INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2010; **00**:1–24

A scalable, efficient and informative approach for anomaly-based Intrusion Detection Systems : theory and practice

Osman Salem[1], Sandrine Vaton[2] and Annie Gravey[2]

[1] *Laboratoire d'Informatique Paris Descartes (LIPADE)*
Université Paris Descartes
Paris, France
[2] *Département Informatique*
TELECOM Bretagne
Brest, France

SUMMARY

In this paper, we present the design and implementation of a new approach for anomaly detection and classification over high speed networks. The proposed approach is based first of all on a data reduction phase through flow sampling by focusing mainly on short lived flows. The second step is then a random aggregation of some descriptors such as a number of SYN packets per flow in two different data structures called Count Min Sketch and Multi-Layer Reversible Sketch. A sequential change point detection algorithm continuously monitors the sketch cell values. An alarm is raised if a significant change is identified in cell values. With an appropriate definition of the combination of IP header fields that should be used to identify one flow, we are able not only to detect the anomaly but also to classify the anomaly as DoS, DDoS or flash crowd, network scanning and port scanning. We validate our framework for anomaly detection on various real world traffic traces and demonstrate the accuracy of our approach on these real-life case studies. Our analysis results from online implementation of our algorithm over measurements gathered by a DAG sniffing card are very attractive in terms of accuracy and response time. The proposed approach is very effective in detecting and classifying anomalies, and in providing information by extracting the culprit flows with a high level of accuracy. Copyright © 2010 John Wiley & Sons, Ltd.

KEY WORDS: Intrusion Detection System, Network anomaly detection, Change Point Detection, Multi-chart Cumulative Sum, DoS, Sketch

1. INTRODUCTION

The daily available patches and updates for servers, and the filtering of malicious packets, offer a significant protection from known vulnerability attacks. A talented attacker can still bypass these defenses by detecting and exploiting new vulnerabilities in the latest software releases. This is a subtle attack that requires a lot of skills and efforts on the part of the attacker, and it is not very common. There are much easier ways to deny service and to silence any web service. While an experienced attacker may use botnets of tens of thousands of compromised hosts with one of the popular DDoS bots (Agobot, SDBot, RBot, SpyBot, etc.), an inexperienced attacker may rent compromised machines from available web sites. The intentions behind these attacks often differ, and range from revenge, vandalism, political reasons up to money extortion.

In DoS attacks, the target machine (or network) spends all of its critical resources (such as bandwidth, CPU time, memory, etc.) on handling the attack traffic and cannot provide service for legitimate clients. With DDoS, security threats for computer network availability have increased significantly. Recent DDoS attacks in August 2009 were performed against Twitter, Google and Facebook. Twitter was driven offline for 3 hours by the attacks, and there was degraded service (slow down) for the users of Facebook. This time the reason was political and directed against an individual who blogged about the independence of a breakaway region of Georgia. Attackers wanted to silence this blogger and they did. Flooding can easily lead to the disruption of critical infrastructure services and degrades the QoS in ISP networks. Legitimate traffic cannot find resources, and gets dropped because of the high attack volume. Therefore, an effective detection of anomalies requires the ability to separate the malicious traffic from legitimate traffic, that is to say that some additional information is necessary, for example some information about the victim servers, the attackers and the type of ongoing attack (DoS, scanning, etc.). This information is needed in order to take the appropriate countermeasures and protect the access for normal users.

Intrusion detection systems (IDS) classify network traffic based on some classification rules. They are divided into two categories: misuse based and anomaly based IDS. A misuse based IDS is based on signatures, and looks into the packets for a matching of the predefined attack signatures (e.g. Snort [45], Bro [43]). It raises an alert when a suspicious activity has been identified. Like an antivirus, misuse based systems require a regular update of their signatures database to detect recent attacks. Although these methods are very efficient in detecting known attacks, they cannot analyze the content of encrypted data. Moreover a zero day attack (security hole without available fix/signature) does not have an available signature, and consequently cannot be detected. On the other hand, anomaly based IDS identify deviations from normal traffic patterns as anomalies. These methods try to detect changes in some traffic descriptors at the flow level (e.g the number of half open connections, requests, etc.) based on some measurements of the same parameters in some past intervals. Any inconsistent deviation in the value of these parameters is considered as an anomaly. The main advantage of anomaly detection based approaches is their independence from any prior knowledge of intrusion signatures, so that such systems may be able to detect new types of attacks. Their drawback is that they need a learning phase without attacks, and that they are not able to detect attacks that do not change the traffic pattern.

The meaning of anomaly is sometimes misunderstood, since anomaly does not necessarily mean malicious, and on the other hand a malicious behavior may not provoke any visible anomaly in the traffic. For example, both DDoS and flash crowds trigger an abnormal increase in the number of SYN towards one destination, but DDoS are malicious attacks aiming at shutting down the victim server, whereas a flash crowd is caused by legitimate users requests. We consider the problem of anomaly detection over high speed links, such as SYN flooding, network & host reconnaissance, and worm propagation leading to significant changes in some measurable network characteristics when compared to the normal behavior. These attacks send a large number of malicious requests towards the victim server/network. As the majority of attacks today are performed using TCP [41] by exploiting its handshake procedure it is primordial to detect these attacks (e.g. SYN flooding attacks) at an early stage of their occurrence, and especially before exceeding the limit of half-open connections of the victim server. It is expected that an early detection will provide sufficient time for defense reaction, such as filtering, pushback and traceback.

In this paper we design a scalable and efficient framework for anomalies detection and classification over high speed links. The proposed framework considers the online detection of some abrupt changes in time series related to some aggregated numbers of SYN. The analysis is performed at the flow level, as flow level monitoring permits a considerable compression of the information. A flow (as defined by

Netflow or IPFIX [17]) is a set of unidirectional packets sharing the same value of 5-tuple in their IP headers: Source IP (SIP), Destination IP (DIP), Source Port (SP), Destination Port (DP), and Protocol (P). To detect anomalies over high speed links, our first step is to reduce the amount of input data through flow sampling, in order to focus mainly on malicious flows. Indeed, most of the time the network traffic is normal and the operation of analyzing all the collected data is extremely resource consuming: it consumes for example bandwidth for report transmission to Network Operation Center (NOC), and CPU time for processing the data in order to identify and extract suspect flows. In this paper we describe a Network based Intrusion Detection System (NIDS) and not a Host Based Intrusion Detection System (HIDS). A record is built for each flow at measurement points. Some of these records are exported to NOC. The procedure of selection of the records is a stochastic sampling procedure. The aim of the sampling procedure is to focus mainly on short-lived flows, to reduce the bandwidth used for exporting the records , and to reduce the CPU load induced by records processing at NOC.

The data reduction phase is based on the observation that short-lived flows are the source of many attacks (DDoS, PortScan, NetScan, worm spreading, etc.). During our experiments and analysis of real IP traces, we indeed checked for example that many non spoofed DoS attacks carry different source port (SP) values in their TCP segment headers so that each packet generates a small flow that is observed in the monitoring process. In real life, IP address spoofing remains a security issue for the attacker, who does not want to loose the control of bots, or be caught after bots identification.

The proposed sampling procedure reduces the large amount of raw data, and aims to improve the detection of anomalies and to reduce the false alarms by focusing on short-lived flows. However, even after flow level filtering, maintaining information for each active flow in the selected subset is a cost prohibitive approach over a high speed link. The memory and CPU requirements induced by the large state space of traffic flow identifiers (e.g. IP addresses) are not adequate for a real time analysis. It is not surprising that many existing anomaly detection algorithms have been applied on time series of some aggregated counters (e.g. # SYN packets) in one measurement point [53, 54, 49, 57]. Aggregating the counters over all the flows does not permit to trace back the culprit flows and thus limits considerably the possibility of countermeasures. As this does not reveal any information about attacker/victim for mitigation the interest of deploying such monitoring infrastructure is rather limited. Moreover the application of change point detection algorithms to the aggregated traffic tends to be inaccurate in finding attacks without generating too many false alarms since many attacks induce only small variations with respect to the whole traffic volume. On the contrary our goal is to identify network anomalies such as DoS or scans in near real-time manner, and to extract the culprit flows for further mitigation.

In response to the scalability limitations of maintaining some information such as SIP, DIP, SP, DP or some combination of these fields for each active flow, an efficient data structure based on $k - ary$ hash tables (Fig. 1), called sketch [16, 32, 37], was proposed and used to handle large state spaces, with a small amount of memory requirement and a constant computational (update/query) complexity. A sketch is a multi-stage Bloom filter based on random aggregations of counters in shared cells, where flows identifiers (denoted by keys) are hashed to index into a set of cells in different stages using k different hash functions, usually chosen to reduce collision effects, to uniformly distribute keys, and to reduce correlations between the hash functions. To use sketch in the context of network anomalies detection, IP flows are typically classified by some combination of fields in their packet header, such as destination IP address (DIP), or source and destination IP address (SIP—DIP), destination port (DP), etc. This flow identifier is a key used to update each of the hash tables by a value which is a reward associated with the key, for example a number of SYN packets. In our paper the selection of which combination of the IP header fields will define a flow is decided in such a manner that it is possible to

recognize some particular types of anomalies, such as DoS, DDoS, flash crowds, alpha flows, worms, network scans or port scans. Authors in [21] have shown that a random aggregation of flow related counters in a sketch does not significantly disrupt flow variations. Moreover they introduced the Count-Min Sketch (CMS) algorithm [21], which returns an approximate value of the counter of a given key, i.e. the accumulated value for a key. In our proposed architecture we will use a multi-stage Bloom filter or sketch structure similar to the one used in the CMS algorithm.

The proposed framework is based on detection of change points in the cells of the CMS data structure, which aggregates multiple data streams from high speed links in the stretched database. To detect significant changes in the sketch cell values, we use the Multi-chart Non-Parametric CUSUM algorithm (MNP-CUSUM [51, 31]) over the time series inside all cells of the sketch. Each time series is an accumulated number of SYN packets for all flows indexing into a particular cell. The time series are monitored by a MNP-CUSUM algorithm which goal is to identify that a change has occurred in one of the cells, and also in which cell it has occurred. MNP-CUSUM was selected because of its optimality properties in terms of false alarm rates, delays, and false localization probabilities.

With an appropriate definition of which IP header fields define a "flow' an anomaly (DoS, DDoS, port or network scanning, etc.) results in an abrupt increase of the times series in some cells of the CMS data structure. This abrupt change is detected and the cell index values are identified by the MNP-CUSUM algorithms. In order to permit the mitigation of the ongoing anomaly some identifier of the culprit flows is necessary. Therefore, an additional Multi-Layer Reversible Sketch (MLRS) is introduced and used in parallel to the CMS structure for software efficient sketch inversion, in order to extract bad flows after the detection of the anomalies.

The functional operation of the system is the following: we proceed to filter flow-record data by focusing on small flows only, and we update the counters of two compact sketches (MLRS and CMS) for discrete time interval T. Afterward, MNP-CUSUM algorithm is used to check the presence of buckets which value deviates significantly from normal behavior. After the detection of anomalies by the CUSUM algorithm, we recover the key associated to cells with raised alarm by CUSUM in MLRS, through exploiting the cell index to recover responsible flows identifier, and we achieve verification through count-min query of alarm value for suspect key over the CMS sketch.

The main contribution of our approach is to combine skilfully different components into a complete framework for Intrusion Detection Systems (IDS). The proposed IDS has many attractive properties: scalability, small delay as well as low False Alarm Rate (FAR), and a rich and accurate output information (attack type and instant, IP addresses and port numbers, etc.). The complete framework inherits from the good properties of its main component blocks : the reversibility of the MLRS permits retrieving useful information about the ongoing attacks, the MNP-CUSUM makes it possible to detect the anomalies with a low delay and a low FAR, the CMS is designed in order to avoid that collisions generate false alarms. The performance of the complete framework is extensively assessed on several real-life traffic traces including online experimentations with well-known attack types and instants.

The remainder of this paper is organized as follows. Section 2 presents some related works about anomaly detection. In Section 3, we briefly review two important building blocks in our algorithm : the CMS data structure and the MNP-CUSUM anomaly detection algorithm. Section 4 describes our complete framework for anomaly detection and classification over high speed networks. In Section 5, we present the analysis results from the application of the proposed framework over real Internet traces, and we discuss its effectiveness in terms of true detection, false alarm rate and accuracy of the output. Finally, Section 6 presents concluding remarks.

2. RELATED WORKS

Many important contributions have been proposed to undermine anomalies in network traffic. We can cite for example Haar-wavelet analysis [39], entropy based methods [42] , sequential change point detection methods with the CUSUM algorithm [49, 54, 51], adaptive threshold analysis [12], exponentially weighted moving average method (EWMA) [57], Holt-Winters seasonal forecasting based methods [11], data reduction techniques with sketches [32, 37], SNMP MIB Support Vector Machine (SVM) analysis [58], Principal Component Analysis (PCA) [33, 28], etc.

When early approaches for anomaly detection were focused on the definition of models able to represent the traffic pattern, other advanced works aggregate the whole stream of packets in one time series, and apply a change point detection algorithm to detect the instant of anomaly occurrence [49, 54]. The latter have a good performance in terms of spatial and temporal complexities, but present the drawback of aggregating all the traffic in one flow, especially over high speed networks, where low intensity attacks cannot be detected with such a method. Furthermore, discovering the time instant of an attack occurrence without any additional information about the malicious source or victim is not enough to react against the attack. Usually, the amount of traffic is huge, and manual search/extraction of the malicious flows is a difficult operation. Therefore, to increase the accuracy of these methods, and to uncover the victim or attacker and classify the detected anomalies, several approaches have been proposed in the literature [37, 59, 24, 47]. However, the applicability of such on-line approaches for packet processing requires FPGA equipment [47].

Schweller *et al.* in [47, 48] propose the use of random aggregation counters for more fine grained detection. To discover the victim of flooding, they propose a method based on Galois Field $GF(2^l)$ for mangling and for simplifying sketch inversion. The proposed method is hardware efficient, and has been implemented in FPGA. Bu *et al.* in [13] propose an extension to the previous method through sequential hashing to reduce the complexity of previous sketch inversion methods. Feng *et al.* in [24] propose a method based on XOR operator and linear algebra for sketch inversion. In this paper, we will briefly show another method for reversing sketch through the use of an additional 2D table and RC4 stream ciphers.

All these proposed approaches have been used either to detect the heavy hitter flows (most frequent flows) or to detect an abrupt deviation between two discrete intervals via a simple comparison. Many different methods have been used in order to uncover anomalies in traffic flows. In-house methods do not have optimality properties and suffer from many shortcomings such as false alarms, instabilities, sensitivity to the training period, etc. [44]. In this paper we will use the multi-chart non parametric CUSUM algorithm [51] over sketch in order to uncover changes. Indeed the optimality property of CUSUM algorithms is translated in practice into less false alarms and smaller detection delays than in-house methods.

We are not the only ones who detect anomalies in traffic with CUSUM based approaches. Indeed the CUSUM algorithm has been used in order to detect a variety of different security problems (mainly DoS/DDoS and worms) from traffic inspection. For example, Wang *et al.* [54, 53, 55] detect SYN flooding and DDoS attacks. Wang *et al.* in [54] aggregate the whole traffic in one flow, and use a non parametric version of CUSUM for detecting TCP SYN flooding. They consider different metrics such as number of SYN, FIN and SYN/ACK in CUSUM for detecting flooding attacks. Siris *et al.* in [49] evaluate and compare two anomaly detection algorithms (adaptive threshold and CUSUM) for the detection of TCP SYN flooding. The result of the comparison shows that CUSUM is more efficient for detecting low intensity attacks than adaptive threshold. Lim *et al.* [38] implement SYN flooding detection methods on a programmable network processor. He *et al.* [26] focus on available bandwith

estimation and DDoS detection. As the vulnerabilities of wireless communication protocols are the vectors of many attacks today, some papers use in their detection mechanism the CUSUM algorithm : Lee *et al.* [35] detect DoS attacks on 3G wireless networks ; Yan *et al.* (2009) [56] detect Bluetooth worm propagation with CUSUM and GLR. CUSUM has also been used by several authors in order to detect worm propagation : Bo *et al.* (2005) [9] focus on worm outbreaks and SYN flooding, Bu *et al.* [14] detect scan-based worms in darknets. Darknets traffic has been studied by different authors : Ahmed *et al.* [4, 5] detect nested anomalies in darknet traffic with a sliding-window mechanism and CUSUM. Tartakovsky *et al.* (2007) have introduced the now celebrated multi-chart CUSUM [51] and initially applied it to the monitoring of production networks (packet size analysis). The same author (Tartakovsky *et al.*, 2006, [50]) has compared the performance of decentralized distributed change detection methods. Kang *et al.* study botnet detection with entropy based multi-chart CUSUM [29]. As attacks against encrypted protocols such as SSH and SSL evade signature-based IDS, statistical approaches such as CUSUM are particularly useful in that case ; Fadlullah *et al.* (2007) [23] detect attacks against ciphered protocols with CUSUM.

As the main problem is the huge amount of data to process, many researchers have tried to reduce the size of collected data before processing, through packet filtering and sampling. The most famous example is NetFlow [18] in CISCO IOS, which can be configured with uniform packet sampling. Packet sampling has been widely studied, some works have investigated how packet sampling impacts the precision of anomaly detection algorithms [10, 40], and others propose various sampling techniques to improve accuracy in anomaly detection.

As many packet sampling methods change the characteristic features of traffic flows, it is easy to miss flows with a small number of packets, and to incorrectly estimate the flow size, especially during the identification of mice and elephant flows. In general, if the flow is built from sampled packets, the precision of the anomaly detection system depends on the sampling rate. Kawahara *et al.* [30] show that packet sampling degrades the results of anomaly detection algorithms. Hohn *et al.* [27] compared packet sampling with flow sampling and found that flow sampling outperforms packet sampling in recovering flow size distributions. Mai *et al.* [40] present the impact of random packet sampling and random flow sampling on anomaly detection. Results revealed that both degrade anomaly detection. However, random flow-sampling outperforms random packet sampling, because packet sampling introduces a bias that degrades the detection effectiveness and increases the number of false alarms.

Recently, Androulidakis *el al.* in [7, 6] designed and analyzed selective flow sampling in order to improve the accuracy of anomaly detection algorithms. They investigate the impact of sampling on the performance of non parametric CUSUM and entropy based anomaly detection, and they prove that selective flow sampling achieves ”magnification” of the anomalies. This sampling method has inspired the sampling algorithm used in this paper.

The approach presented in this paper uses sampling to reduce the amount of data and to discard unpredictable variations of legitimate traffic. Afterward, it uses the sequential MNP-CUSUM over sketch for anomaly detection thus allowing us to detect changes with a small delay and a low false alarm rate. A new software efficient approach for sketch inversion through encryption and index exploitation is proposed to provide information about victim/attacker. An appropriate definition of which IP header fields define a flow makes it possible to classify the anomalies by categories (DoS, DDoS, network or port scanning, etc.).

After the seminal papers by Cormode and Muthukrishnan [19, 20, 21] a few methods which combine the use of sketches and time series analysis methods have been published [34, 46, 22, 15]. Lakhina *et al.* in [34] improve the performance of their Principal Component Analysis (PCA) based system by

inserting the use of sketch structures. Analysis of PCA methods performance in the discovery of traffic anomalies have revealed that these approaches are sometimes prone to false alarms and misdetections [44]. Indeed PCA methods are data-driven and require a particularly long anomaly free training phase in order to calibrate the decomposition basis. This leads to a risk of incorporating some of the anomalies of the training dataset into the decomposition basis. Some authors combine the use of sketch and a multiresolution (or multiscale) analysis based on wavelets in order to undermine traffic anomalies [22, 15]. Wavelet analysis is used to detect discontinuities (irregular patterns) in traffic data. One of the advantages of wavelet analysis is that it does not need a training phase with anomaly free traffic. Dewaele *et al.* analyze a 6 years long traffic trace on a trans-Pacific backbone link (MAWI dataset). They discover a large number of irregular patterns; some are due to known anomalies but the method also reveals a large number of unexpected flows, be their nature legitimate or not remaining still an open issue. Example of ”anomalous” flows can be elephant flows (HTTP traffic, FTP or SSH connections), or destination IP addresses receiving small number of packets from a large number of source IP addresses (P2P traffic). The key used in order to index the cells in the sketch data structure in [22, 15] is a destination IP address (DIP) or a source IP address (SIP). In [36] the authors also detect changes in high-dimensional traffic data. They use a non parametric change detection test based on the U statistics. Two different dimension reduction techniques are used: TopRank which is based on record filtering and HashRank which is based on random aggregation.

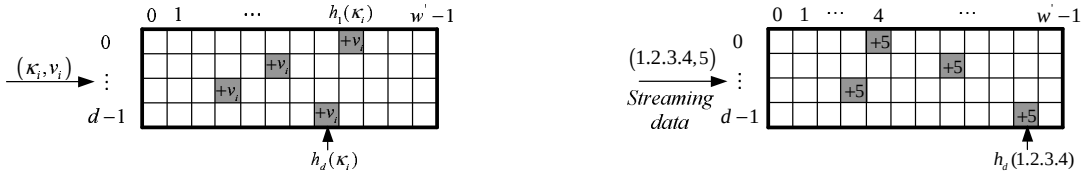
Our method combines sketch data structures (CMS and reversible sketch) and the MNP-CUSUM to discover in real-time significant changes in the number of SYN packets associated to some ”flows”. Flows are associated to some particular fields in the IP and TCP headers : source IP address (SIP), destination IP address (DIP), source port (SP), destination port (DP) or a combination of those fields. There are a number of advantages to using our algorithm compared to other published algorithms. MNP-CUSUM algorithm is particularly simple and computationally efficient compared to decomposition methods such as PCA or wavelet analysis. Moreover CUSUM is sequential by nature which makes it adapted to on-line treatments. The optimality properties of the CUSUM have been theoretically established (low false alarm rate, low delay) and this translates into practice into fewer raised false alarms than decomposition based methods. We recognize that MNP-CUSUM is less sensitive to subtle irregularities in traffic than wavelet based analysis and thus reveals less ”anomalies” in traffic than [22]. Using a non-parametric version of the CUSUM algorithm (NP-CUSUM) makes the algorithm robust against non stationarities in traffic. Another very important feature of our algorithm is that it reveals precise information about the ongoing attack. Many methods raise alarms but then a manual inspection of the traffic is often necessary in order to analyze the traffic and to classify the anomaly (DoS, scan, false alarm, etc.) Our algorithm automates the classification step by using appropriate combination of TCP/IP header fields as keys to index cells in the sketch data structures. Our algorithm moreover pinpoints malicious flows as it includes a sketch reversion step in order to recover the value of the keys from the index of anomalous cells.

3. THEORETICAL BACKGROUND

In this section, we briefly survey the underlying Count-Min Sketch (CMS) data structure and Multi-chart Non-Parametric CUSUM (MNP-CUSUM) used in our framework.

3.1. Count-Min Sketch

Cormode and Muthukrishnan [21] introduced another kind of multi-stage Bloom filter called Count Min Sketch (CMS). The advantage of the proposed algorithm is to provide an estimate of the associated counter with a key in the stretched data structure (CMS). Let $S = s_1 s_2 \dots s_n$ be the set of input stream, where each item $s_i = (\kappa_i, \nu_i)$ is identified by a key $\kappa_i \in U$, drawn from a fixed universe of items U. $\nu_i \in \mathbb{R}$ is the reward associated with each key. For example, with $\nu_i = \#SYN$ and $\kappa_i =$ DIP, the goal will be to count the number of SYN packets corresponding to the different destination IP addresses. The sketch data structure is made up of d hash tables. The arrival of an item with key κ_i increments its associated counter in the j^{th} hash table by ν_i ($C_{j,h_j(\kappa_i)} += \nu_i$), as shown in Figure 1. The update procedure is realized by d different hash functions, chosen from the set of 2-universal hash functions $h_j(\kappa_i) = \{((a_j \kappa_i + b_j) \bmod P_U) \bmod w'\}$, to uniformly distribute κ_i over hash tables and to reduce collisions. The parameter P_U is a prime number larger than the maximum number in the universe, where Mersenne prime numbers of the form $2^i - 1$ are generally chosen for fast implementation. a_j and b_j are random integers smaller than P_U, with $a_j \neq 0$. To highlight the use of 2-universal hash function in the context of IP address ($P_U = 2^{61} - 1$) let us take κ_i equal to $1.2.3.4$, with random values of $a_j = 2$ and $b_j = 3$, and a sketch width $w' = 256$, the result of hash function is $h(1.2.3.4) = 4$. This means that the associate bucket in the first line of the sketch is the number 4.



(a) Sketch structure.

(b) Update with $\kappa_i = 1.2.3.4$ & $\nu_i = 5$.

Figure 1. Sketch data structure.

The Count-Min point query returns an estimate of the counter for a given key κ_i as the minimum of the corresponding d counter values : $\hat{s}_k(\kappa_i) = \min_{0 \leq j < d} \{C[j][h_j(\kappa_i)]\}$.

The definition of the accuracy of the result of a CMS query is probabilistic. The distance between the estimated and true values of the count should be lower than a precision ϵ with an error probability at most δ : $\mathbb{P}(d(\hat{s}, s) \leq \epsilon) \geq 1 - \delta$. With this constraint, the parameters d (number of pairwise independent universal hash functions) and w' (number of hash values) should be chosen as follows : $d = \lceil \ln(1/\delta) \rceil$ and $w' = \lceil e/\varepsilon \rceil$ where e is the base of the Neperian logarithm. Thus, it maintains modest storage requirements of $\ln(1/\delta) \times (1/\varepsilon)$ count cells.

Flow records from collected data traces, can be classified into series of (κ_i, v_i), where κ_i can be the concatenation of DIP and DP ($DIP|DP$), or any other combination from flow record identifiers, and the value ν_i can be, for example, the number of SYN requests. The CMS query can check for example *if* a given $DIP|DP$ is under SYN flooding attack by verifying the value of $\hat{s}_k(\kappa_i)$. But it is unable to identify *which* $DIP|DP$ is under attack.

3.2. MNP-CUSUM

To uncover anomalies we use the multi-chart non parametric CUSUM algorithm (MNP-CUSUM, [51]). The CUSUM algorithm is a sequential change point detection method. Suppose that a flow

of data is monitored sequentially : $y_1, y_2, y_3, \ldots$. The goal of a sequential change point detection algorithm is to detect with a delay as small as possible a change in the distribution of the data y_i. Suppose that the pre-change and post-change distributions are known and that they are characterized by probability density functions (p.d.f.) $f_{\theta_1}(y_k)$ and $f_{\theta_2}(y_k)$. The alarm time in the parametric version of the CUSUM algorithm is defined as $t_a = \min\{k \geq 1 : g_k \geq h\}$ where h is a threshold and g_k is the test statistics which can be computed sequentially : $g_0 = 0, g_k = \max(0, g_{k-1} + \log \frac{f_{\theta_2}(y_k)}{f_{\theta_1}(y_k)})$. The rational behind the CUSUM algorithm is that before the change the quantity $\log \frac{f_{\theta_2}(y_k)}{f_{\theta_1}(y_k)}$ is on the average negative, whereas after the change it is on the average positive : as a consequence, the test statistics g_k remains around 0 before the change, and it increases linearly with a positive slope after the change, until it reaches the threshold h when the alarm is raised. This is illustrated on Figure 2. In this case a change in the mean of a Gaussian time series is detected. S_k is the cumulated log-likelihood ratio which is defined as $S_0 = 0, S_k = S_{k-1} + \log \frac{f_{\theta_2}(y_k)}{f_{\theta_1}(y_k)}$. $s_k = \log \frac{f_{\theta_2}(y_k)}{f_{\theta_1}(y_k)}$ is the log-likelihood ratio. One can observe from this figure that S_k has a negative slope before the change point and a positive slope after the change has occurred.

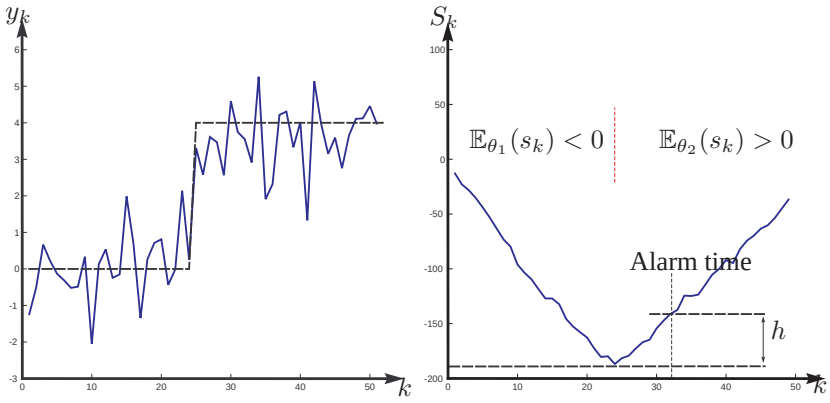


Figure 2. Intuitive derivation of the CUSUM.

The CUSUM algorithm is asymptotically optimal in the class K_γ of tests with average time between false alarms bounded by γ. More precisely, it reaches the Lorden bound which states that among the class K_γ the average detection delay is at best proportional to $\log(\gamma)$ as $\gamma \to \infty$ with a proportionality factor that depends on the Küllback-Leibler divergence between pre-change and post-change distributions.

Now suppose that we have to monitor several channels or sensors jointly : for example, $y_k(i)$ will be the number of SYN packets during a time interval k for a subset i of all DIP adresses (e.g. those who collide in the same cell number i of a given hash table). Until the unknown change time t_0 each random value follows a distribution $f_{\theta_1,i}(y_k(i))$ and after t_0 a change occurs in the distribution $f_{\theta,i}(y_k(i))$ of one channel only, say channel number i. The goal is to identify with a small delay that the change has occurred and also in which channel it has occurred. The multi-chart parametric CUSUM [51], simply called LR-CUSUM, is defined as : $t_a = \min_{1 \leq i \leq N} t_a(i)$ where $t_a(i) = \min\{k \geq 1 : g_k(i) \geq h_i\}$. h_i is a threshold adapted to the channel i (constant in our case) and $g_k(i)$ is a CUSUM test statistics for channel number i : $g_0(i) = 0, g_k(i) = \max(0, g_{k-1}(i) + \log \frac{f_{\theta_2,i}(y_k(i))}{f_{\theta_1,i}(y_k(i))})$. The alarm is raised when

one of the test statistics $g_k(i)$ reaches the threshold h_i and the change is declared in the channel i. The LR-CUSUM is asymptotically optimal in the sense that (i) the average detection delay for a change of type i is asymptotically linear with h (when $h \to \infty$) with a slope that is related to the entropy of the pre-change distribution, and (ii) if $h = \log(N\gamma)$ then the mean time before false alarms is greater than γ and the worst case of the average detection delay is asymptotically linear with $\log(\gamma)$ (when $\gamma \to \infty$).

Due to large variations in traffic patterns, to non stationarities, and to lack of consensus on network traffic characteristics, we cannot assume that the monitored variables follow a specific distribution. Therefore, we use the non-parametric version of the multi-chart CUSUM, as it only requires a very loose information on the distribution of the traffic time series before and after the change. In the non-parametric multi-chart CUSUM (MNP-CUSUM) the log-likelihood ratio $\log \frac{f_{\theta_2,i}(y_k(i))}{f_{\theta_1,i}(y_k(i))}$ is replaced with some function $L_i(y_k(i))$ which is chosen in such a way that its average value $\mathbb{E}(L_i(y_k(i)))$ is negative before the change and positive after the change. For example, in our simulations we chose $L_i(y_k(i)) = y_k(i) - (\mu_i + c\sigma_i)$, where μ_i and σ_i represent the pre-change mean and standard deviation respectively. With this definition the non parametric CUSUM is sensitive to a change in the mean value of the time series, supposing that μ_i is the average value before change, and supposing that the average value of the time series after the change is greater than $\mu_i + c\sigma_i$.

Network traffic is naturally variable. Without any anomaly (attacks, flashcrowds, etc.) the traffic is subject to natural variations due, for example, to day/night effects. These variations occur on a significantly longer time scale than anomalies which can then be identified as abrupt changes in traffic patterns. In this paper, we will use CUSUM algorithm to detect short term anomalies, and the detection of longer-term anomalies (several minutes or hours) is not addressed in this paper. Because of natural variability the parameters of traffic distribution in normal operation, for example μ_i and σ_i, slowly vary along the time. In order to follow the slow trends of traffic parameters on the long term the mean μ_i and variance σ_i^2 before change are estimated recursively using the EWMA (Exponentially Weighted Moving Average) : $\hat{\mu}_i(k) = \alpha\hat{\mu}_i(k-1) + (1-\alpha)y_k(i)$ and $\hat{\sigma}_i^2(k) = \alpha\hat{\sigma}_i^2(k-1) + (1-\alpha)(y_k(i) - \hat{\mu}_i(k))^2$. The estimation of mean and variance using the EWMA can deal with seasonal variations (working hours, night, day of week, etc.), by slowly updating these statistical parameters ($\hat{\mu}_i$,$\hat{\sigma}_i^2$). It is worth noting that the algorithm is not much sensitive to the value of the weighting parameter α that should be taken close to 1 ; a typical value is $\alpha = 0.9$.

4. PROPOSED APPROACH

To detect network anomalies over high speed networks, the first logical step is data reduction and aggregation of the huge amount of collected flow records. This step is useful for several reasons, including saving the bandwidth used for report transmission, and reducing the memory requirement and the processing complexity of analyzing collected data. We achieve this through flow level sampling, by selecting small size flows (size is the number of packets in the flow), as done in [7, 6].

Flows with a small number of packets are the source of many anomalies. In DDoS/DoS, the attacker uses a spoofed IP address with each packet to evade detection, identification, filtering and tracing back. Even when using botnets, the disclosure of zombies is not in the interest of the attacker. Firstly, the identification of the attacker may be more easily discerned, and secondly he wants to keep his remote control of the zombie for future use. Therefore, spoofed source IP addresses are used by bots in each packet when launching DDoS.

In the case of NetScan, worms usually scan networks to infect new vulnerable hosts as quickly as possible (e.g. Code Red, Slammer). Network scan also allows an attacker to identify an active host for security assessment, where all exploitations are always preceded by a reconnaissance phase. In security assessment, the malicious user scans port for enumerating services to identify vulnerable applications. Each packet in all these attacks results in a new flow.

To select all small flows with size $x < B$, we choose a sampling rate inversely proportional to the number of packets in the flow ($p = 1/x$) *if* ($x \geq B$) and no sampling *if* ($x < B$). B is a threshold for the number of packets. This is a slight modification of selective sampling ($p = B/n.x$) suggested in [7]. Despite the sampling, the reduction gain ratio was approximately 5% over the used traces. Manual inspection of these IP traces with P2P traffic, shows that the majority of TCP flows have less than 3 packets/flow. In fact, the dominant number of records are small flows. Therefore, to achieve a reduction ratio of more than $k\%$, we use the random sampling technique to reduce the number of records with small numbers of packets, as shown in eq. 1:

$$q = \begin{cases} rand[1,k] == 1\ ?\ 1\ :\ 0 & \textit{if } x \leq B \\ rand[1,x] == 1\ ?\ 1\ :\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

If $q = 1$ then the flow is kept otherwise it is discarded. This means that the sampling rate is ($p = 1/k$) *if* ($x \leq B$) and ($p = 1/x$) *if* ($x > B$). In our simulations we took $k = 2$ and $B = 3$. At monitoring points each flow generates a record. Once the flow has expired (either because of flow termination or because of timeout expiration) the number x of packets in the flow determines the value of the probability $p(x)$. The record is then exported to the central NOC with probability $p(x)$.

It is worth noting that per flow record collection is performed at monitoring points and that records processing for anomalies detection and malicious flows identification is performed at the central NOC. Monitoring points can be for example routers in the infrastructure of the ISP. In this paper, we assume the monitoring infrastructure (monitoring points, and central collector) has always enough computing resources to monitor and analyze the attack, while the consumption of communication resources are mitigated by sampling. The sampling procedure is beneficial since it reduces the number of exported flows which results in bandwidth usage reduction for flow exports and in CPU usage reduction at NOC. Under attack situations, load shedding mechanisms, such as proposed in [8], might be considered as extensions, but these are left out of scope of the current work.

When designing a traffic monitoring system (IDS, application recognition system, etc.) it is important to consider how easy it can be for an adversary to evade the detector. One could imagine that the attacker would try to evade the detector by generating artificially large flows. This strategy would not succeed in the context of SYN flooding attacks for some reasons that we are going to explain. Let us assume that the attacker sends a SYN packet and goes on transmitting data without acknowledging the SYN-ACK packet. Then the server will send a RST packet and the TCP connection will be closed without having generated a ”large flow”. In fact, the attacker cancels the SYN flooding attack by generating packets in the same flow without sending SYNACK. On the other hand if the attacker sends several SYN packets in the same TCP connection these packets will be considered as duplicate SYN packets by the server and this will not result in a DoS attack. It is important to note, that small flow sampling is applicable for TCP SYN flooding and SYN scanning attacks, and can not be applied to detect other attacks based on UDP or ICMP.

After data reduction, the anomaly detection phase uses random aggregation, to avoid aggregating the whole set of records into one, and to reduce the required memory in per-record treatment. The proposed approach is based on two data structures: Count-Min Sketch (CMS) and Multi-Layer reversible Sketch

(MLRS) as shown in Figure 3. The theoretical background on the CMS data structure and on the MNP-CUSUM algorithm that operates over this structure is summarized in section 3. The shared counters are continuously updated from the input data stream. κ_i is a key that identifies a flow (e.g. SIP—DIP) and ν_i is a reward associated to that flow ; in our case ν_i is a number of SYN packets for this particular flow during a fixed duration period T (e.g. $T = 1$ min.) The cells in the sketch are continuously updated that is to say that for each new SYN packet the d corresponding cells in the CMS sketch are incremented by 1. d instances of the multi-channel non parametric CUSUM (MNP-CUSUM) algorithm run in the background in order to monitor each row of the sketch. The aim is to detect a significant change in one of the cells of each row. At the end of each period T the MNP-CUSUM statistics $g_k(i,j)$ are updated as follows : $g_k(i,j) = max(0, g_{k-1}(i,j) + \Delta x_{i,j} - (\hat{\mu}_{ij} + c\hat{\sigma}_{ij}))$. $\Delta x_{i,j}$ is the difference between the value of the cell (i,j) in two consecutive intervals or, equivalently, the increment of cell (i,j) during that interval. $\hat{\mu}_{ij}$ and $\hat{\sigma}^2_{ij}$ are the sample mean and sample variance of the time series of cell increments $\Delta x_{i,j}(k), k = 1, 2, \ldots$; they are computed as an exponential moving average as it was explained in section 3. In our implementation, each cell in the 2D table becomes a data structure, containing the current and previous value of the number of SYN, the sample mean $\hat{\mu}_{ij}$ and variance $\hat{\sigma}^2_{ij}$ and the value of MNP-CUSUM statistics $g_k(i,j)$. Once the statistics $g_k(i,j)$ exceeds the threshold h for one of the cells, say cell j, then the MNP-CUSUM that monitors row i raises an alarm. When we have at least one alarm in all rows of the sketch, then a global alarm is raised.

As we want to uncover culprit flows, e.g. the flows responsible for the CUSUM raised alarms, one solution for verification could be to keep the key values inside a file/database, and to re-hash these data to find the key that maps to cells with raised alarms. This procedure is heavy in terms of storage space and update speed as it requires storing all keys for verification.

In fact, due to random aggregation and collision occurrences with hash functions, reversing sketch is a difficult operation. However, some interesting works have been proposed in [47, 13, 24] to reverse hashing, in order to identify keys associated with cells having raised alarms. The first approach [47] is based on modular hashing and mangling via Galois Field $GF(2^n)$ operators, which is complex and more efficient for hardware implementation, as it was done with FPGA equipment in [47]. The second approach [13] is an extension of the previous method. The third approach [24] is based on nonsingular matrix on $GF(\{0,1\}^n, \oplus, .)$, and requires more memory and update cost than the previous method.

Our approach to reverse sketch is based on the idea of exploiting cell index to store keys. An additional 2D table, so-called Multi-Layer Reversible Sketch (MLRS), also containing shared counters is used (as shown by the first table in Figure 3). The key is implicitly stored in this 2D table. Firstly, the key in binary is divided into l equal parts, where each part is used as index of the shared counter in each line of the MLRS. The width of each line in the MLRS table is given by $w = 2^P$, and the number of lines is $l = \lceil N/P \rceil$. N is the number of bits used to represent the largest number in the universe of key, and P is the number of bits in each part of the key. The update procedure of the CMS and MLRS is summarized by Algorithm 1.

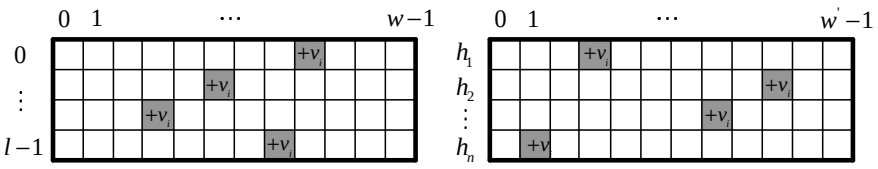


Figure 3. Multi-Layer Reversible Sketch *MLRS* and Count-Min Sketch *CMS*.

Algorithm 1 Sketches Update procedure

1: $Ckey = Encrypt\_RC4(key)$;
2: **for** $i = 0$ to $d - 1$ **do**
3: $\quad j = univ\_hash_i(Ckey)$;
4: $\quad CMS[i][j].counter+ = \nu$;
5: **end for**
6: **for** $i = 0$ to $l - 1$ **do**
7: $\quad MLRS[i][Ckey \ \& \ (2^P - 1)].counter+ = \nu$;
8: $\quad Ckey >>= P$;
9: **end for**

A MNP-CUSUM runs in the background for each row of the MLRS structure, as it is the case for the CMS structure. At the end of each interval T, we release a hierarchical search in each layer of the MLRS for cells with alarm raised by the MNP-CUSUM. We must have at least one cell in each layer with a CUSUM raised alarm. Otherwise, no needs to continue searching in other layers or to look in the result of the CMS.

In the simple case, when we have one alarm in each layer, the key can be recovered by concatenating the index of the l cells in MLRS. We cannot be sure of the suspect key before verification, since because of collisions with other key prefixes, the corresponding counter value can become large without any attack going on. The suspect key is verified through hashing and verifying if an alarm was raised for the corresponding cells of the CMS.

In the case of $key = DIP$, even with different widths for the decomposition procedure (8bit, 10bit, 12bit, etc.), many cells in different layers will be subject to collision occurrence (same prefix, etc.), and in some cases, we will end up with a larger set of keys to verify through *CMS* than the original key list since all possible combinations must be considered. Nevertheless, it is important to note that even if the set of suspect keys is larger than the departure one, it requires smaller memory and has a fast update time with respect to maintaining a database/file of the original key list.

To resolve this problem and reduce collision in MLRS, we use encryption with stream cipher RC4 (Ron's Code [25]) algorithm rather than Galois Field $GF(2^n)$. In cryptography RC4 is the most popular stream cipher. It is used in popular protocols such as SSL or WEP. RC4 is remarkable for its simplicity and speed in software. The used C implementation of the RC4 code is available from [3]. The RC4 algorithm is ideal for software implementation, as it requires only byte manipulations and its implementation is based on a few lines of code. It has been proven to be powerful in our experimentations for mangling and destroying any correlation between keys having some portion in common (prefixes). The Hamming distance between cipehered keys is large even if there is some correlation between the original ones.

Encryption is a bijective function, which transforms clear text key into cipher text, denoted by $Ckey$, where $Ckey = E_S(key)$ and $key = D_S(Ckey)$. S is the shared key, and the function $E()$ must be chosen in a way to destroy any correlation between clear texts with for example the same prefix. This principle is shown in Table I for the key built from the concatenation of DIP with DP, with the use of RC4.

Any bijective function able to destroy correlation between keys, and return a completely random set of keys, can be used. Afterward, $Ckey$ is divided into l parts and used as an index in MLRS. To recover the key from cells with raised alarms in MLRS, we must concatenate the raised alarm indexes in each layer to get $Ckey$, and use $D_S(Ckey)$ to recover the suspect key κ from MLRS. This key is

DIP:DP	Encrypted
192.168.100.101:80	6E96A9468CF5
192.168.100.102:80	DD08C66271E4
192.168.100.103:80	2F0F5EB19313
192.168.100.104:80	3FE7204B0435
...	...

Table I. Encryption of *DIP—DP* by RC4.

used for verification through CMS to ensure that all associated cells in the d hash tables have raised alarms. The hierarchical search procedure, as well as verification and sketch inversion are summarized in Algorithm 2, for a universe of size 2^N, and a width of 2^P for MLRS, $P = N/2$ and $l = 2$. A boolean alarm variable is used to indicate if the MNP-CUSUM algorithm has raised an alarm for the considered cell.

Algorithm 2 Hierarchical search and verification

for $i = 0$ to $2^P - 1$ **do**
  if $(MLRS[0][i].Alarm)$ **then**
    for $j = 0$ to $2^P - 1$ **do**
      if $(MLRS[1][j].Alarm)$ **then**
        $Ckey = (j << P) \mid i$;
        $Alarm = cms\_alarm\_query(CMS, Ckey)$;
        if $(Alarm)$ **then**
          $key = decrypt\_RC4(Ckey)$;
          $output(key)$
        end if
      end if
    end for
  end if
end for

The proposed framework can be applied to detect different types of attacks, e.g. TCP SYN flooding, UDP packet storms, TCP/UDP PortScan, NetScan, Smurf, etc. Nevertheless, in this paper, we will only focus on TCP traffic and especially on the number of connection requests (SYN).The proposed method does not only perform anomaly detection, that is to say to raise an alarm when an anomaly is detected. It is also able to able to identify malicious flows through the use of associated key values, and to classify the anomaly by using different key definitions.

To classify anomaly, we extract from each flow record three keys ($key_1 = DIP|DP$, $key_2 = SIP|DIP$, $key_3 = SIP|DP$) through the concatenation of the binary value of two fields from each entry. These keys are used to update three instances of the proposed approach with the observed number of SYN packets with the corresponding key value for each discrete time interval (say every minute). We denote by F_i the framework instance associated with key_i. The classification algorithm can be described as follows :

Step 1. We seek to detect victims of DoS/DDoS SYN flooding. We update the counters of F_1 with the key_1 during predefined T time intervals, and we output the list L_1 of all victim servers $DIP|DP$.

Step 2. The key_2 is used to update a second instance F_2. Outputs of this step are malicious SIP, which try to scan the ports of a given DIP, if the latter is not a victim of DDoS/DoS. In contrast, if DIP is in list L_1 (i.e. victim of flooding), we store a list of suspect (LoS) whose elements are (SIP, DIP, DP), because SIP are suspects of contribution in DDoS/DoS through a static source address.

Step 3. The key_3 is used to update a third instance F_3, where output keys are SIP trying to perform a NetScan activity, if the SIP does not belong to the list LoS. Otherwise, it is the source of DDoS/DoS flooding.

The preceding three steps are used in our implementation to early identify three types of anomalies (DDoS/DoS victim, NetScan and PortScan), and provide useful information about victim or attacker. The identification of scanning attacks is based on source addresses generating too many connection requests at different ports/hosts. The PortScan and NetScan were chosen for their association with malicious attacks and worms. PortScan is often used by attackers for vulnerability assessment of running applications at victim host. NetScan are usually performed by worms in their spreading phase (random scan in code Red, linear in Blaster, bias in code Red II and Nimda, etc.) to gain access to new machines and infect them. Our proposed approach is able to detect all these kinds of scan activities.

5. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of using MNP-CUSUM over a Multi-Layer Reversible Sketch (MLRS) and a Count Min Sketch (CMS) structures in the detection and classification of attacks (SYN flooding attack, NetScan and PortScan). We have implemented MNP-CUSUM over sketch in C by extending the code of CMS available from [1]. We have tested the proposed algorithm over MAWI traces available from [2], and other IP traces used in the OSCAR project funded by the french National Research Agency. The so-called OTIP and ADSL traces were collected and provided to the OSCAR consortium by France Telecom (FT). The ADSL trace was collected on a geographical and technical subdivision of an ADSL network. The OTIP trace is a 6.9 GB trace made of Netflow records from CISCO routers in a FT backbone network. These traces were used as a benchmark and have been widely analyzed in the project. We also use one of the traces collected during online experiments, which were carried out to test the detection performance of the algorithms designed during the project. The topology of the measurement overlay deployed during the OSCAR project is displayed on Figure 4.

Each partner is equipped with a DAG card which sniffes all the traffic between a laboratory network and a central router in the partner institution. Flow level reports are collected at the sniffing point by each partner and sent to a central Network Operation Center (NOC) by UDP sockets. Different algorithms run at the NOC in order to detect anomalies in the aggregated reports. Figure 4 describes a centralized detection architecture. Semi-decentralized architectures with anomaly detection algorithms running at the different monitoring points have also been implemented in OSCAR.

In order to reduce the spatial and temporal complexity of the proposed algorithms, OSCAR partners decided to enhance the capture process of high speed sniffing card (Endace DAG card), by adding a small C program to transform captured packets over one minute into OSCARFIX flows.

Therefore, we keep approximately the same definition of flow as the one used by Netflow [18] in Cisco routers or standardized by the IETF in the IPFIX protocol [17]. An OSCARFIX flow is defined as a unidirectional stream of packets that share the same five tuples (source IP, destination IP, protocol, source port, destination port). When a flow is considered as finished (through flow aging or TCP connection termination) a flow record is exported. The OSCARFIX flow record contains a variety of information such as the source and destination IP, source and destination ports. Instead of including

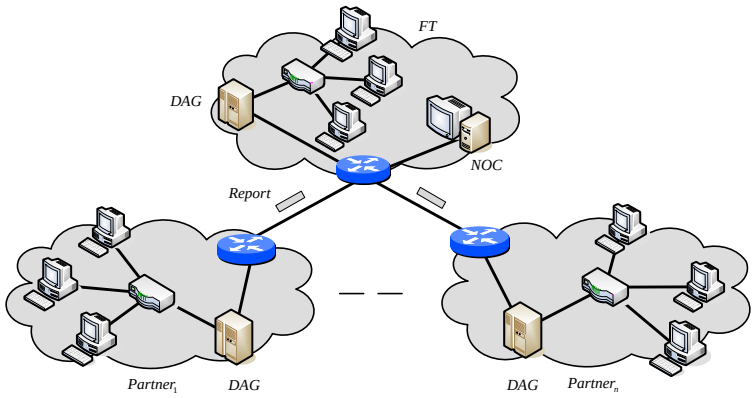


Figure 4. Topology of the measurement overlay

the binary XOR of all TCP flags of the flow in the record as it is the case in Netflow, OSCARFIX flow records include the number of SYN, SYNACK, RST & FIN packets in the flow. In OSCARFIX we took a timeout value of 1 minute. A new flow record is generated with the first new packets crossing the DAG sniffing card. To simplify implementation, even if the flow doesn't finish in the current time interval (1 min), subsequent packets are considered as belonging to a new flow in the following minute.

Although the time measurement interval has an impact on the detection precision, an agreement at 1 minute of data collection has been adopted by all the project partners. In fact, a small interval enhances the detection delay at the cost of potentially increasing the False Alarm Rate (FAR). On the other hand, a large interval increases the detection delay. One minute was considered as a good tradeoff by the OSCAR consortium. Online implementation over Endace DAG card has been realized, and many experiments have been conducted online for accuracy analysis. Our results are encouraging in terms of accuracy and response time. All the experiments have been performed using a Ubuntu box with an Intel core 2 DUO (E4500) with $2.2\ Ghz$, $3\ GB$ of RAM and $750\ GB$ SATA disks.

In this paper, we will present the analysis results obtained over 3 traces. The first trace is made up of two hours of OSCARFIX flow records collected during online experiments in the framework of the OSCAR project. The second set of measurements is made up of an ADSL download trace ; it contains unidirectional packet traces collected during 3 hours on a geographical and technical subdivision of an ADSL access network. The OTIP trace contains 3 days of bidirectional traces collected with NetFlow on some routers of a backbone network. The main objective of parsing the last traces (ADSL & OTIP) is to check the scalability of our algorithms that is to say their ability to analyze traffic at a high data rate in real time. Each report (1 minute) must be analyzed within the next minute of data collection. The last experiment has been conducted with the aim of analyzing the performance of the algorithms ; we conducted an off-line experiment on a synthetic trace to study the influence of the various parameters on the detection and false alarm rates.

The parameters we considered for the MNP-CUSUM algorithm were the following ones: threshold value $h = 7$ in the MNP-CUSUM algorithms, weighting factor $\alpha = 0.9$ in the EWMA algorithms (estimation of the sample mean and variance), standard deviation scaling factor $c = 0.5$ in the update procedure of the MNP-CUSUM algorithms. Sketches parameters for MLRS were $P = 12$ ($l = 4$) for keys with 48bits, and $P = 14$ ($l = 5$) for keys with 64bits. CMS parameters were: $w^{'} = 4096$ and the number of hash functions $d = 4$ chosen from the set of 2-universal hash functions, and with the use of

tabulation [52]. The used configuration parameters of the sampling algorithm were: $B = 3$ and $k = 2$.

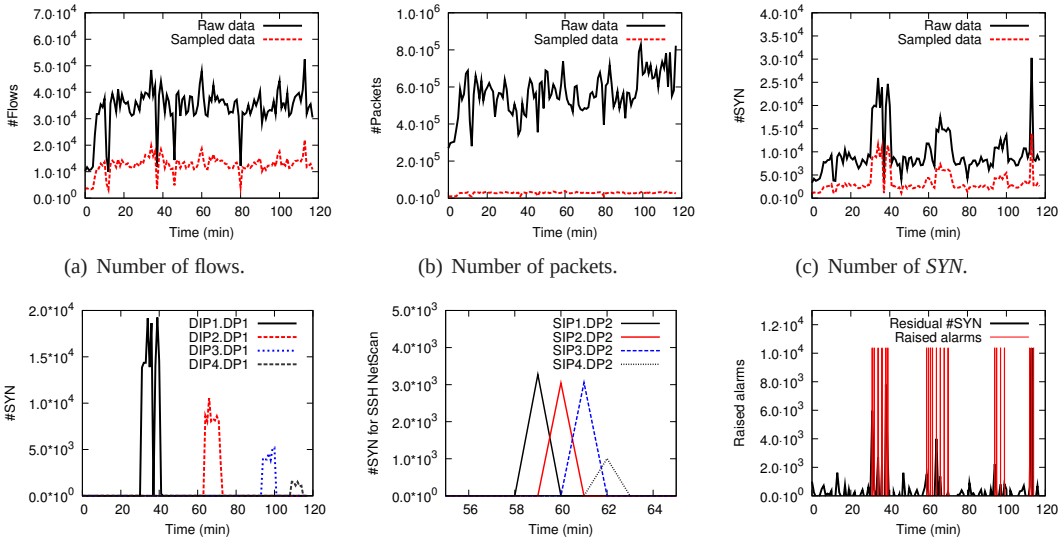
Experiment 1. In the first experiment, we validate and tune the model using an online experiment trace. This online experiment has been conducted using Planetlab machines, and with the collaboration of some French research laboratories (project partners). The data trace corresponds to traffic collected at NOC in Figure 4, and each minute is the concatenation of OSCARFIX reports collected, using DAG cards with GPS-synchronized timestamp, over the experimental network of each partner. These reports are exported to NOC over UDP sockets. Reports contain much background traffic (HTTP, SSH, etc.) with mainly P2P traffic. During this experiment, well known attacks type/instant have been generated by FT to test the detection efficiency of the proposed algorithms. Figures 5(a), 5(b) & 5(c) display the variation of the total number of: flows, packets, and SYN before and after sampling. We can obviously conclude from Figure 5(c) to the efficiency of the sampling algorithm in reducing legitimate variations and the amount of traffic to process. After the application of our proposed approach over this trace, we identify 4 victims of SYN flooding attacks, and 4 hosts scanning the network for an SSH server. The number of malicious SYN received by the four different victim servers are shown in Figure 5(d), where we can observe 4 attacks of different intensity. The number of SYN received by victims (10.0.0.1-4:18019) are obtained by filtering the trace to extract flow records with the identified $DIP.DP$ using our approach. We demonstrate the ability to detect low intensity attacks. In this trace, we also identify NetScan against SSH server. The number of SYN sent by scanners are displayed in Figure 5(e), where even for a small intensity, the attack has been detected and identified. The total number of SYN as well as raised alarms are shown in Figure 5(f). It is important to note that we had 0 false positives and 0 false negatives reflecting the efficiency of our proposed approach. The response time for the analysis of the 2 hour trace is less than one minute.

As we are collecting unsampled flow-records, we evaluate the impact of the used sampling technique on the precision of the anomaly detection algorithm. To compare the results, we count the number of raised alarms before and after sampling. We have one additional raised alarm after sampling during the last flooding attack. This is due to the ability of the sampling technique used to magnify suspect flows.

Furthermore, to obtain a comparison, we apply single channel non parametric CUSUM over the raw sequence which results from the aggregation of all flows into one time series. The result is very interesting and deserves to be noted. First, all NetScan attacks were not detected due to aggregation, because low intensity traffic fluctuations are not observable. Second, we had 3 false alarms even when tuning the parameters h and c. Either the last 2 flooding attacks were missed after tuning, or we had a larger number of false alarms. On the other hand, only one false alarm was obtained by applying single channel CUSUM over the sampled traffic, where many legitimate deviations have been smoothed or discarded by the proposed sampling technique.

Experiment 2. Since the performance observed over the previous 2 hours trace can not be generalized without further analysis, we consider 3 hours of unidirectional packet level capture (pcap) with anonymized IP addresses over an ADSL infrastructure, and we transform this trace into *OSCARFIX* flow records. We carry out the same analysis and manual verification as in the first experiment. In fact, we conduct the same analysis over upload and download ADSL traces, but due to space limitations, we omit to comment on the results over the download trace since they are very similar to those over the upload trace.

Figures 6(a), 6(b) & 6(c) present the total number of flows, packets, and SYN before and after sampling. One SYN flooding attack and one PortScan have been identified by our approach. Figure 6(d) shows the number of SYN received by the identified victim server of distributed SYN flooding ($DIP1.DP1 = 97.65.192.238 : 35415$), and figure 6(e) displays the number of SYN generated for PortScan by $SIP2 = 240.178.148.21$ to scan the ports of $DIP2 = 97.68.23.88$. After filtering, we



(a) Number of flows. (b) Number of packets. (c) Number of *SYN*.

(d) Malicious *SYN* for DDoS. (e) Malicious *SYN* for NetScan. (f) Number of SYN & raised alarms.

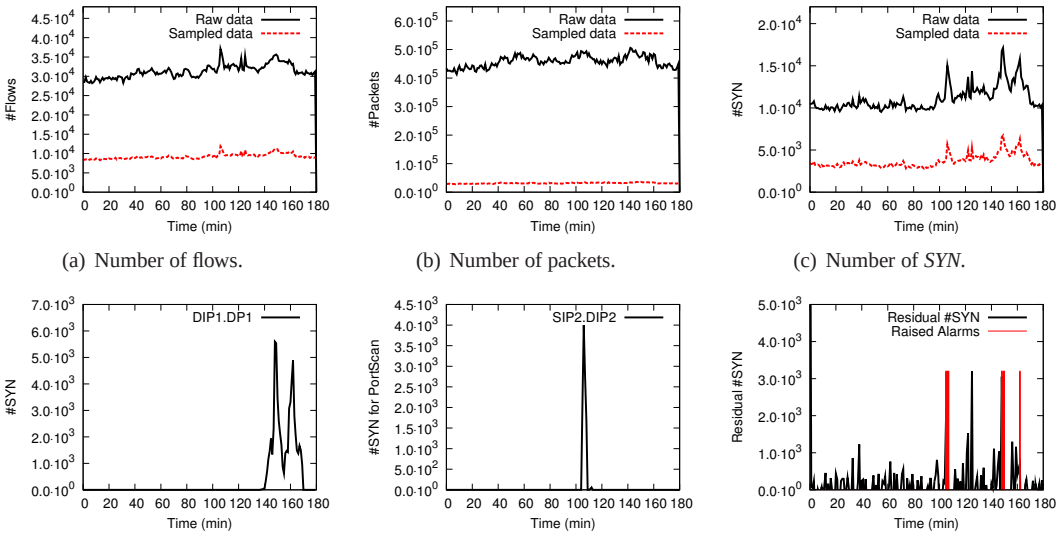
Figure 5. Analysis results for the online experiment.

get only one additional alarm for SYN flooding at its end. As the only available information about this trace is P2P contents with some attacks, this trace may contain other undetected anomalies and we can not conclude to 0 false negatives, but we can conclude to 0 false positive. The identified SYN flooding additional alarm has been manually verified during our analysis, and also detected by other partners. The PortScan is verified manually by flow-records extraction and verification. Figure 6(f) shows the time series of the total number of SYN for sampled flows, and the raised alarms. The response time for the analysis of the 3 hours ADSL trace is less than 5 minutes.

Experiment 3. This trace is exported from CISCO routers (with Netflow v5) to a central collector (NOC), whose role is to store the received records in a database/file after anonymizing IP addresses. It contains 3 days of flow records ($\sim 896.10^5$ flows) and has a size of $\sim 6.9GB$.

Figures 7(a), 7(b) and 7(c) show the variation of the total number of flows, packets and SYN before and after sampling during the 3 days. After the application of our proposed framework over traces to uncover attacks, we compare our results with other partners' results. We conclude to one misdetection after comparison and manual verification. Afterward, we isolate the number of connection requests received by each identified victim at the specific port as shown in figure 7(d) for manual verification. The number of SYN received by victim (attack missed by our approach) is presented in figure 7(f). The manual verification of missed attack shows a TCP flooding at different ports, and this explains its misdetection. Furthermore, the detection of flooding at different ports can be achieved by monitoring the DIP instead of $DIP.DP$. Also, we notice the presence of one NetScan by $SIP = 224.87.77.70$ with $DP = 65506$ (figure 7(e)). Manual verification of *OTIP* trace proves the NetScan by the given SIP at raised alarm instants. It is worth noting that response time for analyzing the whole 3 days *OTIP* trace is about a few minutes for the 3 days trace over a Pentium $2.2\ Ghz$ with $3\ GB$ of RAM memory.

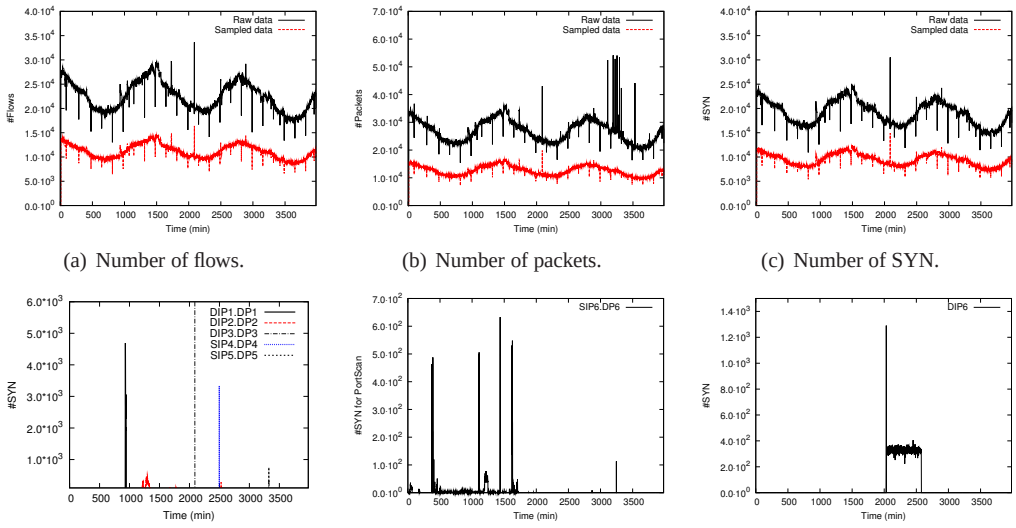
The previous plots over real IP traces demonstrate that the number of anomalous source/destination adress pairs (6 abnormal behaviors) is so small with respect to the whole number of collected records



(a) Number of flows. (b) Number of packets. (c) Number of *SYN*.

(d) Malicious *SYN* for DDoS. (e) Malicious *SYN* for PortScan. (f) Number of SYN & raised alarms.

Figure 6. Analysis results for ADSL download trace.

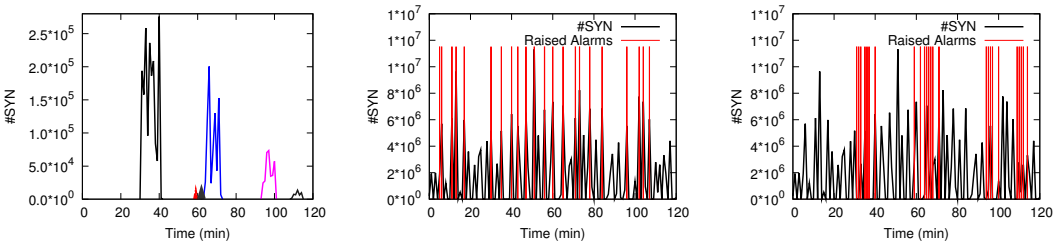


(a) Number of flows. (b) Number of packets. (c) Number of SYN.

(d) Malicious *SYN*. (e) Malicious *SYN* for NetScan. (f) Missed attack.

Figure 7. Analysis results for the OTIP trace.

in the trace, that providing only an attack instant without further information is not enough to uncover culprit flows. Our proposed approach identifies the bad flows and helps understanding the malicious activities behind these flows.



(a) DDoS & NetScan attacks. (b) CUSUM raised alarms. (c) MNP-CUSUM over sketch raised alarms.

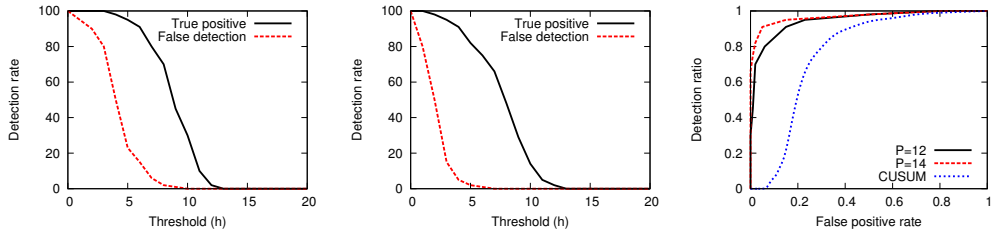
Figure 8. Comparison between CUSUM over raw data & CUSUM over sketch.

Experiment 4. This experiment is conducted in order to study the accuracy of the proposed framework and to test its sensitivity with respect to the parameters of the detection algorithm. We begin our evaluation by a comparison between CUSUM over raw data and CUSUM over sketch. By raw data we mean that the input of the CUSUM algorithm is the time series of aggregated number of SYN. We use an IP trace of 2 hours, with 6 known anomalies instant & type (4 DDoS and 2 NetScan attacks) as shown in figure 8(a). The 2 scan attacks are represented by filled curves in figure 8(a). The alarms raised by single channel NP-CUSUM over raw data are shown in figure 8(b), where we can observe false alarms due to variations in the aggregated number of SYN, and misdetection of low intensity attacks, which evade the detector after aggregation of the whole traffic into one time series. The 2 NetScan attacks are not detected when aggregating the whole traffic in one time-series, because the change is smoothed by the aggregation of the number of flows. Furthermore, to detect DDoS with small intensity attacks, a low threshold value for CUSUM is required. However, low threshold value in CUSUM algorithm incurs high false alarm rate with the detection of only 4 over the 6 existing attacks. Figure 8(c) shows the alarms raised by the MNP-CUSUM over sketch. A finer grained analysis results from using many channels in order to monitor the traffic. This clearly reduces the number of false alarms, and increases the hit ratio of low intensity attacks.

The efficiency of an anomaly detection algorithm is usually described by two values: the power of the test (or detection rate) and the false alarm rate (FAR). In our context false alarms are legitimate flows that are classified as malicious, whereas the power of the test is the proportion of malicious flows that are effectively detected as malicious. There is an inherent tradeoff between power and FAR. Indeed, in an attempt to detect malicious flows with a higher probability one could be tempted to decrease the detection threshold but this would automatically result in a higher FAR. The ROC (Receiver Operating Characteristics) curve is usually used to depict such a tradeoff between power and FAR.

In our analysis we conduct off-line and many times the same experiments with different values of detection threshold h in order to test the impact of the threshold h over the power of the test and over the FAR. We also test the impact of the *MLRS* sketch width P, considering different values of P. Because of the lack of public and well documented traces with well known attacks, we use the *OTIP* trace (the largest trace with $6.9GB$ of data) as background traffic in which we manually delete the previous anomalies, and instead insert 100 SYN flooding attacks of different intensities in different times. Firstly, we apply CUSUM algorithm over this trace by aggregating whole flows in one time series. Afterward, we apply our implementation over the same trace while changing the value of the parameters h and P.

Power and FAR values are easily established because we know in advance the IP address of the



(a) $P = 12$. (b) $P = 14$. (c) $P_{TP} = f(P_{FA})$.

Figure 9. $P_{TP} = f(h)$, $P_{FA} = f(h)$, ROC curve for CUSUM and MNP-CUSUM over sketch ($P = 12$,$P = 14$).

victim servers, the number of existing attacks, and their instants. The power or true positive ratio P_{TP} is the number of detected attacks divided by the total number of existing ones (100). The false alarm rate P_{FA} is the percentage of raised alarms that did not correspond to real attacks. Figure 9 shows the variations of the power and the FAR as functions of the detection threshold, $P_{TP} = f(h)$ (figure 9(a)), and $P_{FA} = f(h)$ (figure 9(b)) as well as the ROC curve (figure 9(c)) for two values of the *MLRS* width ($P = 12$ and $P = 14$). As it was expected the FAR as well as the power decrease as the threshold increases. Hence, a tradeoff between false alarm rate and detection rate must be found to control the sensitivity of the test and prevent false alarms. From this study it seems that a good choice of operating point is to select $h \simeq 7$ for $P = 12$ (and $h \simeq 5$ for $P = 14$) since for these values the detection rate is high while keeping a low FAR. We also notice that a large sketch width value upgrades the performance as displayed by ROC curves on figure 9(c). As a matter of comparison Figure 9(c) displays the ROC curve summarizing the performance of a single channel CUSUM algorithm over the raw traffic data for the same dataset. We notice that low intensity attacks are not detected after aggregation of the whole traffic into one time series. False alarms continue to raise even with large threshold value whereas the algorithm is still unable to detect any existing attacks. Thus is due to the high variability in the aggregated traffic pattern.

6. CONCLUSION

In this paper, we propose a new framework that integrates multi-stage sketch and multi-chart CUSUM for anomaly detection over high speed links. The proposed framework is able to automatically pinpoint the IP flows responsible for anomaly, through exploiting the matrix index in an additional multi-layer reversible sketch. The proposed approach consists of three stages: data reduction, anomaly detection and classification. The contributions are: data reduction when collecting flow records for bandwidth saving and analysis complexity reduction, software efficient sketch inversion method, making up overall an efficient algorithm to uncover hidden anomalies in the overall traffic.

It is obvious that worm signatures are unknown in their outbreak phase, and as some polymorphic worms (change their signatures to evade detection) use encryption with different keys and different encryption algorithms for every instance, it becomes a challenge for a signature based IDS to detect them. However, worms spreading phase tend to have a large number of destinations (NetScan) to infect all vulnerable systems, and thus can be identified by our proposed approach.

We proved that our approach is effective through implementation and testing on real traces with DDoS & Scan attacks. The sampling technique used discards many deviations generated by legitimate

flows, and thus reduces the false alarm rate. False positive reduction is the most important factor in measuring the performance of any detection system. The use of sketch and MNP-CUSUM increases the accuracy and reduces the detection delay with respect to the aggregation of all the traffic in one time series. We have shown the ability to detect hidden anomaly in overall traffic, and to reduce the false positives. Furthermore, online experiments have proven the effectiveness of the proposed approach as well as the early detection of attacks.

The proposed method is easily decentralized due to the linear property of sketch values. Ongoing work will concern the decentralized version of the proposed approach, and the reduction of the size of exchanged sketch information between different monitoring nodes in different layers. Decentralized anomaly detection is indeed a very important issue since many attacks are deployed over the Internet and it is necessary for the different monitoring points to manage to exchange information. These monitoring points can be located for example in peering points between Autonomous Systems (AS) or in different AS. The exchange of information between AS raises new challenges concerning for example end-user privacy related questions, or trustiness between the different organizations.

Acknowledgment

This work has been partially funded by the French National Research Agency through the OSCAR project. The authors would like to thank the OSCAR consortium for their participation to some of the experiments related in this paper. They would also like to thank the anonymous reviewers for some useful feedback as well as the help of Fabio Ricciato who shepherded this article.

REFERENCES

1. Count-Min sketch source code. http://www.cs.rutgers.edu/∼muthu/massdal-code-index.html.
2. MAWI working group traffic archive. http://mawi.wide.ad.jp/mawi/.
3. Optimized RC4 code. http://www.zengl.net/freeswan/.
4. E. Ahmed, A. Clark, and G. Mohay. A Novel Sliding Window Based Change Detection Algorithm for Asymmetric Traffic. In *Proceedings of the IFIP International Conference on Network and Parallel Computing (NPC'08)*, pages 168–175, 2008.
5. E. Ahmed, A. Clark, and G. Mohay. Effective Change Detection in Large Repositories of Unsolicited Traffic. In *Proceedings of the 4th International Conference on Internet Monitoring and Protection (ICIMP'09)*, pages 1–6, 2009.
6. G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou. Network Anomaly Detection and Classification via Opportunistic Sampling. *IEEE Network*, 23(1):6–12, 2009.
7. G. Androulidakis and S. Papavassiliou. Improving Network Anomaly Detection via Selective Flow-Based Sampling. *IET Communications Journal*, 2(3):399–409, 2008.
8. P. Barlet-Ros, G. Iannaccone, J. Sanjuàs-Cuxart, D. Amores-López, and J. Solé-Pareta. Load Shedding in Network Monitoring Applications. In *Proceedings of the USENIX Annual Technical Conference (ATC'07)*, pages 1–14, 2007.
9. C. Bo, B.-X. Fang, and X.-C. Yun. A new approach for early detection of Internet worms based on connection degree. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, pages 2424–2430, Guangzhou, China, 2005.
10. D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina. Impact of packet sampling on anomaly detection metrics. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (IMC'06)*, pages 159–164, 2006.
11. J. D. Brutlag. Aberrant Behavior Detection in Time Series for Network Monitoring. In *Proceedings of the 14th USENIX conference on System administration (LISA '00)*, pages 139–146, 2000.
12. S. Bu, R. Wang, and H. Zhou. Anomaly Network Traffic Detection Based on Auto-Adapted Parameters Method. In *Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM'08)*, pages 601–607, 2008.
13. T. Bu, J. Cao, A. Chen, and P. Lee. A Fast and Compact Method for Unveiling Significant Patterns in High Speed Networks. In *26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, pages 1893–1901, May 2007.
14. T. Bu, A. Chen, S. VanDerWiel, and T. Woo. Design and Evaluation of a Fast and Robust Worm Detection Algorithm. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM'06)*, pages 1–12, 2006.

15. C. Callegari, S. Giordano, M. Pagano, and T. Pepe. On the Use of Sketches and Wavelet Analysis for Network Anomaly Detection. In *Proceedings of the 1st International Workshop on Traffic Analysis and Classification (TRAC)*, 2010.
16. M. Charikar, K. Chen, and M. Farach-Colton. Finding Frequent Items in Data Streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP '02)*, pages 693–703, 2002.
17. B. Claise, S. Bryant, G. Sadasivan, S. Leinen, and T. Dietz. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101, Jan. 2008.
18. B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes. Cisco Systems NetFlow Services Export Version 9. RFC 3954, Oct. 2004.
19. G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Diamond in the Rough: Finding Hierarchical Heavy Hitters in Multi-Dimensional Data. In *Proceedings of the 23rd ACM SIGMOD*, pages 155–166, 2004.
20. G. Cormode and S. Muthukrishnan. What's New: Finding Significant Differences in Network Data Streams. In *Proceedings of IEEE Infocom*, pages 1534–1545, 2004.
21. G. Cormode and S. Muthukrishnan. An Improved Data Stream Summary: the Count-Min Sketch and its Applications. *Journal of Algorithms*, 55(1):58–75, April 2005.
22. G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho. Extracting Hidden Anomalies using Sketch and Non Gaussian Multiresolution Statistical Detection Procedures. In *Proceedings of the ACM SIGCOMM Workshop on Large-Scale Attack Defense (LSAD'07)*, 2007.
23. Z. Fadlullah, T. Taleb, N. Ansari, K. Hashimoto, Y. Miyake, Y. Nemoto, and N. Kato. Combating Against Attacks on Encrypted Protocols. In *Proceedings of the IEEE International Conference on Communications, 2007 (ICC'07)*, pages 1211–1216, 2007.
24. W. Feng, Z. Zhang, Z. Jia, and Z. Fu. Reversible Sketch Based on the XOR-Based Hashing. In *Proceedings of the Asia-Pacific Conference on Services Computing (APSCC '06)*, pages 93–98, 2006.
25. S. Fluhrer and D. McGrew. Statistical Analysis of the Alleged RC4 Keystream Generator. In *Proceedings of the 7th International Workshop on Fast Software Encryption (FSE '00)*, pages 19–30, 2001.
26. L. He, B. Tang, and S. Yu. Available bandwidth estimation and its application in detection of DDoS attacks. In *Proceedings of the 11th IEEE Singapore International Conference on Communication Systems (ICCS'08)*, pages 1187–1191, 2008.
27. N. Hohn and D. Veitch. Inverting Sampled Traffic. *IEEE/ACM Transactions on Networking*, 14(1):68–80, 2006.
28. L. Huang, X. Nguyen, M. Garofalakis, and J. M. Hellerstein. Communication-Efficient Online Detection of Network-Wide Anomalies. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 134–142, 2007.
29. J. Kang and J.-Y. Zhang. Application Entropy Theory to Detect New Peer-to-Peer Botnet with Multi-chart CUSUM. In *Proceedings of the 2nd International Symposium on Electronic Commerce and Security (ISECS'09)*, pages 470–474, 2009.
30. R. Kawahara, T. Mori, N. Kamiyama, S. Harada, and S. Asano. A Study on Detecting Network Anomalies Using Sampled Flow Statistics. In *Proceedings of the International Symposium on Applications and the Internet Workshops*, 2007.
31. H. Kim, B. Rozovskii, and A. Tartakovsky. A Nonparametric Multichart CUSUM Test for Rapid Intrusion Detection. *International Journal of Computing and Information Science*, 2(3):149–158, 2004.
32. B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based Change Detection: Methods, Evaluation, and Applications. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC'03)*, pages 234–247, 2003.
33. A. Lakhina, M. Crovella, and C. Diot. Diagnosing Network-Wide Traffic Anomalies. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '04)*, pages 219–230, 2004.
34. A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. *SIGCOMM Comput. Commun. Rev.*, 35(4):217–228, 2005.
35. P. Lee, T. Bu, and T. Woo. On the Detection of Signaling DoS Attacks on 3G Wireless Networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'07)*, pages 1289–1297, 2007.
36. C. Levy-Leduc and F. Roueff. Detection and localization of change points in high-dimensional network traffic data. *Annals of Applied Statistics*, 3(2):637–662, 2009.
37. X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina. Detection and Identification of Network Anomalies Using Sketch Subspaces. In *Proceedings of the 6th ACM SIGCOMM on Internet measurement (IMC '06)*, pages 147–152, 2006.
38. B. Lim and M. Uddin. Statistical-Based SYN-Flooding Detection Using Programmable Network Processor. In *Proceedings of the 3rd International Conference on Information Technology and Applications (ICITA'05)*, pages 465–470, 2005.
39. W. Lu and A. Ghorbani. Network Anomaly Detection Based on Wavelet Analysis. *EURASIP Journal on Advances in Signal Processing*, pages 1–16, 2009.
40. J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang. Is Sampled Data Sufficient for Anomaly Detection? In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (IMC'06)*, pages 165–176, 2006.
41. D. Moore, G. M. Voelker, and S. Savage. Inferring internet denial-of-service activity. In *Proceedings of the 10th conference on USENIX Security Symposium (SSYM'01)*, pages 9–22, 2001.
42. G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang. An Empirical Evaluation of Entropy-based Traffic Anomaly Detection. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement (IMC '08)*, pages 151–156, 2008.

43. V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *Computer Networks*, volume 31 (23–24), pages 2435–2463, 1999.
44. H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for Traffic Anomaly Detection. In *Proceedings of the ACM SIGMETRICS'07 Conference*, pages 109–120, 2007.
45. M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *LISA '99: Proceedings of the 13th USENIX conference on System administration*, pages 229–238, Berkeley, CA, USA, 1999.
46. O. Salem, S. Vaton, and A. Gravey. An Efficient Online Anomalies Detection Mechanism for High-Speed Networks. In *IEEE Workshop on Monitoring, Attack Detection and Migitation (MonAM 2007)*, November 2007.
47. R. Schweller, Z. Li, Y. Chen, Y. Gao, A. Gupta, E. Parsons, Y. Zhang, P. Dinda, M.-Y. Kao, and G. Memik. Reverse Hashing for High-speed Network Monitoring: algorithms, evaluation, and applications. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM 06)*, pages 1–12, April 2006.
48. R. Schweller, Z. Li, Y. Chen, Y. Gao, A. Gupta, Y. Zhang, P. Dinda, M.-Y. Kao, and G. Memik. Reversible Sketches: Enabling Monitoring and Analysis Over High-Speed Data Streams. *IEEE/ACM Transactions on Networking*, 15(5):1059–1072, Oct. 2007.
49. V. A. Siris and F. Papagalou. Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '04)*, volume 4, pages 2050–2054, Dallas, USA, 2004.
50. A. Tartakovsky and K. Hongjoong. Performance of Certain Decentralized Distributed Change Detection Procedures. In *Proceedings of the 9th International Conference on Information Fusion*, pages 1–8, 2006.
51. A. Tartakovsky, B. Rozovskii, R. Blazek, and H. Kim. Detection of Intrusion in Information Systems by Sequential Chang-Point Methods. *Statistical Methodology*, 3(3):252–340, 2006.
52. M. Thorup and Y. Zhang. Tabulation Based 4-Universal Hashing with Applications to Second Moment Estimation. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA '04)*, New Orleans, Louisiana, USA, January 2004.
53. H. Wang, D. Zhang, and K. G. Shin. Detecting SYN Flooding Attacks. In *Proc. of IEEE Infocom'02*, 2002.
54. H. Wang, D. Zhang, and K. G. Shin. SYN-dog: Sniffing SYN Flooding Sources. In *Proceedings of the 22th International Conference on Distributed Computing Systems (ICDCS'02)*, pages 421–429, 2002.
55. H. Wang, D. Zhang, and K. G. Shin. Change-Point Monitoring for the Detection of DoS Attacks. *IEEE Trans. On Dependable and Secure Computing*, 1(4):1993–2004, 2004.
56. G. Yan, L. Cuellar, S. Eidenbenz, and N. Hengartner. Blue-Watchdog: Detecting Bluetooth Worm Propagation in Public Areas. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'09)*, pages 317–326, 2009.
57. N. Ye, S. Vilbert, and Q. Chen. Computer intrusion detection through EWMA for autocorrelated and uncorrelated data. *IEEE Transactions on Reliability*, 51(1):75– 82, March 2003.
58. J. Yu, H. Lee, M.-S. Kim, and D. Park. Traffic flooding attack detection with SNMP MIB using SVM. *Computer Communications*, 31(17):4212–4219, 2008.
59. Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund. Online Identification of Hierarchical Heavy Hitters: Algorithms, Evaluation, and Applications. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 101–114, 2004.

