (TERENA) 2004, Rhodes, Greece, June 2004.
[13] A. Lazarevic, L. Ert¨oz, V. Kumar, A. Ozgur, and J. Srivastava, .A comparative study of anomaly detection schemes in network intrusion detection,. in Proc. of 3rd SIAM International Conference on Data Mining, May 2003.
[14] D. Barbar´a, J. C. S. Jajodia, L. Popyack, and N. Wu, .ADAM: Detecting intrusions by data mining,. in Proc. of IEEE Workshop on Information Assurance and Security, West Point, NY, USA, June 2001, pp. 11.16.

[23] H. Wang, D. Zhang, and K. G. Shin, .SYN-dog: Snifng SYN Flooding Sources,. in Proc. of 22nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, July 2002.
[25] O. Paul and J. E. Kiba, .RequIn, a tool for fast web trafc inference,. in 48th annual IEEE Global Telecommunications Conference (GLOBECOM 2005), Saint Louis, MO, USA, Nov./Dec. 2005.
[26] G. M¨unz, A. Fessi, G. Carle, O. Paul, D. Gabrijelcic, Y. Carlinet, S. Yusuf, M. Sloman, V. Thing, J. van Lunteren, P. Sagmeister, and G. Dittmann, .Diadem Firewall: Web Server Overload Attack Detection and Response,. Bordeaux, France, Dec. 2005.
[27] O. Paul, .Improving web servers focused DoS attacks detection,. in Proc. of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2006), Tuebingen, Germany, Sept. 2006.
==========
==========
==========
Al-Jarrah, Omar, and Ahmad Arafat. "Network Intrusion Detection System using attack behavior classification." 2014 5th International Conference on Information and Communication Systems (ICICS). IEEE, 2014.

This paper addresses Probes attacks or reconnaissance attacks, which try to collect any possible relevant information in the network. Network probe attacks have two types: Host Sweep and Port Scan attacks. Host Sweep attacks determine

the hosts that
exist in the network, while port scan attacks determine the available services that exist in the network. This paper uses an intelligent system to maximize the recognition rate of network attacks by embedding the temporal behavior of the attacks into a TDNN neural network structure. The proposed system consists of five modules: packet capture engine, preprocessor, pattern recognition, classification, and monitoring and alert module.

The most widely used classification for attacks in the research communities is the one adopted by K. Kendall [21]. This classification categorizes the computer attacks into:
- DOS Attacks: DOS attacks try to render the system or certain service unstable.
- User to Root: it tries to gain the root or admin privilege from normal user privilege.
- Remote to User: it tries to gain local account privilege for unauthorized entity.
- Probes: In this type of attacks, the attacker tries to collect any possible relevant information in the network. In this class, two famous types of attacks exist: Host Sweep and Port Scan Attacks. Host Sweep attacks determine the hosts that exist in the network,
while port scan attacks determine the available services that exist in the network.

The most well-known classification for IDSs is based on the Approach of Intrusion Detection. There are two common approaches, Anomaly detection and Misuse detection [12]. Anomaly Detection works on two phases: learning phase and detection phase. In the learning phase, it builds a system profile for the normal system parameters and states. While in the detection phase, the system compares the current system parameters with ones in the stored normal system profile, if they deviate in some way, an alert of intrusion is reported to the administrator or an appropriate action could be taken[1][3][4]. The most common advantage of Anomaly Detection is the ability to detect new and novel attacks.

This paper uses neural networks as a data processing technique. It shows that the temporal behavior of the attack can be represented in neural network structure with suitable design aspects. The detection process for the stored signatures will be very effective and fast, and the time taken to detect an attack will be constant. The temporal attack behavior is hardwired in the neural network structure to gain fast response and fast rate of attack detection. The system is characterized by high rate of attacks detection and low rate of false positives.

Host Sweep Attacks
Network attacks usually start by performing network discovery, which includes host sweep and port scan attacks. Host sweep attack determines the live hosts in

the network; the attacker tries to find a gateway to start with or enter from. So, intruders will be searching for weak points to compromise.

Port Scan Attacks

Port scanning tries to discover the running services on the victim machine, or tries to check the availability of certain service on the victim. It is well known that each network application running on the machine has a unique port number that it listens to such as port 80 for web browsing. By finding which services are running, a certain attack can be launched against the discovered service, e.g. a mail bomb attack can be launched against mail service to break it down.

- One host- different ports: The attacker scans different ports on a certain host, which is the typical behavior of the port scan. The order of ports is not important; scanning may be sequential or random.
- Different hosts-one port: The attacker scans multiple hosts at the same time with the same port number. This attack is launched against network of hosts looking for hosts running a certain service like DNS, SMTP, or HTTP. This type of scans is more complex than the standard attack.
- Different hosts-different ports: The attacker scans multiple hosts at the same time and each host is possibly scanned with different port. This is an advanced technique of port scanning that tries to hide its activity by initiating random port scans among random hosts. This attack is the most complicated port scan.

In this paper, we will consider seven port scan attacks including TCP SYN, TCP ACK, TCP SYN|ACK, TCP FIN, TCP NULL, TCP XMAS, and UDP/ICMP Error

The system continuously captures the packets on the medium using the packet capture engine module, which is also responsible for extracting relevant features that are required in the subsequent stages. The features are transferred to the next stage using two PIPEs: one for the host sweep and the other for the port scan.

The packet capture engine writes the extracted features to the PIPEs which will be read by the next component. The preprocessor reads these features from the PIPEs and process them. The preprocessor consists of two sub components: a host sweep preprocessor and a port scan preprocessor. The function of the preprocessor is to convert the sequential inputs which have a temporal relation to a set of patterns that can be processed simultaneously, i.e. it rearranges the features of the current and previous inputs to make it easier to detect the attacks. The preprocessing stage collects the features from the packet capture engine stage by reading the features from the read sides of the pipes.

The output of the preprocessor is directly connected to the pattern recognition component, which is responsible for detecting attacks using neural networks. The pattern recognition consists of two components, one for host sweep and the other

for port scan.
The alert and monitoring module is responsible for alerting the system administrator about any existing attacks or threats that exists in the network.

The pattern recognition neural networks use Principal component analysis (PCA) to produce the principle components which differ between host sweep recognition neural network and port scan recognition neural network. PCA is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components [18] [19].

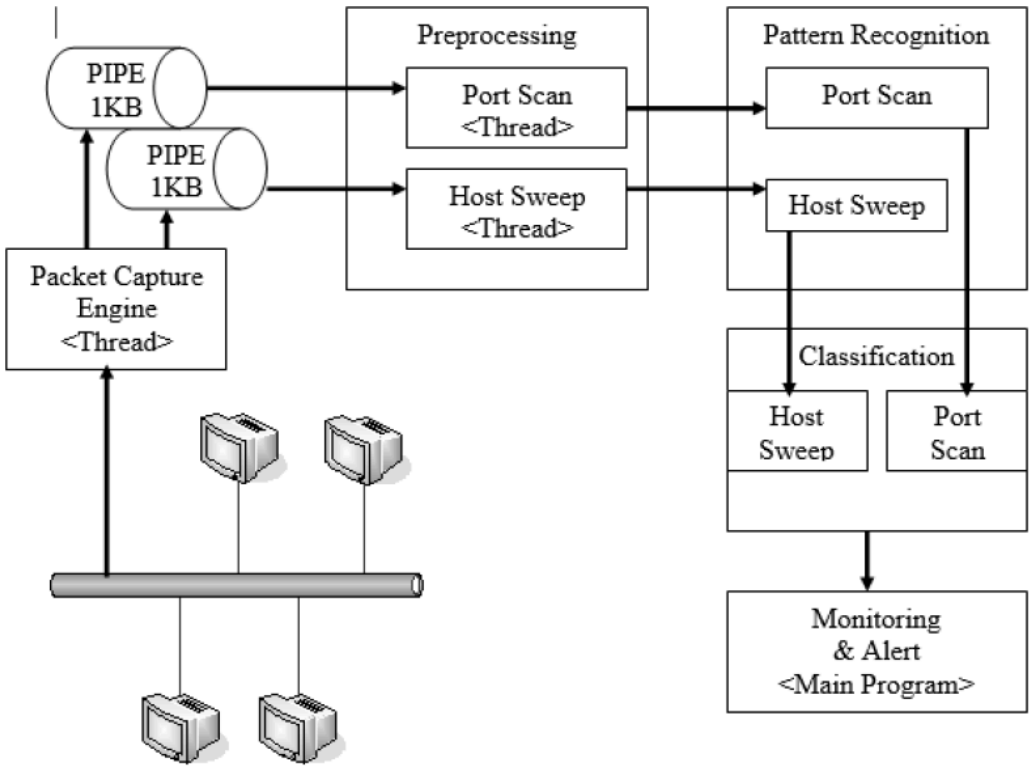


Fig. 1. System block diagram.

References:
[1] Anita K.jones, Robert S. Sielken. Computer System Intrusion Detection: a Survey. University of Virginia, Computer Science Department Technical Report; September 2000
[2] SNORT Network Intrusion Detection System. [Online] [accessed 2006]. Available from URL http://www.snort.org
[3] Sundaram Aurobindo. An Introduction to Intrusion Detection. http:// www.acm.org/crossroads/xrds2-4/intrus.html

[4] Przemyslaw kazienko, Piotr Dorosz. Intrusion Detection Systems (IDS)–Part 2-Classification, Methods and Techniques. IT FAQ 2003; 1(4):17-22.
[12] Rung-Ching Chen, Kai-Fan Cheng, and Chia-Fen Hsieh. Using Rough Set and Support Vector Machine for Network Intrusion Detection. International Journal of Network Security & Its Applications (IJNSA),Vol 1, No 1, April 2009.
[18] XLMiner, Data Mining Software online help. [Online] [accessed 2006]. Available from URL http://www.resample.com/xlminer/help/PCA/pca_intro.htm
[19] Erkki Oja. Unsupervised learning in neural computation. Theoretical Computer Science 2002; 287:187 – 207.
[21] Winpcap: packet capture tool. [Online] [accessed 2006]. Available from URL http://www.winpcap.org

==========

Wang, Ke, and Salvatore J. Stolfo. "Anomalous payload-based network intrusion detection." International Workshop on Recent Advances in Intrusion Detection. Springer, Berlin, Heidelberg, 2004.

We present a payload-based anomaly detector, we call PAYL, for intrusion detection. PAYL models the normal application payload of network traffic in a fully automatic, unsupervised and very effecient fashion. We first compute during a training phase a profile byte frequency distribution and their standard deviation of the application payload flowing to a single host and port. We then use Mahalanobis distance during the detection phase to calculate the similarity of new data against the pre-computed profile. The detector compares this measure against a threshold and generates an alert when the distance of the new input exceeds this threshold.
Some attacks exploit the vulnerabilities of a protocol, other attacks seek to survey a site by scanning and probing. These attacks can often be detected by analyzing the network packet headers, or monitoring the network traffic connection attempts and session behavior. Other attacks, such as worms, involve the delivery of bad payload (in an otherwise normal connection) to a vulnerable service or application. These may be detected by inspecting the packet payload (or the ill-effects of the worm payload execution on the server when it is too late after successful penetration).

The primary= design criteria and operating objectives of any anomaly detection system entails:
· automatic “hands-free” deployment requiring little or no human intervention,
· generality for broad application to any service or system,
· incremental update to accommodate changing or drifting environments,
· accuracy in detecting truly anomalous events, here anomalous payload, with low (or controllable) false positive rates,

· resistance to mimicry attack and
· efficiency to operate in high bandwidth environments with little or no impact on throughput or latency.

In this paper, the method we propose is based upon analyzing and modeling normal payloads that are expected to be delivered to the network service or application. These normal payloads are specific to the site in which the detector is placed. The system first learns a model or profile of the expected payload delivered to a service during normal operation of a system. Each payload is analyzed to produce a byte frequency distribution of those payloads, which serves as a model for normal payloads. After this centroid model is computed during the learning phase, an anomaly detection phase begins. The anomaly detector captures incoming payloads and tests the payload for its consistency (or distance) from the centroid model. This is accomplished by comparing two statistical distributions. The distance metric used is the Mahalanobis distance metric, here applied to a finite discrete histogram of byte value (or character) frequencies computed in the training phase. Any new test payload found to be too distant from the normal expected payload is deemed anomalous and an alert is generated. The alert may then be correlated with other sensor data and a decision process may respond with several possible actions. Depending upon the security policy of the protected site, one may filter, reroute or otherwise trap the network connection from being allowed to send the poison payload to the service/application avoiding a worm infestation.

We chose to consider “language-independent” statistical modeling of sampled data streams best exemplified by well known n-gram analysis. Many have explored the use of n-grams in a variety of tasks. The method is well understood, efficient and effective. The simplest model one can compose is the 1-gram model. Network payload is just a stream of bytes. Unlike the network packet headers, payload doesn’t have a fixed format, small set of keywords or expected tokens, or a limited range of values. Any character or byte value may appear at any position of the datagram stream. To model the payload, we need to divide the stream into smaller clusters or groups according to some criteria to associate similar streams for modeling. The port number and the length are two obvious choices. We may also condition the models on the direction of the stream, thus producing separate models for the inbound traffic and outbound responses.

Usually the standard network services have a fixed pre-assigned port number: 20 for FTP data transmission, 21 for FTP commands, 22 for SSH, 23 for Telnet, 25 for SMTP, 80 for Web, etc. Each such application has its own special protocol and thus has its own payload type. Each site running these services would have its own “typical payload” flowing over these services. Payload to port 22 should be encrypted and appear as uniform distribution of byte values, while the payload to

port 21 should be primarily printable characters entered by a user and a keyboard. Within one port, the payload length also varies over a large range. The most common TCP packets have payload lengths from 0 to 1460. Different length ranges have different types of payload. The larger payloads are more likely to have nonprintable characters indicative of media formats and binary representations (pictures, video clips or executable files etc.). Thus, we compute a payload model for each different length range for each port and service and for each direction of payload flow. This produces a far more accurate characterization of the normal payload than would otherwise be possible by computing a single model for all traffic going to the host. However, many centroids might be computed for each possible length payload creating a detector with a large resource consumption. To keep our model simple and quick to compute, we model the payload using ngram analysis, and in particular the byte value distribution, exactly when n=1. An ngram is the sequence of n adjacent bytes in a payload unit. A sliding window with width n is passed over the whole payload and the occurrence of each n-gram is counted. N-gram analysis was first introduced by [2] and exploited in many language analysis tasks, as well as security tasks.

For a payload, the feature vector is the relative frequency count of each n-gram which is calculated by dividing the number of occurrences of each n-gram by the total number of n-grams. The simplest case of a 1-gram computes the average frequency of each ASCII character 0-255. Some stable character frequencies and some very variant character frequencies can result in the same average frequency, but they should be characterized very differently in the model. Thus, we compute in addition to the mean value, the variance and standard deviation of each frequency as another characterizing feature.. So for the payload of a fixed length of some port, we treat each character's relative frequency as a variable and compute its mean and standard deviation as the payload model.

Mahalanobis distance is a standard distance metric to compare two statistical distributions. It is a very useful way to measure the similarity between the (unknown) new payload sample and the previously computed model. Here we compute the distance between the byte distributions of the newly observed payload against the profile from the model computed for the corresponding length range. The higher the distance score, the more likely this payload is abnormal. The formula for the Mahalanobis distance is:

$$d^2(x, \overline{y}) = (x - \overline{y})^T C^{-1} (x - \overline{y})$$

where x and y are two feature vectors, and each element of the vector is a variable. x is the feature vector of the new observation, and y is the averaged

feature vector computed from the training examples, each of which is a vector. And C-1 is the
inverse covariance matrix as

$$C_{ij} = Cov(y_i, y_j) \cdot \ y_i, y_j$$

are the ith and jth elements of the training vector.

The advantage of Mahalanobis distance is that it takes into account not only the average value but also its variance and the covariance of the variables measured. Instead of simply computing the distance from the mean values, it weights each variable by its standard deviation and covariance, so the computed value gives a statistical measure of how well the new example matches (or is consistent with) the training samples. In our problem, we use the "naïve" assumption that the bytes are statistically
independent. Thus, the covariance matrix C becomes diagonal and the elements along the diagonal are just the variance of each byte. Notice, when computing the Mahalanobis distance, we pay the price of having to compute multiplications and square roots after summing the differences across the byte value frequencies. To further speed up the computation, we derive the simplified Mahalanobis distance:

$$d(x, \overline{y}) = \sum_{i=0}^{n-1} (| x_i - \overline{y_i} | / \overline{\sigma_i})$$

where the variable is replaced by the standard deviation. Here n is fixed to 256 under the 1-gram model (since there are only 256 possible byte values). However there is the possiblity that the standard deviation is zero and the distance will become infinite. To avoid this situation, we give a smoothing factor a to the standard deviation similar to the prior observation:

$$d(x, \overline{y}) = \sum_{i=0}^{n-1} (| x_i - \overline{y_i} | / (\overline{\sigma_i} + \alpha))$$

References

2. M. Damashek. Gauging similarity with n-grams: language independent categorization of text. Science, 267(5199):843--848, 1995.

==========

Mantere, Matti, Mirko Sailio, and Sami Noponen. "Network traffic features for anomaly detection in specific industrial control system network." Future Internet 5.4 (2013): 460-473.

In this paper we analyze a possible set of features to be used in a machine learning based anomaly detection system in the real world industrial control system network environment under investigation. Intrusion Detection Systems have been studied for over three decades, but using such systems in an ICS environment is a relatively new topic. The first study of an IDS that uses anomaly detection is presented by Denning [4] in 1986. Her model uses statistics for anomaly detection and can be seen as foundation for most anomaly based IDS deployments today. In addition, the network security monitoring models presented by Heberlein et al.

3.3. Features

In the paper [2] a selection of 12 features useful for machine learning approach to network security monitoring in ICS network were discussed. We reviewed these potential features against the findings of the network traffic analysis with mixed results. Some of the features were applicable for this particular network, some decisively not, whilst the feasibility of some remained uncertain.

3.3.1. Throughput

Feature feasibility: Stability of throughput is typical for ICS networks. Changes in throughput can be used to detect high traffic events (scanning, DoS, malfunction). Attacks with low traffic requirements are not detectable with monitoring throughput.

3.3.2. IP Address— Port Pairs

Feature feasibility: In ICS networks that use static IP allocation, IP address—port pairs are expected to remain constant: appearance of new IP-port pair indicates a new service being started on the system containing the IP.

3.3.3. Average Size of Packets

Feature feasibility: Average size of packets is another good network behavior indicator. Many device level systems with high real time requirements send packets without extra buffering creating distinct average size statistics for these networks. In many networks the average size of packets can be considered to be a good indicator of normal behavior.

3.3.4. Timing

Feature feasibility: Packet timing and interval between packets from a network node is meaningful data for many ICS networks. The strict real time requirements of the system, especially in the device level of the network, make the timing statistics of the traffic stand out from typical application traffic.

3.3.5. Flow Direction

Feature feasibility: Flow direction indicates which system initiates the connection. For typical operation, the flow is initiated by one system, requiring a service, after which the amount of data sent from one system to the other is likely to be predictable, especially when the service in question is known. Variance in this behavior indicates abnormal behavior.

3.3.6. Flow Duration

Feature feasibility: The duration of flow is typically the TCP session length. It represents the amount of time, that a system or service requires to conduct it's network functions with it's destination system. As the number of services in industrial network nodes is limited, the duration of flows is deemed to have little variance.

3.3.7. Payload Form

Feature feasibility: Payload of packets for applications in a ICS environment are often strictly defined. The changes in payload form indicates a change in system behavior.

3.3.8. Payload Data

Feature feasibility: Payload data can be used to detect misconfigured systems or malicious actions. Payload abnormalities are a good sign for detection. The usage of default user credentials should be avoided in an ICS network. This enables the detection of systems trying out default credentials as a first line of attack. Detection also exposes poorly configured legitimate systems.

3.3.9. MAC—IP Mapping

Feature feasibility: MAC to IP mapping can be used in any LAN to detect changes in hardware components. The appearance of new MAC indicates that new hardware has appeared in the network. While MAC addresses can be forged, they are still usable for impersonation detection. It also helps the ICS network operator to be aware of all legitimate hardware in the system.

3.3.10. Network Protocol

Feature feasibility: The protocols used in the ICS network should be strictly defined and limited. Appearance of new protocols in the traffic indicate serious change in the network. If the networking protocol is identified using only port number, as is typical with Wireshark, the detection may be incorrect. Many systems use non-standard port addresses to conduct their network activity.

3.3.11. Protocol Settings

Feature feasibility: The protocols settings used in ICS networks are typically static, selected to guarantee the best performance of the ICS network. Monitoring the protocol settings of used protocols will detect misconfigured services and malicious activities

3.3.12. Connectivity Number

Feature feasibility: The number of connections for different systems is very static in most ICS networks. This depends widely on the role of the node.

The features that were investigated against the results of the network trace analysis did show that some of the initially thought features could have been useful, while some would definitely not have worked in this particular environment. Interestingly the average packet size in the factory side of the network appeared the most promising, with very little variance present. When the longer trace was split in to 600 s intervals and analyzed, the average packet size appeared nearly

constant 135 bytes.

References:
4. Denning, D. An intrusion-detection model. IEEE Trans. Softw. Eng. 1987, SE-13, 222–232.
==========
==========
==========

Sperotto, Anna, et al. "An overview of IP flow-based intrusion detection." IEEE communications surveys & tutorials 12.3 (2010): 343-356.

The goal of this paper is to provide a survey of current research in the area of flow-based intrusion detection. The survey starts with a motivation why flow-based intrusion detection is needed. The concept of flows is explained, and relevant standards are
identified. The paper provides a classifcation of attacks and defense techniques and shows how flow-based techniques can be used to detect scans, worms, Botnets and Denial of Service (DoS) attacks.

NOWADAYS hackers are continuously attacking networked systems; in fact, it would be interesting to investigate if there are still Internet users who have not been victim of an attack yet. Considering the damage caused by the attacks (billions of U.S. dollars) [1], it is important to detect attacks as soon as possible, and take, if feasible, appropriate actions to stop them. This task is particularly challenging due to the diversity in form (information gathering, password stealing, viruses, Trojan horses, Denial of Service (DoS)...) attacks exhibit.

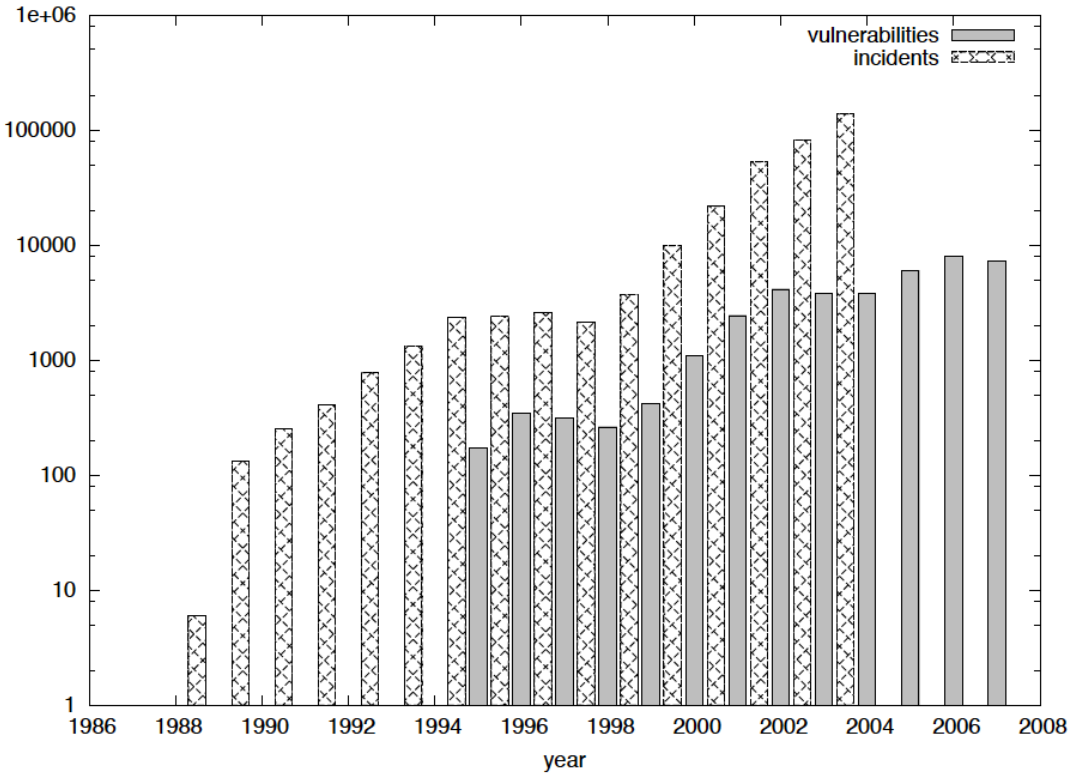


Fig. 1. Trends in incidents and vulnerabilities (logarithmic scale).

For the detection of network attacks, special systems have been developed; these systems are called Network Intrusion Detection Systems (NIDS). In an attempt to find known attacks or unusual behavior, these systems traditionally inspect the contents (payload) of every packet [2], [3]. The problem of packet inspection, however, is that it is hard, or even impossible, to perform it at the speed of multiple Gigabits per second (Gbps) [4], [5]. For high-speed lines, it is therefore important to investigate alternatives to packet inspection. One option that currently attracts the attention of researchers and operators is flow based intrusion detection. This paper provides a survey of current research in the area of flow-based intrusion detection.

Well known systems like Snort [2] and Bro [3], exhibit high resource consumption when confronted with the overwhelming amount of data found in today's high-speed networks [18]. In addition, the spread of encrypted protocols poses a new challenge to payload-based systems. An example is the work of Taleb etc al [19], [20], where the authors propose an intrusion detection systems based on per-packet inspection that rely only on header information in order to identify misuses in encrypted protocols.

Given these problems, flow based approaches seem to be a promising candidate for Intrusion Detection research.
In literature, several definitions of an IP [?]ow can be found [8], [24], [9]. This article follows the definition of flow as it was described by the IPFIX (IP Flow Information Export) working group within IETF [10], [22]:
“A flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties.”

Several attack classifications have been described in literature [37]. These classifications usually distinguish between the following basic categories [38], [39]:
• Physical attacks: attacks based on damaging the computer and network hardware.
• Buffer overflows: attacks that gain control or crash a process on the target system by overflowing a buffer of that process.
• Password attacks: attacks trying to gain passwords, keys, etc. for a protected system.
• (Distributed) Denial of Service attacks: an attack which leads to situations in which legitimate users experience a diminished level of service or cannot access a service at all.
• Information gathering attacks: an attack that does not directly damage the target system, but gains information about the system, possibly to be used for further attacks.
• Trojan horses: a program disguised as a useful application,
which deliberately performs unwanted actions.
• Worms: a program that self-propagates across a network. Self-propagation is the characteristic that differentiates worms from viruses (see below). A worm spread can be extremely fast: an example is the Sapphire/Slammer worm, which is known to have infected 90% of the vulnerable hosts in 10 minutes [40].
• Viruses: a virus is regarded as a worm that only replicates on the (infected) host computer. Hence, it needs user interactions to propagate to other hosts.

- Nowadays, an additional threat has evolved pertaining Botnets. Botnets are groups of computers “infected with malicious program(s) that cause them to operate against the owners’ intentions and without their knowledge”, as defined in

Lee etc al[42]. Botnets are remotely controlled by one or more bot master. Moreover, Botnets are the perfect infrastructure for setting up and supporting any kind of distributed attack, such as, for example, DoS attacks and SPAM campaigns.

In particular, the research community currently provides approaches to detect the

following classes of attacks:
- Denial of Service;
- Scans;
- Worms;
- Botnets.

TABLE I
CATEGORIZATION OF THE PROPOSED SOLUTIONS ACCORDING TO THE TAXONOMY.

System	Detection Method	Behaviour on detection	Usage Frequency	Data processing	Data collection
Li *et al.* [51] Gao *et al.* [52]	anomaly	active	real-time	centralised	distributed
Zhao *et al.* [54]	not spec	not spec	real-time	centralized	distributed
Kim *et al.* [55]	misuse	passive	real-time	centralized	distributed
Münz *et al.* [56]	compound	passive	real-time	centralized	distributed
Lakhina *et al.* [58], [59], [60], [61]	anomaly	passive	real-time	centralized	centralized
Wagner *et al.* [65]	anomaly	passive	real-time/batch	centralized	centralized
Gates *et al.* [67]	misuse	passive	batch	centralized	centralized
Stoecklin *et al.* [68]	anomaly	passive	batch	centralized	centralized
Dübendorfer *et al.* [69] [70]	compound	passive	real-time	centralized	centralized
Collins *et al.* [75]	anomaly	passive	real-time	centralized	centralized
Dressler *et al.* [74]	misuse	passive	real-time	centralized	centralized
Karasaridis *et al.* [76]	misuse	passive	real-time	centralized	centralized
Livadas *et al.* [44] [77]	not spec	passive	batch	centralized	centralized
Gu *et al.* [78]	anomaly	passive	real-time	centralized	centralized

DDOS

Detection of Denial of Service is often addressed in flow based intrusion detection. These attacks, by their nature, can produce variations in the traffic volume that are usually still visible at flow scale. Unfortunately, it is almost impossible to directly detect semantic DDOS attacks,i.e., attacks in which the service interruption is caused by the payload contents. Lets consider the Ping of Death In such attack, the attacker sends malformed or otherwise malicious ping packets, which causes the victim system to crash. Since this attack does generate a single ICMP flow, the attack would most likely go undetected. There would be, indeed, no change in flow frequency and intensity.

There are two main examples of anomaly-based DoS detection in high-speed networks, using flow information only. The work of Li etc[51] and Gao etc al [52], in the first place, approach the problem using aggregate flow measures collected in appropriate data structures, named sketches. A sketch is originally a one-dimensional hash table suitable for fast storing of information: it mainly counts occurrences of an event. A simple example of the use of sketches in DoS attacks is the detection of SYN Flooding attacks [53], as
described in Gao etc [52]. In this case, the sketch is supposed to store, for each time frame and each tuple (dest_IP, dest_port), the difference between the number of SYN packets and the number of SYN/ACKs. If the stored value for the current time deviates from the expected one, a DoS SYN Flooding attack is going on.

A similar approach is proposed by Zhao etc al [54]. In this case, a data-streaming algorithm is used to filter part of the traffic, and identify IPs that show an abnormal

number of connections. The authors consider both the case in which a host is the source of an abnormal number of outgoing connections (large fan out), as well as the case in which a host is the destination of an unusual number of connection attempts (large fan in). The first case matches the definition of a scanning host, while the second is used for detecting DoS victims.

Scans

Scans are usually characterized by small packets that probe the target systems. Keeping this characteristic in mind, it is easy to imagine that scans can easily create a large number of different flows. There are three categories of scans: (i) a host
scanning a specific port on many destination hosts (horizontal scan); (ii) a host scanning several ports on a single destination host (vertical scan); (iii) a combination of both (block scan). Irrespectively of the kind of scan, the result will be a variation of the flow traffic in the network. In literature, scans have generally been investigated by considering their most obvious characteristic: the scanning source shows an unnaturally high number of outgoing connections. The problem has been approached in this way by Zhao etc al [54], already cited in the previous section. Looking at host behavior from an incoming/outgoing connection perspective allows addressing DoS and scan attacks as faces of the same problem: hosts with an inadequate and unusual fan-in/out. Similarly, Kim etc al [55] attempt to describe a scan in terms of traffic patterns, as already explained in the case of DoS. The authors differentiate between network (horizontal) scans and host (vertical) scans.

The approach described in Wagner etc al [65] is not related to traffic volume anomalies. In this case, the probabilistic measure of entropy is used to disclose regularity in connection-based traffic (flows). Wagner etc al created an efficient analysis procedure based on compression of sequences of network measurements. They observe that, in the case of a scanning host, the overall entropy in a specific time window is subdued to a change. In particular, the presence of many flows with the same source IPs (the scanning host) will lead to an abrupt decrease of the entropy in the distribution of the source IP addresses. At the same time, the scanning host will attempt to contact many different destination IPs on (possibly) different ports, generating an increase in these entropy measurements. The combined observation of multiple entropy variations helps in validating the presence of an attack.

Botnets

As explained in Section IV, Botnets consist of infected hosts (bots) controlled by a central entity, known as master (or bot-master). As a fact, many Botnets used to rely on IRC channels, which can be identified at flow level, as described in the work of Karasaridis etc al [76]. The authors propose a model of IRC traffic that does not rely on specific port numbers. Karasaridis etc address two main points.

First, they propose a multistage procedure for detecting Botnets controllers. Starting from reports of malicious activity obtained from diverse sources (e.g., scan logs, spam logs, and viruses), the authors identify groups of flows involved in suspicious communications (candidate controller conversations). These conversations may happen between a host and a candidate server (controller) that use either an IRC port (e.g., 6667, 6668 or 7000) or that hides the control traffic using a different protocol. In the second case, the candidate conversation is checked against the flow model. The second aim of Karasaridis etc al is, once the controllers have been identified, to group the suspected bots into behavioral groups, i.e., clusters of bots that show the same activity pattern.
For scans and DoS, current research aims at real time identification, with alerts that permit the network administrator to intervene as soon as possible. In the case of Botnets, only long time observations can lead to the identification of the bots and controller.fi

References

[1] Computer Economics, “2007 malware report: The economic impact of viruses, spyware, adware, botnets, and other malicious code,” Jul. 2008. [Online]. Available: http://www.computereconomics.com
[2] M. Roesch, “Snort, intrusion detection system,” Jul. 2008. [Online]. Available: http://www.snort.org
[3] V. Paxson, “Bro: a system for detecting network intruders in real-time,” Computer Networks, vol. 31, no. 23–24, pp. 2435–2463, 1999.

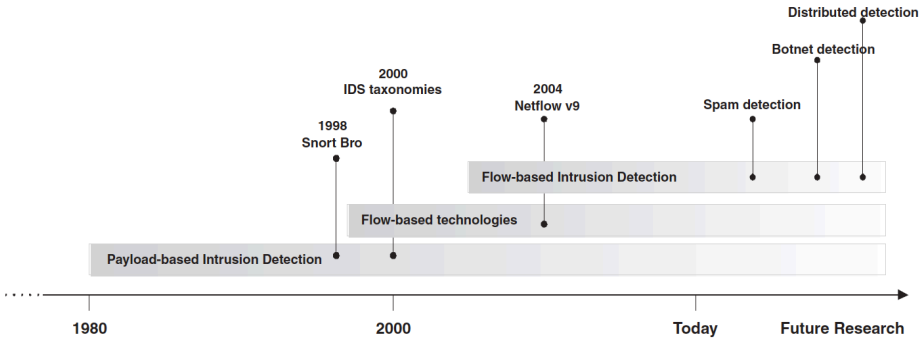


Fig. 8. Time line of evolution of intrusion detection and flow-based technologies.

==========

Strayer, W. Timothy, et al. "Botnet detection based on network behavior." Botnet detection. Springer, Boston, MA, 2008. 1-24.

==========
==========
==========
Mahoney, Matthew V., and Philip K. Chan. PHAD: Packet header anomaly detection for identifying hostile network traffic. 2001.

We developed an anomaly detection algorithm (PHAD) that learns the normal ranges of values for each packet header field at the data link (Ethernet), network (IP), and transport/control layers (TCP, UDP, ICMP). PHAD uses the rate of anomalies during training to estimate the probability of an anomaly while in detection mode. If a packet field is observed n times with r distinct values, there must have been r "anomalies" during the training period. If this rate continues, the probability that the next observation will be anomalous is approximated by r/n.
To consider the dynamic behavior of real-time traffic, PHAD uses a non stationary model while in detection mode. In this model, if an event last occurred t seconds ago, then the probability that it will occur in the next one second is approximated by 1/t. Often, when an event occurs for the first time, it is because of some change of state in the network, for example, installing new software or starting a process that produces a particular type of traffic. Thus, we assume that events tend to occur in bursts.
During training, the first anomalous event of a burst is added to the model, so only one anomaly is counted. This does not happen in detection mode, so we discount the subsequent events by the factor t, the time since the previous anomaly in the current field. Thus each packet header field containing an anomalous value is assigned a score inversely proportional to the probability, scorefield = tn/r

Finally, we add up the scores to score the packet. Since each score is an inverse probability, we could assume that the fields are independent and multiply them to get the inverse probability of the packet. But they are not independent. Instead, we use a crude extension of the stationary model where we treat the fields as occurring sequentially. If all the tn/r are equal, then the probability of observing k consecutive anomalies in a nonstationary model is (r/tn)(1/2)(2/3)(3/4)...((k-1)/k) = (1/k)r/tn. This is consistent with
the score ktn/r that we would obtain by summation.
where anomalous fields are the fields with values not found in the training model. In PHAD, the packet header fields range from 1 to 4 bytes, allowing 28 to 232 possible values, depending on the field. It is not practical to store a set of 232 values for two reasons. First, the memory cost is prohibitive, and second, we want to allow generalization to reasonable values not observed in the limited training data. The approach we have used is to store a list of ranges or clusters, up to some limit C = 32. If C is exceeded during training, then we find the two closest ranges and merge them. For instance, if we have the set {1, 3-5, 8}, then merging the two closest clusters yields {1-5, 8}. There are other possibilities, which we

discuss in section 5, but clustering is appropriate for fields representing continuous values, and this method (PHAD-C32) gives the best results on the test data that we used.

PHAD examines 33 packet header fields, mostly as defined in the protocol specifications. Fields smaller than 8 bits (such as the TCP flags) are grouped into a single byte field. Fields larger than 4 bytes (such as the 6 byte Ethernet addresses) are split in half. The philosophy behind PHAD is to build as little protocol-specific knowledge as possible into the algorithm, but we felt it was necessary to compute the checksum fields (IP, TCP, UDP, ICMP), because it would be unreasonable for a machine learning algorithm to figure out how to do this on its own. Thus, we replace the checksum fields with their computed values (normally FFFF hex) prior to processing.