Flooding Attacks Detection and Victim Identification over High Speed Networks

Osman Salem, Ahmed Mehaoua
UFR Mathématiques et Informatique
Université Paris Descartes
Paris, France

Sandrine Vaton, Annie Gravey
Departement of Computer Science
TELECOM Bretagne
Brest, France

***Abstract*—With the rapid dependency on the internet for business, and the fast spread of powerful destructive DoS/DDoS attack tools, the detection and thwarting of these attacks is primordial for ISP, enterprises, hosting centers, etc. In this paper, we present the implementation of a new framework, for efficient detection and identification of flooding attacks over high speed links. To accomplish that, we apply multi-channel non-parametric *CUSUM* (MNP-CUSUM) over the shared counters in the proposed reversible sketch, in order to pinpoint flows with abrupt change via a new approach for sketch inversion. Shared counters are used to minimize the memory requirements and to identify the victim of flooding attacks. We apply our system at various real traces, some traces are provided by France Telecom (FT) within the framework of ANR-RNRT OSCAR project, other traces are collected in FT backbone network, during online experiments for testing and adjusting the proposed detection algorithms in this project. Our analysis results from real internet traffic, and from online implementation over Endace DAG 3.6ET sniffing card, show that our proposed architecture is able to quickly detect various kinds of flooding attacks and to disclose culprit flows with a high level of accuracy.**

I. Introduction

With the fast identification of operating systems and services vulnerabilities, and the availability of daily update (patch, service pack, etc.) which recover from discovered security holes, gaining and maintaining an illegal remote access become more and more difficult for non experimented attackers. However, flooding for denial of service attack does not require any skill in exploiting vulnerability, especially with the wide spread of free attack tools, able to make silent any web site. When a professional attacker may use Botnets of tens thousands of compromised machines, a non experimented attacker may rent compromised machines in few dollars at the web to prevent tracking. Usually, intentions behind these attacks are wide, and range from vandalism to extort money or obtain commercial advantage from the hosting centers.

Recent flooding attacks against commercial web sites, such as Yahoo and eBay, have motivated the development of several approaches to detect denial of service attacks. Flooding can easily lead to the disruption of critical infrastructure services and degrades the QoS in ISP network. Effective detection of anomalies in captured traffic requires the ability to separate them from normal traffic, i.e. some additional information about victim servers or attackers are required, in order to take the appropriate countermeasures and protect the access for normal users.

There are two approaches for network intrusion detection: signature based and anomaly detection. Signature based systems (e.g. Snort [22], Bro [23]) look for various malicious activities signatures inside each packet. This approach works in the same manner as antivirus, by searching inside the packet for strings, patterns and signatures of known attacks. Therefore, it cannot detect new malicious activity that it doesn't have their signatures.

Network anomaly detection is statistic based approach, and does not require prior knowledge about the signature of attacks. It operates by building a statistic model of normal network traffic in learning phase, and updating this model in every discrete time interval. Any inconsistent deviation from the built model is considered as anomaly. While a wide range of anomaly detection algorithms have been proposed to undermine attacks, the effectiveness of these models is largely depending on assumptions about the underlying traffic distributions parameters, and the built models to fit network traffic [16], [7]. They lack the capability of handling shape irregularities and unpredictable large fluctuations in real IP traffic.

When some of anomaly detection algorithms are host based and not scalable for high speed network, most of existing network anomaly detection algorithms are engaged in the detection of these anomalies as soon as possible. They are applied on the overall traffic, i.e. the whole stream of packets are aggregated in one flow, and a signal processing algorithm is applied over the analyzed time-series (number of UDP packets, SYN packets, etc.).

These algorithms only raise an alarm when there is a deviation larger than a predefined threshold in time-series analysis (EWMA [30], Holt-Winter [10], CUSUM [21], [4]). Given the variability of the traffic, most focus were on reducing false positive and delay detection. The application of change point detection algorithms to the overall traffic tend to be inaccurate in finding attacks, and does not reveal any information about attacker/victim for mitigation. Intuitively, if too many flows are aggregated, only the bigger anomalies will be visible. Furthermore, the identification task of victim or anomaly type without any information to separate malicious traffic from normal traffic is like searching for a needle in a haystack.

In this paper, we consider the problem of online detection

978-1-4244-4624-7/09/$25.00 ©2009 IEEE

of flooding attacks over high speed links, and the identification of victims in order to cope with attacks as soon as possible. Denial of Service (DoS) and it distributed version (DDoS) are the most common attacks over the Internet. This type of attack consumes the resources of a remote host or network for preventing legitimate users to access the service. Wide range of flooding attack tools are available to accomplish DoS and DDoS, by sending a significant amount of packets (TCP, UDP, ICMP) from one/many sources to one/many destinations.

When the aggregation of whole stream of packets in one flow may miss the detection of low intensity attacks, maintaining one time-series for each active flow over high speed link is not scalable, because the spatial and temporal complexities are not adequate for real time analysis. In response to these limitations, an efficient data structure based on $k - ary$ hash tables (Fig. 1), called sketch [9], [18], [20], was proposed and used to handle large state space, with a small amount of memory requirement and a linear computational (update/query) complexity. It is a multi-stage bloom filter based at random aggregation, where flows identifier (denoted by key) are hashed to index into a set of buckets in different stages using k different hash functions, usually chosen to reduce collision effect and to uniformly distribute keys.

The flow identifier (key) used in this paper is the destination IP address (DIP). This key is used to update the data structure of the k^{th} hash table by some value $(DIP, value_i)$. Sketch is usually used with universal hash functions for random aggregation, which are not reversible. Thus mean when monitoring high speed links, all keys must be recorded and verified. Unfortunately, this approach require storing the whole keys for further verification. To avoid storing keys, we use an additional Multi-Layer Reversible Sketch (*MLRS*) to identify the victim of flooding attacks.

In contrast to our previous work in [24], where our focus were only at TCP SYN flooding packets, the current work is able to detect a wide range of flooding attacks (TCP, UDP, ICMP). The proposed framework is based at online detection of change point in the time-series of residual observations, stored in shared counters of sketch data structure ($\Delta X_i = X_{i+1} - X_i$). To detect anomalies, we use a non-parametric version of multi-channel CUSUM (*MNP-CUSUM*) to detect change point in one of 3 time series inside every bucket of sketch. These 3 time-series monitor significant variation in the number of packets, bytes or flows between discrete time interval T. The motivation behind the choice of these parameters is that most of anomalies can be detected at least by one of these times series (bytes, packets, flows).

The proposed method begins by recording the value of the 3 preceding observations in the counters data structure contained in each bucket of the sketch, during T time interval. Afterward, *MNP-CUSUM* algorithm is used to check the presence of buckets which value deviates significantly from normal behavior. In fact, CUSUM algorithm is able to react quickly when observing abrupt changes in bucket counters, while being able to distinguish effect of attack deviation from usual traffic fluctuations, through dynamic update for the normal profile of each time-series. After the detection of anomalies by CUSUM algorithm, we recover associated keys to buckets with raised alarm by CUSUM, through exploiting bucket index in *MLRS*.

The rest of this paper is organized as follows. In section II, we give a glance at prior work in the area of anomalies detection. Section III gives a brief overview of *CMS* Sketch and MNP-CUSUM mechanisms that are related to our work. Section IV describes our proposed method for detecting change point in a reversible sketch. In section V, we present some analysis results from the application of our proposed framework over real traffic traces, some captured during online experimentations with well known attacks type and instant, other traces are provided by France Telecom. Finally, section VI presents concluding remarks and the future work.

II. Related works

Many important contributions have been proposed to undermine anomaly in network traffic [26], [29], [7], [19], [5]. When early approaches for anomaly detection were focused in the definition of models able to represent the traffic pattern, other advanced work aggregates the whole stream of packets in one flow, and apply change point detection algorithm to detect anomaly occurrence instant [26], [29]. These approaches have a good performance in terms of spatial and temporal complexities, but present the drawback of aggregating the whole traffic in one flow, especially over high speed network, where low intensity attacks can not be detected. Furthermore, discovering an attack instant without any additional information about the malicious source or victim are not enough for reacting against the attack. Usually, the amount of traffic is huge, and manual searching/extracting the malicious flows are difficult operations. Therefore, to increase the accuracy of these methods, and to pinpoint the victim or attacker, several approach have been proposed in the literature [20], [31], [14], [25]. This is the spirit of our work.

Schweller *et al.* in [25] proposed the use of random aggregation counters for more grained detection. To discover the victim of flooding, they propose a method based on galois field $GF(2^l)$ for mangling and for simplifying the sketch inversion. The proposed method is hardware efficient, and has been implemented in FPGA. In [8], BU *et al.* propose an extension to the previous method of sketch inversion by focusing on reducing the complexity of inversion method. In [14], Feng *et al.* 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 additional table.

All these proposed approaches have been used either to detect the heavy hitter flows (most frequent flows) or to detect abrupt deviation between two discrete interval via simple comparison. We can pinpoint that due to the complex mathematical analysis (signal processing, statistic, etc.) of existing detection approaches. In this paper, we will apply the recursive CUSUM algorithm over sketch. CUSUM is a simple algorithm with a linear compelxity $O(1)$. In [29], Wang *et al.* 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. In [26], Siris *et al.* evaluate and compare two anomaly detection algorithms (adaptive threshold and CUSUM) also for detecting the TCP SYN flooding. The result of the comparison is that *CUSUM* is more efficient for detecting low intensity attacks. In [6], [27], authors prove that non-parametric CUSUM is asymptotically optimal, where the detection delay reaches the theoretic minimum value for exponential distributions. When the threshold is well regulated, it reduces the average delay detection and increases the mean time between false alarms.

In this paper, we will use muti-channel non-parametric CUSUM over the sketch, in order to detect abrupt deviation in three time series of residual number of bytes, flows, packets. When an alarm is raised, we will reverse the sketch to uncover the victim server, in order to take further action for reacting against the ongoing attacks.

III. Background

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

A. Count-Min Sketch

let $S = s_1 s_2 \dots s_n$ be the set of input stream that arrives sequentially, item by item [13]. Each item $s_i = (\kappa_i, v_i)$ is identified by a key $\kappa_i \in U$ drawn from a fixed universe of items U. A reward (or frequency occurrence) value $v_i \in \mathbb{R}$ is associated with each key. The arrival of 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 function, chosen from the set of 2-universal hash function $H_j(\kappa_i) = \{((a_j\kappa_i + b_i) \bmod P_U) \bmod w^{'}\}$, to uniformly distribute κ_i over hash tables and to reduce collision. Parameter P_U is a prime number larger than the maximum number in universe, where Mersenne prime numbers of the form $2^i - 1$ are generally chosen for fast implementation.

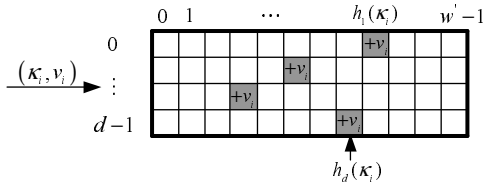


Fig. 1. Sketch data structure.

The Count-Min point query return an estimate of the counter for a given key, as the minimum of d counters value ($\hat{s}_k(\kappa_i) = \min_{0 \le j < d} \{C[j][h_j(\kappa_i)]\}$).

In our proposed framework, each bucket in this 2D table is a data structure, which contains many variables (X_{i,nb_bytes}, X_{i,nb_flows}, $X_{i,nb\_packets}$, X_{i-1,nb_bytes}, X_{i-1,nb_flows}, $X_{i-1,nb\_packets}$, $mean_i$, σ_i) and the three CUSUM function ($G_{ij}(T)$) used to detect heavy deviation in each time-series. From each exported flow record, we update the sketch with $(\kappa_i, nb\_bytes, 1, nb\_packets)$, where κ_i is the DIP, and 1 is for the number of flow. *CMS* query can estimate if a given DIP is under flooding attack by returning the minimum value of $G_{ij}(T)$ through all the buckets.

In *CMS* we use $d = \lceil \ln(1/\delta) \rceil$ pairwise independent universal hash function, where each one take κ_i as parameter and return a random integers in the range $w^{'} = [0, e/\epsilon]$. e is the base of the Neperian logarithm, and ε is the an error rate with probability less than δ. Thus, it maintains modest storage requirements of $O\left(\ln(1/delta) \times (1/\varepsilon)\right)$ count cells.

B. MNP-CUSUM change detection algorithm

We will use the sequential change point detection algorithm CUSUM, in order to detect flooding attacks instant, as it has been proven that CUSUM [17] has an optimal small detection delay, with a low computational overhead and small storage requirements.

Due to large variation in traffic pattern, and lack of consensus on network traffic characteristics, we consider non $i.i.d$ traffic characteristics, and we will apply a non-parametric version of M-CUSUM [17] over the 3 residual of observations in every sketch bucket. MNP-CUSUM is insensitive to topology and traffic patterns. It relies on two phases: training and detection. In training phase, it establishes and updates a dynamic behavior profiles for normal flows. In detection phase, it uses log likelihood ratio to detect any kind of abrupt deviation from well established profile. In multi-channel version of CUSUM, the algorithm is applied over many channels, and once an anomaly is detected in any channel, an alarm is raised.

Let $\{X_{ijk}^{nT}, 1 \le i \le d, 1 \le j \le w, 1 \le k \le 3\}$ be the value of each bucket for each of 3 monitored parameters during the n^{th} time interval. Observations X_{ijk}^{nT} are i.i.d with a *pdf* $f_{ijk,\gamma_0}(x)$ for $n < t_a$ (before attack occurrence) and with another *pdf* $f_{ijk,\gamma_1}(x)$ for $n \ge t_a$ (after attack), where t_a is the instant of attack detection. MNP-CUSUM tests statistical hypotheses H_{ijk} (eq. 1) to detect abrupt change in time-series k stored in bucket with index (i, j) at the time epoch $n = t_a$:

$$H_{ijk,0}: \quad \gamma_{ijk} = \gamma_0 \quad \text{versus} \quad H_{ijk,1}: \quad \gamma_{ijk} = \gamma_1 \qquad (1)$$

Where γ_0 and γ_1 are respectively the *pdf* parameters before and after change occurrence. The detection of anomaly is based on log likelihood ratio for an observation X_{ijk}^{nT} between the two hypotheses:

$$s_{ijk}^{nT} = \ln\left(\frac{P(\Delta X_{ijk}^{n} T|\gamma_1)}{P(\Delta X_{ijk}^{n} T|\gamma_0)}\right) \qquad (2)$$

If s_{ijk}^{nT} is positive, the monitored random variable ΔX_{ijk}^{nT} is changing the distribution from $H_{ijk,0}$ to $H_{ijk,1}$. Usually, a threshold h is defined for confirmation of the change, and the value of h is a tradeoff between the average delay detection and the mean time between false alarms.

The cumulative sum function is a summation of the log likelihood ratio:

$$S_{ijk}^{nT} = \sum_{r=1}^{n} s_{ijk}^{rT} \tag{3}$$

S_{ijk}^{nT} will decrease under normal condition ($s_{ijk}^{nT} < 0$), and increases if a change occurs ($s_{ijk}^{nT} > 0$). When the value of S_{ijk}^{nT} become greater than threshold h, a decision can be taken about the hypotheses ($H_{ijk,0}$ for normal condition and $H_{ijk,1}$ under attack condition). Therefore, the relevant information for detecting change lies in the difference between the value of the log-likelihood ratio and its current minimum value [26]. Hence the stopping time for the M-CUSUM algorithm is given by:

$$t_a = t_a(h) = \min\{n \geq 1 : G_{ijk}^{nT} \geq h\} \tag{4}$$

Where:

$$\underset{1\leq k\leq 3}{G_{ijk}^{nT}} = S_{ijk}^{nT} - m_{ijk}^{nT} \quad \text{and} \quad m_{ijk}^{nT} = \min_{\substack{1\leq i\leq d \\ 1\leq j\leq w'}} S_{ijk}^{nT} \tag{5}$$

The statistic function G_{ijk}^{nT} obeys the recursion:

$$G_{ijk}^{nT} = \left\{ 0, G_{ijk}^{(n-1)T} + \ln\left(\frac{P(\Delta X_{ijk}^{nT}|\gamma_1)}{P(\Delta X_{ijk}^{nT}|\gamma_0)}\right)\right\}^{+} \wedge\ G_{ijk}^{0} = 0 \tag{6}$$

Where $\{y\}^{+} = \max(0, y)$.

Like the distribution function of ΔX_{ijk}^{nT} is unknown, the log-likelihood ratio in eq. (6) must be replaced by a statistic function $u(\Delta X_{ijk}^{nT})$ with the same properties, i.e. $u(\Delta X_{ijk}^{nT})$ must be negative under $H_{ijk,0}$ and positive under $H_{ijk,1}$. An appropriate function for detecting change in the mean is $u(\Delta X_{ijk}^{nT}) = \Delta X_{ij}^{nT} - (\mu_{ijk} + \varepsilon\sigma_{ijk})$. As a result, the decision function becomes:

$$G_{ijk}^{nT} = \left\{ G_{ijk}^{(n-1)T} + (\Delta X_{ijk}^{nT} - (\mu_{ijk} + \varepsilon\sigma_{ijk}))\right\}^{+} \tag{7}$$

We apply MNP-CUSUM over sketch, and we consider three time-series (nb_bytes, nb_flows, nb_packets) in a discrete time interval T of 1 minute. At the end of each interval, we compute the value of MNP-CUSUM functions G_{ijk}^{nT}. If its value is larger than a predefined threshold h (if $G_{ijk}^{nT} > h$) then an alarm is raised. μ_{ijk} and σ_{ijk} are the mean and standard deviation of the corresponding bucket estimated in the previous interval, and they are updated in each interval as shown in eq. (8):

$$\begin{aligned} \mu_{ijk}^{nT} &= \alpha\mu_{ijk}^{(n-1)T} + (1-\alpha)\Delta X_{ijk}^{nT} \\ var_{ijk}^{nT} &= \beta var_{ijk}^{(n-1)T} + (1-\beta)(\Delta X_{ijk}^{nT} - \mu_{ijk}^{nT})^2 \end{aligned} \tag{8}$$

A detection algorithm should have a low false alarm rate FAR and small detection delay. In [27], [6], it was proven that CUSUM minimizes the average delay detection $ADD_{t_0}(t_a)$ for a given false alarm rate FAR. The FAR increases by decreasing the speed of detection, and a trade-off between low FAR and minimum delay detection is required. The threshold value should be chosen to minimize delay detection given a fixed FAR. It is worth noting that value of threshold h controls the sensitivity of the attack detection, hence large value of h decreases the FAR, but true attacks may also completely missed.

IV. Proposed approach

Our proposed framework is based at 2 data summary architecture: a Multi-Layer Reversible Sketch (*MLRS*) and a Count-Min Sketch (*CMS*) as shown in figure 2. Operations of the proposed framework are performed by two steps. First, it continuously updates the two sketches (*MLRS* and *CMS*) data structure from input data stream $(DIP, nb\_bytes, 1, nb\_packets)$ for a fixed time interval T. Secondly it applies MNP-CUSUM in the background at each bucket to detect anomalies in each time series. Afterward we identify and output DIP that mapped to buckets with a CUSUM triggered alarm.

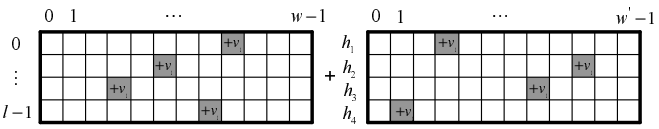


Fig. 2. Multi-Layer Reversible Sketch *MLRS* and *CMS* sketch.

In order to detect victim servers of flooding attacks, we use the destination IP (DIP) as key for updating the shared counters in each bucket of the proposed sketch. At the end of each interval, the MNP-CUSUM analyzes the time-series of these 3 counters and raises an alarm in buckets with abrupt change. However, with the presence of collision inside the *MLRS*, the abrupt change may be caused by collision, and a verification through the *CMS* sketch is required. Only if all shared counters associated with DIP in the second *CMS* raises an alarm, we conclude to flooding attack against this server.

Our idea to reverse sketch is based at exploiting index in an additional multi-layer reversible sketch (Figure 2), where indexes are used to store keys. Related works for sketch inversion can be found in [25], [14]. In fact, the *MLRS* is used in the same way of *CMS* sketch, where each flow increments its 3 counters in the two 2D tables. In *MLRS* each DIP has l counter (one by layer), where we split the DIP of 32 bit into $l \times w$ bit, with $w = 2^P$, and $l = \lceil N/P \rceil$. P is the number of bits used to split the key, and w is used as layer width in *MLRS*. The update procedure is summarized in algorithm 1.

Algorithm 1 Sketches Update procedure

```
Mkey = encrypt(DIP, key);
for i = 0 to d − 1 do
    j = univ_hash_i(Mkey);
    CMS.counter[i][j].X[k] += κ_i;
end for
for j = 0 to l − 1 do
    MLRS.counter[i][Mkey&(2^P − 1)].X[k] += κ_i;
    Mkey >>= P;
end for
```

To reveal the victim server (or DIP with raised alarm by MNP-CUSUM), we can release hierarchical search procedure in *MLRS*. If we don't find at least one bucket with raised alarm in each of the i^{th} ($i \leq l-1$) first layers of *MLRS*, there is no need to continue searching in other deep layer or through the second *CMS* sketch. Malicious flows must have one alarmed bucket in each layer.

We will begin by the simple case, where we assume that there is at most one bucket with CUSUM raised alarm in each layer as shown in Fig 2. To recover key, we concatenate the l index in *MLRS* and we get the DIP. We can not be sure of suspect key before verification, where due to collision with other IP prefix, their accumulated value become large. The suspect key is verified through hashing and verification (by count-min query of CUSUM function) in the *CMS* for confirmation.

In general, even with a different value of width (e.g. 2^{12} or 2^{14}) for the *MLRS*, many buckets in different layers will be subject to collision occurrence, and in some case, we will be found with a bigger set of keys to verify through *CMS* than the original one. Nevertheless, it is important to notify that even if the set of suspect key is larger than departure one, it requires a small memory and with respect to original list.

To resolve this problem and reduce collision in *MLRS*, we use use the encryption algorithm optimized version RC4 (Ron's code [15]) available in [2]. Encryption is a bijective function which randomizes the input data in an attempt to destroy correlation between adjacent DIP, and may disperse IP address with the same prefix uniformly at all available bucket (table I). After encryption $MDIP = E(DIP)$, the new value is used as a key to update the sketch, and if there is any alarm raised by CUSUM in all layer, the decryption operation $DIP = D(MDIP)$ recover the DIP for verification through the *CMS*.

192.168.92.40	10010100101001011110100010011011
192.168.92.41	10101011011001000011001000100110
192.168.92.42	10010110111011000010010010101110
192.168.92.43	00100000001101001000000001101101
...	...

TABLE I
ENCRYPTION OF *DIP* BY OPTIMIZED VERSION OF RC4.

At the end of each time interval T, MNP-CUSUM run in the background to detect heavy deviation in the function G_{ijk}^{nT}. If there is a raised alarms in all layers of *MLRS*, decryption function is called to recover the value of DIP, and a verification through count-min query over the *CMS* is realized to verify raised alarms through all hash function. Algorithm 2 shows the search and verification procedure, where boolean alarm variable is used to indicate the state of CUSUM function.

Algorithm 2 Search and verification procedure

```
1:  for i = 0 to w − 1 do
2:    if (MLRS.counter[0][i].Alarm) then
3:      for j = 0 to w − 1 do
4:        if (MLRS.counter[1][j].Alarm) then
5:          MDIP = (j << P) | i;
6:          Alarm=cms_query(CMS, DIP);
7:          if (Alarm) then
8:            DIP=decrypt(MDIP, key);
9:            output(DIP)
10:         end if
11:       end if
12:     end for
13:   end if
14: end for
```

V. EXPERIMENTS RESULTS

In this section, we start by evaluating the capability of our proposed mechanism to detect flooding and to identify the victim IP address by using *MLRS* and *MNP-CUSUM* presented in the previous sections. We have implemented MNP-CUSUM over sketch in C using the code of CMS available from [1]. We applied the proposed algorithm over many public traces (Abilene, Auckland, etc.) available from [3], and other traces used in OSCAR RNRT French Research project. We also use traces collected during online experimentations conducted for testing detection ratio of the proposed algorithms. Our results are encouraging in terms of accuracy and response time, especially when comparing them to the result obtained by the application of single CUSUM over the aggregated traffic in one flow.

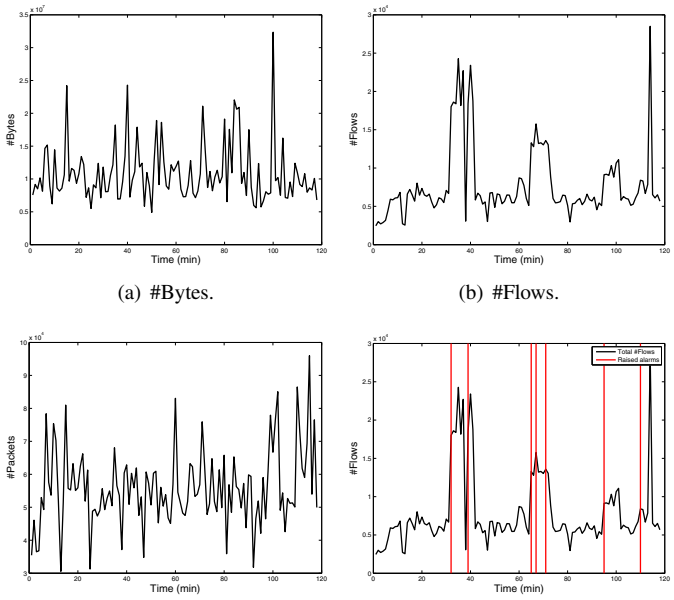
For this work, we present the result with 2 different traces: one captured during online experimentations of OSCAR project, with well known attack flooding instant and victim server, and the second are provided by FT, and collected at their infrastructure. All experiments were performed using Ubuntu box with an Intel core 2 DUO (E4500) with 2.2 Ghz and 3 GB of RAM and 750 GB SATA disks.

In order to reduce spatial and temporal complexity of the proposed algorithms in OSCAR project, partners decide to enhance the capture process of high speed sniffing card (Endace card), by adding a small C program for transforming captured packets in one 1 min into flows. In low speed network from 100 Mbit to 1 Gbit, processing of distinct network packet is possible, however for high speed networks, this is hardly feasible. Therefore, we keep approximately the same definition of flow used by Netflow [12] in Cisco router or the IPFIX [11] protocol. A flow record in *OSCARFIX* is a set of unidirectional stream of packets moving from one source to a destination, and is identified through the same five tuple (source IP, destination IP, protocol, source port, destination port), but with counters for the accumulated value of SYN, SYNACK, RST & FIN, and we took a timeout value of 1 minute. Even if the flow doesn't finish in the current time interval (1 min), subsequent packets are considered as member of new flow in the next minute. Our analysis will use flow records data in order to detect flooding attacks, as they are widely deployed in many monitoring techniques due to the wide deployment of CISCO Netflow.

The parameters we considered for the MNP-CUSUM algorithm were: threshold $h = 10$, $\alpha = 0.9$ as in [26], $\beta = 0.9$ and $\varepsilon = 2$. Sketches parameters were $P = 10$, $w = 1024$, $l = 4$, $d = 4$ hashing functions from the set of 2-universal hash function, and with the use of tabulation [28]. The first trace is two hours of OSCARFIX records traces with P2P traffic and with the contribution of many planetlab machines for generating background traffic. This trace was collected using Endace DAG 3.6ET and a GPS-synchronized timestamp. Many French research laboratories (project partners) have contributed in this experiment. The attack instant in this trace are known and generated by FT. Figure 3 shows the time-series of the total number of bytes, packets, and flows in interval of 1 min, as well the raised alarms instant in MNP-CUSUM (in Figures 3(a), 3(b), 3(c) & 3(d) respectively). Figures 4(a), 4(b) & 4(c) show the residuals number of bytes, flows, packets. The malicious flows received by the identified victim servers, residuals of malicious flows and packets are shown in figures 4(d), 4(e) & 4(f). After the identification of victim server, we extract the number of: flows, bytes & packets received by each victim. The victim servers were: 10.0.0.1,10.0.0.2, 10.0.0.3, 10.0.0.4. The delay between each attack is approximately 30 minutes and the attack duration range from 5 to 10 minutes. This trace contains four attacks with variable intensity (form high to small rate) and can be used to verify the accuracy of detection with high and low flooding attacks rate. The variation curves of the number of flows, packets and bytes received by the victims have exactly the same shape as in figures 4(d) for the three time-series, and the same variation value for the nb_flows & nb_packets as shown in their residuals in figures 4(e) & 4(f). It is logical to have the same value for nb_flows and nb_packets, as it is a TCP SYN flooding attack with randomly spoofed IP address and with one SYN packet in each flow.

Learning phase (or initial phase) is very important for change point detection algorithm, where statistical parameters (mean, variance) are estimated. However, in this trace, all the victim servers do not receive any packets outside the attack, and does not have any learning phase. To resolve this problem, we initialize the statistical parameters of servers that doesn't receive any packets for the first several minutes, to 0.1% of the statistical parameters associated to the aggregation of the overall traffic. With the current value of parameters, no false negative has been occurred during the analysis of this trace.

Our second experiment considers anonymized traces collected over high speed network. We realize the same analysis study and present the results in figures 5 & 6. Figures 5(a), 5(b) & 5(c) show the variation of the total number of bytes, flows and packets respectively. The raised alarms by NMP-CUSUM are shown in figure 5(d). After the identification and the extraction of malicious flows received by victims, all anomalies were manually inspected and verified. Over the 6 raised alarms shown in figure 5(d), four of these alarms are raised by legitimate traffic. In fact, there is an abrupt change either in the number of bytes or packets for these hosts caused by legitimate traffic. These false alertes can easily be avoided



(a) #Bytes. (b) #Flows.

(c) #Packets. (d) MNP-CUSUM Raised alarm.

Fig. 3. Total number of Bytes, Flows, Packets & Raised alarms.

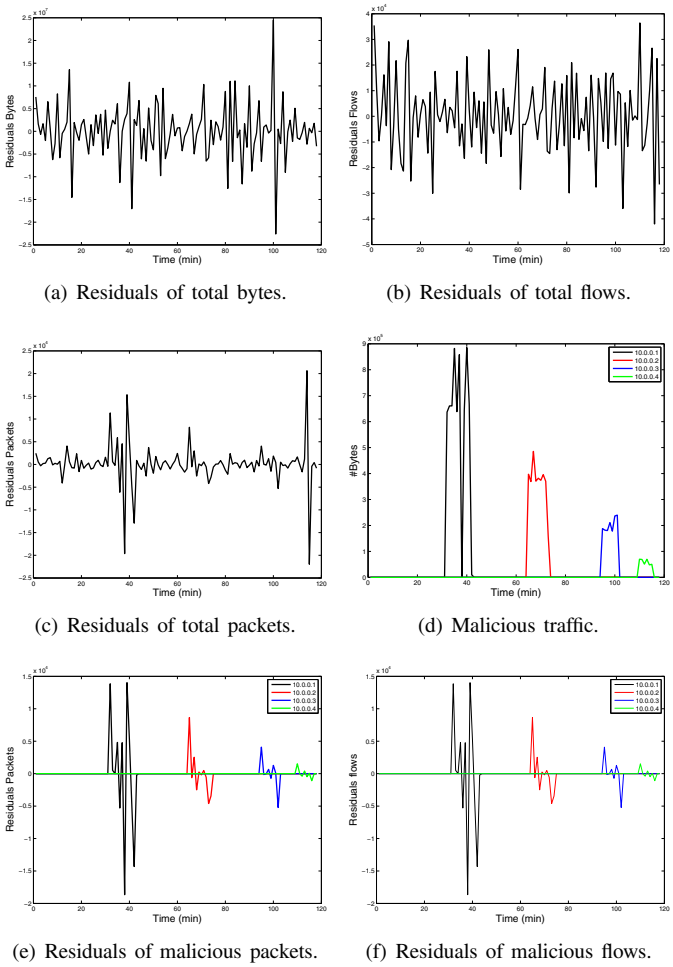
by adjusting the initial value of statistical parameters used by cusum (mean, variance), as well as by increasing the value of the threshold h or the ε parameter in MNP-CUSUM. Only one host with two raised alarms by MNP-CUSUM is under attack, and the corresponding malicious flows & packets are shown in figures 6(d) and 6(e). The residual number of flows is given in figure 6(f).

VI. Conclusion and Perspectives

In this paper, we proposed a new approach based at sketch and MNP-CUSUM for flooding attacks detection (TCP, UDP and ICMP), and victim identification over high speed link. Existing attacks detection formalisms focus on attack detection instant without giving any information about the victim server. Our proposed framework is able to automatically pinpoint the victim server and the malicious IP flows responsible of anomaly, through exploiting bucket index in an additional multi-layer sketch. The detection tool which implements the proposed architecture, has very low computational cost and small memory requirement, and we demonstrate its ability to detect low intensity attacks. Therefore, the proposed mechanism can be used to detect flooding near to the source.

Our online evaluations show that many raised false alarms are due to legitimate traffic. To regulate the number of false alarms, one can adjust the value of the threshold to control the sensibility of MNP-CUSUM.

In our ongoing work, we are interested in exploiting dependencies between different metrics for anomaly classification (Flooding, Scan, etc.). Furthermore, monitoring a sliding window for time interval (multi-scale) may allow discovering other kinds of anomalies, such as slow scan activity, where a malicious user takes his time to scan existing services with a very slow rate, to avoid detection by the existing intrusion

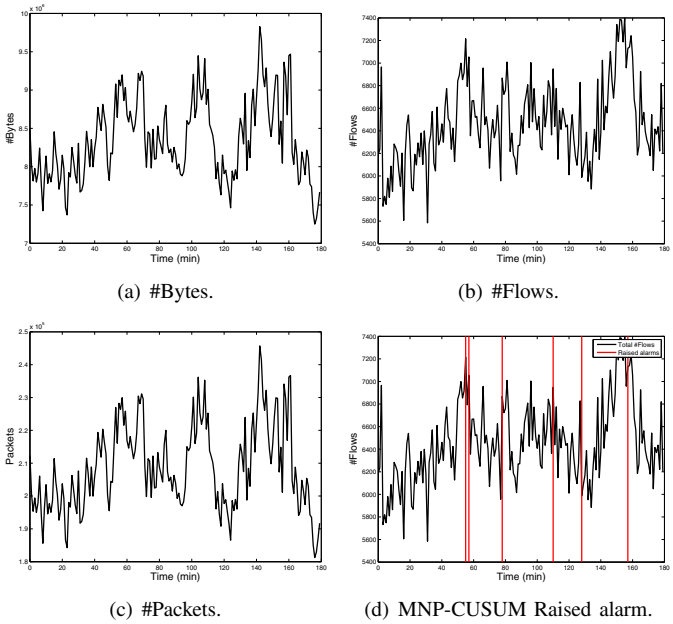


(a) Residuals of total bytes. (b) Residuals of total flows.

(c) Residuals of total packets. (d) Malicious traffic.

(e) Residuals of malicious packets. (f) Residuals of malicious flows.

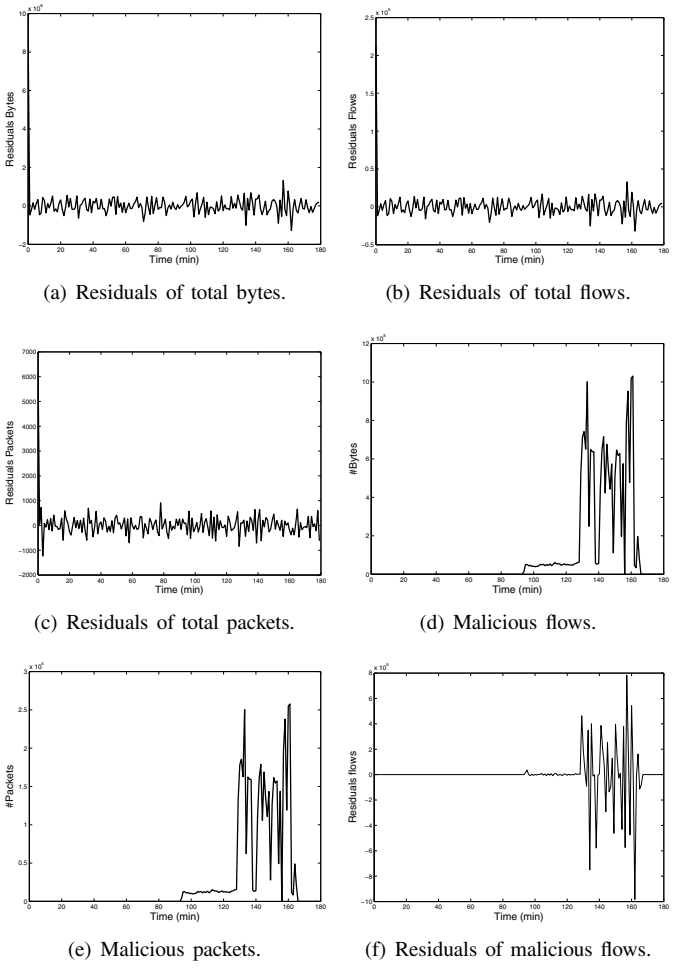
Fig. 4. Analysis results for online experiment traces.



(a) #Bytes. (b) #Flows.

(c) #Packets. (d) MNP-CUSUM Raised alarm.

Fig. 5. Total number of Bytes, Flows, Packets & Raised alarms.



(a) Residuals of total bytes. (b) Residuals of total flows.

(c) Residuals of total packets. (d) Malicious flows.

(e) Malicious packets. (f) Residuals of malicious flows.

Fig. 6. Analysis results.

detection system. Scan activities can easily be detected by a slight modification of our proposed approach, by using another sketch data structure for monitoring the source IP address.

ACKNOWLEDGMENT

This work has been partially funded by the French National Research Agency through the OSCAR project.

REFERENCES

[1] Count-min sketch source code. http://www.cs.rutgers.edu/ muthu/massdal-code-index.html.
[2] Optimized rc4 code. http://www.zengl.net/freeswan/.
[3] Traces archive. http://pma.nlanr.net/Special/.
[4] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes : Theory and Application*. Prentice-Hall Inc, 1993.
[5] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina. Impact of packet sampling on anomaly detection metrics. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 159–164, 2006.
[6] B. Brodsky and B. Darkhovsky. *Nonparametric Methods in Change Point Problems*, volume 243. Kluwer Academic Publishers, 1993.
[7] J. D. Brutlag. Aberrant behavior detection in time series for network monitoring. In *LISA '00: Proceedings of the 14th USENIX conference on System administration*, pages 139–146, Berkeley, CA, USA, 2000.
[8] T. Bu, J. Cao, A. Chen, and P. P. C. 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.

[9] 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, London, UK, 2002. Springer-Verlag.

[10] C. Chatfield. *The Analysis of Time Series: An Introduction*. CRC Press LLC, 6 edition, 2003.

[11] 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.

[12] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes. Cisco Systems NetFlow Services Export Version 9. RFC 3954, Oct. 2004.

[13] 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.

[14] 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, Guangzhou, Guangdong, China, December 2006.

[15] 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, London, UK, 2001. Springer-Verlag.

[16] H. Hajji. Statistical analysis of network traffic for adaptive faults detection. *IEEE Transactions on Neural Networks*, 16(5):1053–1063, September 2005.

[17] 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, December 2004.

[18] 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, New York, NY, USA, 2003.

[19] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *SIGCOMM Comput. Commun. Rev.*, 35(4):217–228, 2005.

[20] 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, New York, NY, USA, 2006. ACM Press.

[21] E. S. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[22] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *Computer Networks*, volume 31 (23–24), pages 2435–2463, 1999.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] A. Tartakovsky. Asymptotic performance of a multichart cusum test under false alarm probability constraint. In *Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference*, pages 320–325, Seville, Spain, December 2005.

[28] 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.

[29] 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, Washington, DC, USA, 2002. IEEE Computer Society.

[30] N. Ye, S. Vilbert, and Q. Chen. omputer intrusion detection through ewma for autocorrelated and uncorrelated data. *IEEE TRANSACTIONS ON RELIABILITY*, 51(1):75– 82, March 2003.

[31] 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.