The High Speed Intrusion Detection System

Yan Gao, Zhichun Li, and Yan Chen

Department of Computer Science
Northwestern University
1890 Maple Ave, Evanston, IL, USA 60201
{yga751, lizc, ychen}@cs.northwestern.edu

Abstract. Traffic anomalies and attacks are commonplace in today's networks, and identifying them rapidly and accurately is critical for large network operators. It was estimated that malicious code (viruses, worms and Trojan horses) caused over $28 billion in economic losses in 2003, and will grow to over $75 billion in economic losses by 2007 [1].

1 Introduction

Traffic anomalies and attacks are commonplace in today's networks, and identifying them rapidly and accurately is critical for large network operators. It was estimated that malicious code (viruses, worms and Trojan horses) caused over $28 billion in economic losses in 2003, and will grow to over $75 billion in economic losses by 2007 [1].

According to several IDS surveys [2–5], most existing IDSs reside on single host or low-end routers, examining application-level [6, 7], system-level [8–10] logs, or the sniffed network packets [11, 12] because such detailed information can identify attacks on individual machines. However, today's fast propagation of viruses/worms (*e.g.*, Sapphire worm) can infect most of the vulnerable machines in the Internet within ten minutes [13] or even less than 30 seconds with some highly virulent techniques [14]. Thus it is crucial to identify such outbreaks in their early phases, which can only possibly be achieved by detection at high speed backbone routers instead of at end hosts [15]. Furthermore, for the DDoS attacks, although the end hosts can detect and block them, they still can consume much of the network bandwidth or capacity to make the network unusable. In fact, it is very hard to implement certain detection techniques, such as those for port scanning, at end hosts. Routers can take appropriate actions to block attack traffic and quarantine infected machines. However, the existing schemes are not scalable to the link speeds and number of flows for high-speed networks.

For the high speed network, *e.g.* OC-192 networks, each packet only has very limit time to be handled, *e.g.* 32ns for OC-192 networks. Therefore, this requires the detection approach to (1) only use a small amount of memory, since although DRAM is quite cheap nowadays, memory such as SRAM with access time commensurate with IP traffic is still quite expensive. And (2) use very few memory accesses per packet or flow record. Both requirements are well known in designing IP lookups and network QoS. For high speed network intrusion detection system we need to consider these two requirements.

Moreover, most of the existing approaches are signature based, which cannot detect unknown network attacks. Statistical IDSs are therefore proposed to detect anomalous behaviors. Current systems are mostly designed to detect based on the overall traffic [16–20], thus they tend to be inaccurate or cannot find real attack flows for mitigation even when spotting anomalies. There are also a few flow-level detection schemes [11, 12, 21], but they are not scalable and thus are themselves vulnerable to attacks. In the other words, they can detect specific types of attacks, but will not perform well when facing a mixture of attacks as in the real world. For instance, state-of-the-art port scan detection using sequential hypothesis testing needs to maintain a per-source-IP table for detection [21], which is vulnerable to DoS attacks with randomly spoofed IP addresses, especially on high-speed networks.

To address these challenges, we propose a new paradigm called High speed Router based Anomaly and Intrusion Detection system, HRAID, which based on reversible sketches [22, 23] and two-demensional sketches. Our approach aims to use change detection to identify the horizontal scan, vertical scan and DoS attacks (SYN Flooding) simultaneously in high speed network, as well as get the accurate attack flows' information.

Reversible sketch is an efficient tool for data streaming computation, recording flow-level traffic as the basis for statistical anomaly detection. The basic idea is to hash intelligently by modifying the input keys and/or hashing functions so that we can recover the keys with certain properties like heavy changes without sacrificing the detection accuracy. Online heavy change detection is a hot topic that has popped up in recent years. By using a reversible sketch we can detect the heavy changes effectively. Our detection scheme includes two stages: online recording, which only uses small amounts of memory and memory accesses per flow and can be efficiently implemented in hardware; and detection, which recovers the heavy change keys (*e.g.* Source IP) in the background in a pseudo real-time manner.

Although heavy change flows can be inferred by reversible sketch, such a structure is still a one-dimensional sketch. It is difficult to differentiate one kind of attack from another by using only one-dimensional information. For example, if a large number of unresponsive connections have the same source IP and destination IP, it could be either a vertical scan or a non-spoof SYN flooding. Without other information we can not draw the conclusion arbitrarily which attack it is. In this case two-dimensional sketch is created. It is the extension of the K-ary sketch proposed in [20, 24], in which each hash table of the K-ary sketch is extended to be a two dimensional hash table (matrix). Based on two independent hash functions, we can hash two different features: *e.g.* one is the joint of source IP and destination IP, another is the destination port, to two different hashes and get more information about the flow. In Section 2.4, we will see that under a certain assumption this special design can help us distinguish the different types of attacks, such as SYN flooding vs. vertical scan.

Based on these data streaming data structures, we analyzed attributes in TCP/IP headers and selected an optimal small set of metrics for flow level anomaly change detection. The evaluation results show our approach had high speed, small memory consumption and high accuracy compared with the other state-of-art intrusion detection approaches, such as Threshold Random Walk (TRW) [21], Change-point detection

(CPD) [18], and BackScatter [16], *etc.*. Furthermore we compared our approach with the ideal infinite memory version of flow level detection using similar algorithm, the experiment result shows the sketch-based approach can get almost same result comparing with the precise one. Based on our manual attack inspection and nonexistent destination IP information or honeypot, we further found our approach detected almost all the attacks in the data sets.

The organization of this paper is as follows. In Section 2 we introduce our overall design. In Section 3, we discuss the threat model and some specific technique implemented in this paper. We describe the experimental evaluation of HRAID and the comparisons with other approaches in Section 4. In Section 5 we survey related work, and finally conclude in Section 6.

2 Architecture

2.1 Generic Framework

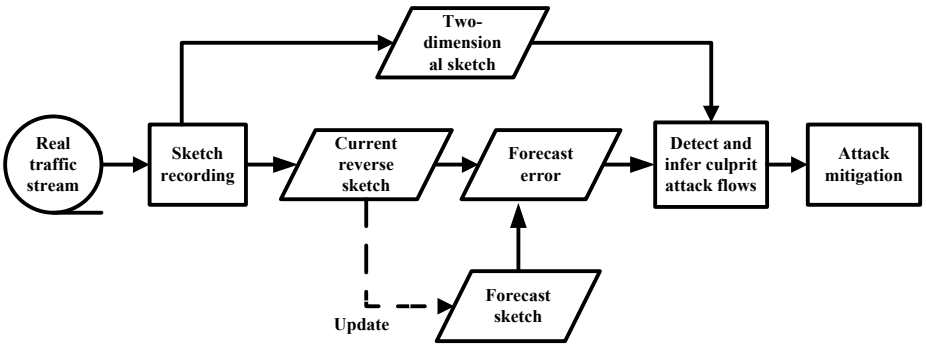


Fig. 1. System design framework.

Figure 1 shows the generic system design. With traffic recorded in sketches, we can apply different time series analysis methods to obtain the forecast sketches for change detection, in terms of the network fluctuation and speed requirements. In order to keep the system simple here we choose exponentially-weighted moving average (EWMA) and Holt-Winter (HW) as the forecast methods. Such forecast time series analysis method and change detection mechanism can help remove noises from detections and make the system not that sensitive to threshold. By subtracting the forecast sketch from the current one, we obtain the forecast error sketches. Intuitively, a large forecast error implies there is an anomaly, thus the forecast error is the key metric for detection in our system. Moreover, we adopt the two-dimensional sketches to further distinguish different types of attacks, such as SYN Flooding vs. vertical scan, and SYN flooding vs. horizontal scan. We refer to Section 2.4 for details about the two-dimensional sketch.

In the following sections, we will first introduce the K-ary sketch, which is the base of the reversible sketch and two-dimensional sketch. Then we will describe the reversible sketch and two-dimensional sketch. Both of them are different extension of K-ary sketch. To the best of our knowledge, we are the *first* ones to enhance the K-ary sketch to two dimensions, and use it to distinguish different types of attacks.

2.2 *K*-ary Sketch

The sketch, a recently proposed data structure, has proven to be useful in many data stream computation applications [25–28]. We have designed, implemented and evaluated a variant of the sketch, namely the k-ary sketch [24], and showed how to detect heavy changes in massive data streams with small memory consumption, constant update/query complexity, and provably accurate estimation guarantees [24]. It is also flexible and can be applied with various forecast models to detect anomalies.

We adopt one of the most general data stream models, the Turnstile Model [29]. Let $I = \alpha_1, \alpha_2, ...$, be an input stream that arrives sequentially, item by item. Each item $\alpha_i = (a_i, u_i)$ consists of a key $a_i \in [u] \equiv 0, 1, \ldots, u-1$, and an (possibly negative) update $u_i \in \mathbb{R}$. Associated with each key $a \in [u]$ is a time varying signal $A[a]$. The arrival of each new data item (a_i, u_i) causes the underlying signal $A[a_i]$ to be updated: $A[a_i] += u_i$. The goal of anomaly detection is to identify all those signals with anomalous behavior.

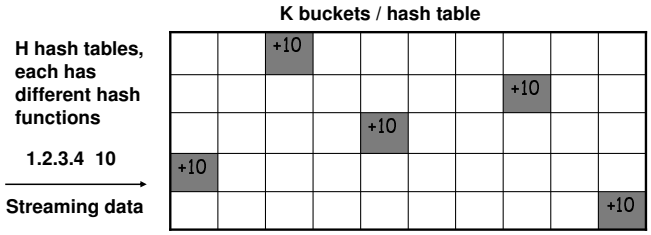


Fig. 2. A sample k-ary sketch structure and its UPDATE operation for a sample input stream with an item (1.2.3.4 10). The key is 1.2.3.4 and the value is 10.

The above model is very general and one can instantiate it in many ways with specific definitions of the keys and updates. In the context of network anomaly detection, the key can be defined using one or more fields in packet headers, such as source and destination IP addresses, source and destination port numbers and protocol number, etc. The updates can be the size of a packet, the total bytes or packets in a flow (when flow-level data is available).

As shown in Figure 2, a k-ary sketch S consists of a $H \times K$ table of registers: $T_S[i][j]$ ($i \in [H], j \in [K]$). Each row $T_S[i][\cdot]$ ($i \in [H]$) is associated with a hash function from $[u]$ to $[K]$: h_i. We can view it as an array of hash tables. The K-ary sketch has three basic functions: UPDATE to update a sketch, ESTIMATE to reconstruct v_a for a given key a, and COMBINE to compute the linear combination of multiple sketches.

Generally, in anomaly detection: UPDATE is in the monitoring module to update the observed sketch; ESTIMATE is in the anomaly detection module to reconstruct the signal series for statistical detection; COMBINE is also in the anomaly detection module to implement various statistical models, and to aggregate signals at various levels. Among them, UPDATE is used most frequently, and has the most stringent real-time requirement. For example, given an OC-192 link (10Gbps), each 40-byte TCP packet only has 32 ns to proceed [30].

2.3 Sketch for High-speed Network Monitoring

Although K-ary sketch has good linear properties and can accurately estimate the value of keys, it is not reversible. This means it is very hard for us to know which flows (keys) are above a certain threshold. For both heavy hitter and heavy changes, the ultimate results we want to obtain are which flows are heavy hitters or cause the heavy changes, e.g., in SYN Flooding, we want to know which victim is being attacked so we can choose some countermeasure to mitigate the attack.

Based on our experience of K-ary sketch we designed a new data structure called reversible sketch, for details please refer to our IMC03 paper [22] and technical report [23]. We only give a brief overview here. The problem with the K-ary sketch is that with standard hashing techniques the change detection cannot be performed efficiently. Naively, this would require either exhaustively testing all possible keys or recording and testing all the data stream keys and the corresponding sketches. Unfortunately, neither option is scalable. Therefore, we introduce the IP mangling and modular hashing to enhance the K-ary sketch to make it reversible.

For modular hashing, instead of hashing the entire key in $[n]$ directly to a bucket in $[m]$, we partition the key into q words, each word of size $\frac{1}{q}\log n$ bits. Each word is then hashed separately with different hash functions which map from space $[n^{\frac{1}{q}}]$ to $[m^{\frac{1}{q}}]$. For example, a 32-bit IP address can be partitioned into $q = 4$ words, each of 8 bits. Four independent hash functions are then chosen which map from space $[2^8]$ to $[2^3]$. The results of each of the hash functions are then concatenated to form the final hash. In our example, the final hash value would consist of 12 bits, deriving each of its 3 bits from the separate hash functions $h_{i,1}, h_{i,2}, h_{i,3}, h_{i,4}$. Then, for the change detection, we can first test possible sub-keys of each word separately, then combine the possible words together to form the whole keys. By using this method, we can get the result in a much smaller search space.

However, since there are strong spatial localities of the IP addresses in network traffic, *i.e.*, many simultaneous flows share the same prefix and only vary in the last few bits of their source/destination IP addresses. This increases the probability of collisions. So, we introduce the IP mangling, a universal hashing scheme based on the Galois Extension Field [**?**], which can uniformly distribute the IP addresses sharing same prefix to the 2^{32} key space.

As shown in Figure 3, our design based on reversible sketches includes two phases for the heavy change detection. The first phase is in the critical path, which updates each data item in the stream into the sketches online. In the second phase we run the detection daemon in the background to output the heavy flows that are above certain threshold.

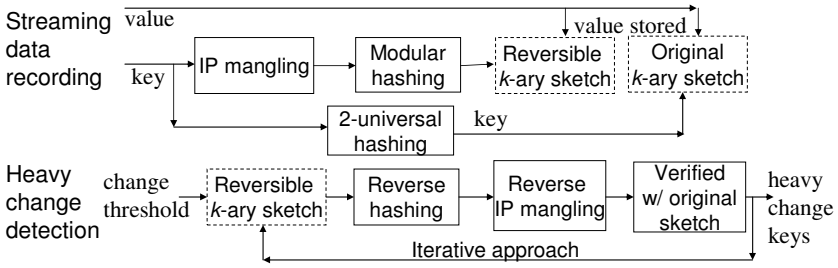


Fig. 3. Architecture of the reversible k-ary sketch based heavy change detection system for massive data streams

For the data stream recording stage, our prototype single FPGA board implementation can achieve a throughput of over 16 Gbps for 40-byte-packet streams (the worst case). For the second stage, when we evaluate our system for inferring 1,000 heavy change keys, with network traces obtained from two large edge routers with an OC-12 link or higher, our schemes find more than 99% of the heavy change keys(32bit) with less than a 0.2% false positive rate within 13 seconds. Both results show that the reversible sketch can be used in online traffic monitoring and accurate change/anomaly detection over massive data streams on high speed links.

2.4 Two Dimensional K-ary Sketch

A major challenge for anomaly detection is that the traffic anomalies often have very complicated structures: they are often multidimensional (*i.e.*, they can only be exposed when we examine traffic with specific combinations of IP addresses, port numbers, and protocols). For example, if we do not have the distribution of how many ports the attacker visited, it will be hard to distinguish non-IP-Spoofing SYN flooding attacks and vertical scans. Both of them could be a hacker sending a lot of SYN packets to the target. The difference lies in whether the hacker sends to one or two ports (SYN flooding) or many ports (vertical scan). So it is essential to know the port distribution given the specific source IP and destination IP ($\{$SIP,DIP$\}$) pair, to distinguish the two kinds of attacks. Because of this, we sometimes need to know the conditional values or the conditional distributions of one dimension given another.

Yin Zhang has proposed hierarchical heavy hitters (HHHs), which could detect the hierarchical, multidimensional heavy hitters (possible heavy changes) in general. However, in some worst cases, that approach will consume large amount of memory, such as 100MB or more. If we only want to obtain the conditional values or distributions, that approach is too expensive for us.

Leveraging over our previous experience of the K-ary sketch, we can extend it to the two-dimensional cases. Instead of using H independent one-dimensional hash tables, we use H independent two-dimensional hash tables (matrix), as shown in Figure 4. For each two-dimensional hash matrix, we can hash two groups of fields from IP headers into it. For instance, let's continue the previous example on discriminate SYN floodings and vertical scans. The x dimension is the combination of source IP and destination IP $\{$SIP,DIP$\}$, and the y dimension is the destination port $\{$DP$\}$. When each packet comes,

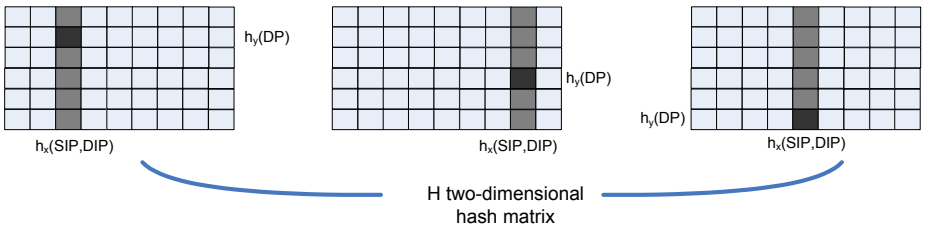


Fig. 4. Architecture of the two-dimensional k-ary sketch

we locate it in the matrix by the two independent hash mappings, as shown in Figure 5. In the same way we can update all the H matrices in data stream recording stage.

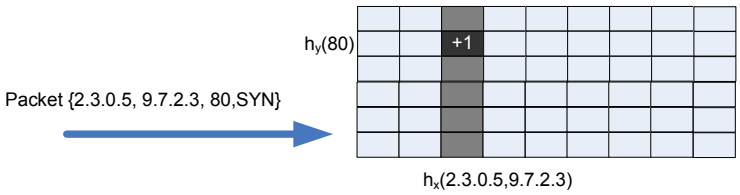


Fig. 5. An example of UPDATE operation for two-dimensional sketch

In the detection stage, when we find an attack by using reversible sketch or other methods, *i.e.*, we know the {SIP,DIP}, we can use the column of buckets in the hash matrix selected by that specific {SIP,DIP} to infer the distribution of DP and further infer which type of attack it is, e.g., a SYN flooding or a vertical scan.

There are many different ways and different metrics can be use to distinguish SYN floodings and port scans. For example, for each bucket in the hash matrix we increment it for a SYN and decrement it for a SYN/ACK. The SYN and SYN/ACK packets are presented here just as an example, but typically it could be incremented for any open parenthesis and decremented for the corresponding closed parenthesis. It is easy to figure out that the expected value of each of these buckets is 0, if the network traffic feed in is benign. Thus, if {SIP,DIP} has been proven by other methods that it contain attacks, then for a SYN flooding attack, only 1-3 buckets in the column selected by {SIP,DIP} should be heavy; on the other hand, for a vertical port scan it should increase most of buckets in the column a small positive number. It seems that we can intuitively use this difference to determine whether the attack is a SYN flooding or a vertical scan.

However, there could be some middle status to make the situation ambiguous. For instance, someone sent 150 SYN packets to the victim in five minutes, but sent to 10 different ports and each port receives 15 SYN packets. Although 150 SYN packets in five minutes without SYN/ACK will exceed the threshold of detection, it is really hard to say if it is a SYN flooding or a vertical scan, since it is not typical. The typical SYN flooding should be many SYN without SYN/ACK sent to 1-3 ports on one victim. The typical vertical scan should be many SYN without SYN/ACK sent to many ports on one victim, and on average each port only receives one SYN without SYN/ACK.

Fortunately, we test the different traces we have, and find the cases like that is very rare in real network traffic. The following Figure 6 shows the distribution of number of attacks we found above the threshold 50, and how many port they related. Figure 7 shows the distribution of number of attacks we found above the threshold 100, and how many different IP they related. Furthermore, for hackers, it's not very meaningful to send to several ports each multiple SYNs without SYN/ACK. Therefore, we can stick on the two point distribution. Because of the random variation, the behavior of SYN flooding is more obvious in two-dimensional sketch than the behavior of vertical scan, we just find the SYN flooding out, and the rest heavy changes find by reversible sketch can be regarded as vertical scan.

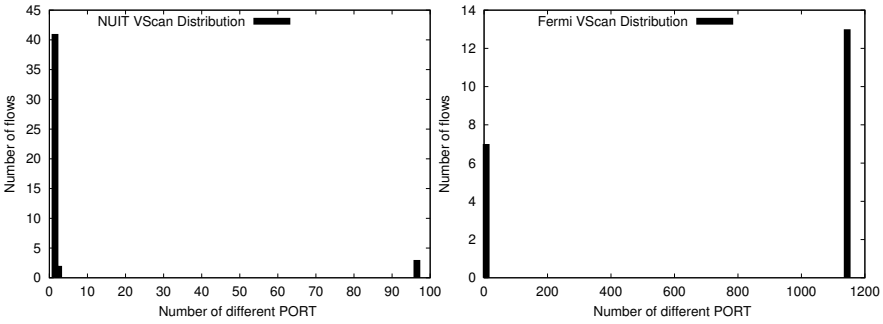


Fig. 6. The different port Distribution of {SIP, DIP}.

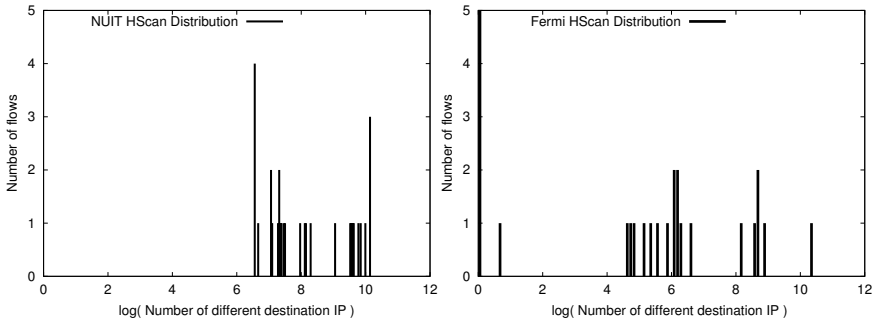


Fig. 7. The different IP Distribution of {SIP, Dport}.

Another problem we need to consider is the above argument ignores random variation. It is well known that even if the expectation of a random work after N steps is zero, the standard deviation is $O(\sqrt{N})$. Thus some false positives and false negatives could happen. Actually, we can use the similar method as PCF used in [20] to obtain the false positives and negatives.

As suggested in [20] we can use *Generalized Increment-Decrement Counter Model* to formalize this problem. If one SYN packet hash to the bucket (counter), the counter increases one, if one SYN/ACK packet hash to the counter, the counter decreases one. At the end of a sufficiently large amount of time called a *measurement interval,* (T), the counter should accumulate all the packets hashed to it together to the outcome value N. For the normal traffic, the expectation of N, $E(N)$, should equal to zero. Since the time between SYN and corresponding SYN/ACK is close to the Round Trip Time (RTT), which is normally very short (most likely will less than 500ms), for the normal traffic, the N should be almost zero. There are several reasons could make N non-zero:

1. When the SYN packet happens at the end of the *measurement interval*, its corresponding SYN/ACK could be in next interval.
2. Sometime, the SYN or SYN/ACK may be resent.
3. Occasionally, the response time from a busy server could be quite long (longer than one interval)
4. Route churn may cause the SYN or SYN/ACK to not be seen.

Observe that in each of these malformed but benign cases, the anomaly is equally likely to add an extra SYN as it is to an extra SYN/ACK to an interval. Therefore all these cases can be modeled as contributing, in the general case, either a SYN or SYN/ACK with probability 0.5. Without loss of generality, let us randomly pick up a counter. let X_i represent the random variable that represents the ith random event that happens in the interval T. For X_i, $P\{X_i = 1\} = 0.5$ and $P\{X_i = -1\} = 0.5$. Therefore, the expectation of X_i say μ_i is 0, and the standard deviation σ_i is 1. For total n random events, the final value of the counter will be $X = \sum_{i=0}^{n} X_i$. For sufficiently large value of n, the Central Limit Theorem assures that the distribution of X will approach to a Gaussian distribution. And for X we have:

$$\begin{aligned} \mu &= \mathrm{E}(X) = 0 \\ \sigma^2 &= \mathrm{VAR}(X) = n \\ Pr\left[a \leq \frac{X-\mu}{\sigma} \leq b\right] &= \frac{1}{\sqrt{2\pi}} \int_a^b e^{-z^2/2} \mathrm{d}z \end{aligned} \tag{1}$$

As we discussed before, such random events would only influence a very small portion of the SYN and SYN/ACK packets, and normally SYN and SYN/ACK packets only account for 1/3 or less of the total packets on network. So the ratio ϵ of random events to the total packets normally is very small about less than 5%. Given network speed S, measurement interval T, the total random events n for each bucket will be:

$$n = \frac{\epsilon S T}{K_x K_y} \tag{2}$$

K_x and K_y are the number of columns and rows of the two dimensional sketch, respectively.

As we discussed before, the SYN flooding should only send SYN to 1-3 ports, and vertical scan evenly distribute the SYN to many ports. In practice, we use this method to differentiate these two attacks: we first obtain the sum of the top p buckets. If it is

larger then ϕ ($\phi < 1$)times the threshold of $D = min(\text{vertical scan}, \text{SYN flooding})$, then we regard it as SYN flooding; otherwise it is vertical scan.

Because we only use the two-dimensional sketch to test the attacks which have already been detected in the reversible sketches, the false positive and negative to normal traffic will be as low as the false positive and negative of reversible sketch (actually, even lower). So we can always ignore it. Then we calculate the ratio that the vertical scans are incorrectly classified as SYN floodings, r_1 and the ratio that the SYN floodings are incorrectly classified as vertical scans, r_2.

Since Equation 1 shows the normal traffic in every bucket can be approximate as a Gaussian random variable, the sum of p buckets is still a Gaussian random variable X_p:

$$\begin{aligned} \mu_p &= 0 \\ \sigma_p^2 &= np \end{aligned} \tag{3}$$

If one vertical scan whose total SYN packets above the threshold D happens, and is misclassified as a SYN flooding, then $\frac{pD}{K_y}$ is the portion of the SYN packets on the top p buckets caused by vertical scan, the possibility of it misclassified as a SYN flooding is

$$P_1 = Pr\left[\frac{X_p}{\sqrt{np}} \geq (\phi - \frac{p}{K_y})D\right].$$

And the possibility of a SYN flooding misclassified as a vertical scan is

$$P_2 = Pr\left[\frac{X_p}{\sqrt{np}} \leq -(1-\phi)D\right].$$

This is because excluding the SYN flooding traffics which are larger than D, the sum of the top p buckets is a Gaussian random variable with $\mu = 0$ and $\sigma^2 = np$.

In a K-ary two-dimensional sketch we have H independent two-dimensional hash matrices, thus $r_1 = P_1^H$ and $r_2 = P_2^H$. If we choose $K_x = 2^{12}$, $K_y = 2^6$, $\phi = 0.7$, $D = 100$, $p = 5$, and $\epsilon = 5\%$, for 10Gb/s network, we assume the average packet size is 400 bytes [], measurement interval is 60 seconds, we can obtain that $n = 37.5$, $r_1 = 2.178 \times 10^{-17}$, and $r_2 = 2.879 \times 10^{-6}$. Therefore, we only need a few stages of Hash matrices obtain very good results.

Moreover, we can use similar approaches to differentiating the horizontal scan and the SYN flooding attacks.

3 Detailed Design

3.1 The Threat Model and Detection Algorithms

Ultimately, we want to detect as many attacks as possible. As a first step, we focus on two arguably the most popular intrusions for detection: DoS TCP flooding attack [1] and port scan (mostly for worm propagation). It is reported that more than 90% of DoS attacks are TCP SYN flooding attacks [17, 18], and we will discuss its detection in details in this paper.

[1] According to the CERT DoS threat model [31], DoS attack may also include corruption attacks, which are excluded here because they are often application/protocol specific.

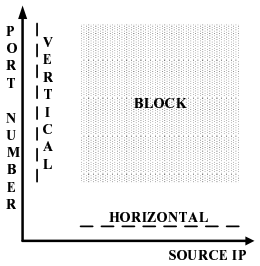


Fig. 8. Visual representation for four types of scans.

Scan is probably the most common and versatile type of intrusion. Based on source/dest IP and the port number involved in the scans, there are three well known types of scans: *horizontal scan, vertical scan,* and *block scan* [32, 33]. The classification is illustrated in Figure 8. Unlike DoS attacks, the attacker needs to use a real source IP address, since she needs to see the result of the scan in order to know what ports are actually open [32, 33]. Horizontal scans are the most common type of scan, and scans ports on IP addresses in some range of interest. The port number is often unique because it reflects the vulnerability the virus/worm or attackers try to exploit. A vertical scan is a scan of some or all ports on a single host, with the rationale that the attacker is interested in this particular host, and wishes to characterize its active services to find which exploits to attempt [33]. The third type of scan, a block scan, is a combination of horizontal and vertical scans over numerous services on numerous hosts [33]. Block scan is not very common in practice, thus we focus on the horizontal and vertical scans.

The challenge is to detect and differentiate these attacks scalably and accurately, and to get the key characteristics (*e.g.*, source IPs for scans) of such attacks for mitigation.

We denote the key of sketch as `K`, the feature value recorded in sketch as `V`, and the reversible sketch as `RS(K,V)`. Since normally we only extract fields in IP header, the possible fields we can use are shown in Table 1.

the destination IP	`DIP`
the source IP	`SIP`
the destination port	`DP`
the source port	`SP`

Table 1. The possible fields in IP header that we can use in detection

Here, we only consider the attacks in TCP protocol, in the other words, the TCP SYN-Flooding attacks and TCP port scans. Normally, the attackers can choose TCP source ports arbitrarily, so `SP` may not be a good metric for attack detection. For the other three fields, we could consider all the combinations of these three fields, but the key (`SIP`, `DIP`, `DP`) will result in a very large key space 2^{80} and it's hard to use it as the key for reverse sketches. The Table 2 shows the other combinations and their uniqueness on different types of attacks. Here, we define the uniqueness of a key as the capability of differentiating different types of attacks, which is measured by how many types of attacks the key can detect. Ideally if only one type of attack it can detect, that is best. Nevertheless, normally a key can detect several attacks, so we need use more dimensions to distinguish one type of attacks

Types of Keys	SYN flooding	horizontal scan	vertical scan	block scan
(SIP,DP)	Part*	Yes	No	Yes
(DIP,DP)	Yes	No	No	No
(SIP,DIP)	Part*	No	Yes	Yes
(SIP)	Part*	Yes	Yes	Yes
(DIP)	Yes	No	Yes	Yes
(DP)	Yes	Yes	No	Yes

Table 2. The strength of different types of keys

From Table 2 we can know, the combinations of two fields have better strength then single fields, so we use the 3 combinations of two fields as keys for the reverse sketches. Our detection has the following three steps:

Step 1, we use `RS((DIP, Dport), SYN-SYN/ACK)` to detect SYN flooding attacks because it usually targets certain service as characterized by the `Dport` on a small set of machine(s). The value of `SYN-SYN/ACK` means that for each incoming SYN packet, we will update the sketch by incrementing one, while for each outgoing SYN/ACK packet, the sketch will be updated by decrementing one. In fact, similar structure can be applied to detect any partial completion attacks [20]. The reversible sketch can further provides the victim IP and port number for mitigation as in Section **??**. We denote this set of `DIP` as $FLOODING\_DIP\_SET$.

Step 2, we use `RS((SIP, DIP), SYN-SYN/ACK)` to detect any intruder try attack to particular IP address. They can be non-spoofed SYN flooding attacks or vertical scans. For each `(SIP, DIP)` entry, if `DIP` $\in FLOODING\_DIP\_SET$, we put the `SIP` into the $FLOODING\_SIP\_SET$ for the next step; otherwise, the `(SIP, DIP)` are the attacker and victim of a vertical scans.

Step 3, we use `RS((SIP, Dport), SYN-SYN/ACK)` to detect any source IP which causes large number of uncompleted SYN connections to a particular destination port. For each `(SIP, Dport)` entry, if `SIP` $\in FLOODING\_SIP\_SET$, it is a non-spoofed SYN flooding; otherwise, it is a horizontal scan.

Attack types	RS((DIP, Dport), SYN-SYN/ACK)	RS((SIP, DIP), SYN-SYN/ACK)	RS((SIP, Dport), SYN-SYN/ACK)
SYN flooding	Yes	Yes	Yes
Vertical scans	No	Yes	No
Horizontal scans	No	No	Yes

Table 3. Different attacks are captured and detected in different reversible sketches.

Thus altogether, we need three reverse sketches to record the traffic. The classification rules are depicted in Table 3.

In addition to (SYN, SYN/ACK) pairs, other features like (SYN, FIN) pairs or (SYN/ACK, FIN) pairs can also be applied for detection as discussed in [17, 18].

3.2 Time Series Analysis based Anomaly Detection

In the HRAID system, we can apply any Time Series Analysis (TSA) or Statistical Learning Theory (SLT) approaches to fit in the building block in Figure 1. Here, as a practical example, we apply exponential moving average model (EWMA) algorithm, and Holt Winter algorithm as the forecast model to do change detection.

The following characteristics we used for detection for both algorithms is the $\#SYN-\#SYN/ACK$ counted in a certain time interval. Denote $\mathbf{M}_0(t)$ as the $\#SYN - \#SYN/ACK$ at the time interval t. And let $\mathbf{M}_f(t)$ be the forecasted $\#SYN - \#SYN/ACK$ at the time interval t. Then the difference between the forecasted value and the actual value is eventually used for detection, *i.e.*, $\mathbf{y}_t = \mathbf{M}_0(t) - \mathbf{M}_f(t)$ is the feature value used for detection.

EWMA for feature extraction EWMA is fast and accurate, and can help remove noises. An EWMA is constructed to forecast the $\#SYN - \#SYN/ACK$ at the time interval t, based on the values at the time intervals which are previous to t, i.e.,

$$\mathbf{M}_f(t) = \begin{cases} \alpha\mathbf{M}_0(t-1) - (1-\alpha)\mathbf{M}_f(t-1) & t > 2 \\ \mathbf{M}_0(1) & t = 2 \end{cases}, \tag{4}$$

Holt Winters forecasting Holt-Winters Forecasting **??** is a more sophisticated algorithm that builds upon exponential smoothing. Holt-Winters Forecasting rests on the premise that the observed time series can be decomposed into three components: a baseline, a linear trend, and a seasonal effect. The algorithm presumes each of these components evolves over time and this is accomplished by applying exponential smoothing to incrementally update the components.

The prediction is the sum of the three components: $M_f(t) = a_t + b_t + c_{t+1-m}$

The update formulas for the three components a, b, c are:

– Baseline ("intercept"):

$$a_t = \alpha(M_0(t) - c_{t-m}) + (1-\alpha)(a_{t-1} + b_{t-1}) \tag{5}$$

– Linear Trend ("slope"):

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \tag{6}$$

– Seasonal Trend:

$$c_t = \gamma(M_0(t) - a_t) + (1-\gamma)c_{t-m} \tag{7}$$

As in exponential smoothing, the updated coefficient is an average of the prediction and an estimate obtained solely from the observed value $M_0(t)$, with fractions determined by a model parameter (α, β, γ). M is the period of the seasonal cycle; so the seasonal coefficient at time t references the last computed coefficient for the same time point in the seasonal cycle.

α, β and γ are the adaptation parameters of the algorithm and $0 < \alpha, \beta, \gamma < 1$. Larger values mean the algorithm adapts faster and predictions reflect recent observations in the time series; smaller values means the algorithm adapts slower, placing more weight on the past history of the time series.

Moreover, if no seasonal factor take effect, Holt-Winters Forecasting can also be non-seasonal, which also called Holt's Linear Exponential Smoothing (LES). Then, we only need to take care α and β. The equation for forecasting are:

$$\begin{aligned} a_t &= \alpha M_0(t) + (1-\alpha)(a_{t-1} + b_{t-1}) \\ b_t &= \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \end{aligned} \tag{8}$$

3.3 Intrusion Mitigation

The HRAID system can instrument the routers or system administrators to mitigate the attacks based on the attack type and their key characteristics obtained from reversible sketches. For distributed SYN flooding, we can start the *SYN defender* [34], *SYN proxy* and/or *SYN cookies* [35] for the particular victim machines and applications to alleviate the DoS effects. Other DoS attack mitigation schemes [36] can also be applied. For port scans and point-to-point SYN flooding, we can use ingress filter to block the traffic from the attacker IP. For vertical scans, we can quarantine the victim machine for further inspection. For horizontal scans, we will pay particular attention to any suspicious internal or outgoing traffic with the same port number because if any internal machine is compromised, it will probably try to infect more machines.

4 Evaluation and Attack Validation

In this section we evaluate our schemes with *netflow* traffic traces collected from three sources:

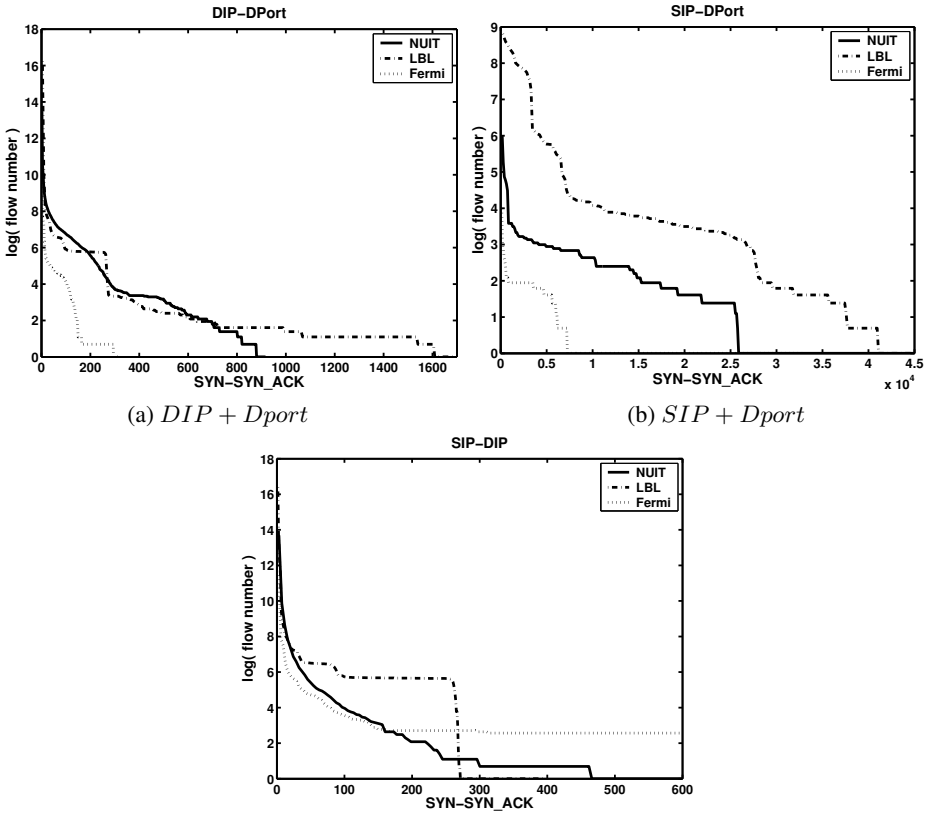
- The first trace was collected at the campus gateway router at Northwestern University (a Class B network, actually Northwesern University has several Class B networks). It consists of 25G netflow records captured in three days in March, 2005. The average packet rate is 37K/s and the peak packet rate is 79K/s. The flows were constructed from packet sampling at a 1:1 rate.
- The third trace was collected at the Fermi National Lab. The one-day trace includes about 170M netflow records. The flows were constructed from packet sampling at a 1:50 rate.
- The third trace was collected at the LBL Lab. The one-day trace consists of about 900M netflow records. The flows were constructed from packet sampling at a 1:1 rate.

We demonstrate two things in the flowing experiments. First, we present how to empirically set the thresholds for the detection system in a principled way. Second, we validate our theoretical analysis in three aspects, i.e., accuracy, memory consumption and speed. Furthermore, we show the detailed detection results of our approach on real network traces and compare them with other existing methods.

4.1 Parameter Setting

The most important parameters are the thresholds to discriminate anomaly network flows from normal ones. Generally, a good threshold should balance between alert number and sensitivity. Here we provide some empirical guidance to choose the threshold values. Note, in the following discussion if we do not specify the length of the time interval we used, the default value is 5 minutes, so all the numbers are shown here is for 5 minutes interval.

Based on Netflow data from NUIT, Fermi Lab and LBL, we plotted curves of the aggregated flow number versus the number of `SYN-SYN/ACK` as Figure 9, where the



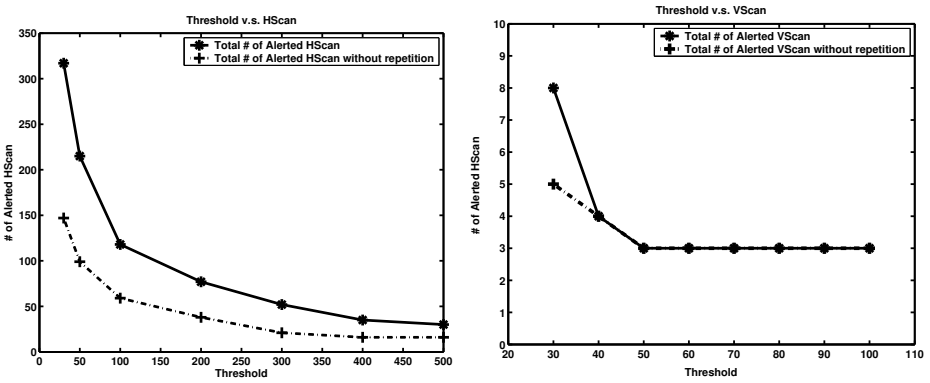
(a) $DIP + Dport$ (b) $SIP + Dport$

(c) $SIP + DIP$

Fig. 9. log of flow numbers v.s. $SYN - SYN/ACK$

flows are aggregated on `(DIP, Dport)`, `(SIP, Dport)` and `(SIP, DIP)` respectively. Note that the vertical axes indicate the aggregated flow number in log scale which have `SYN-SYN/ACK` larger than or equal to what was indicated by the horizontal axes. For example, in Figure 9(a), the curve value at `SYN-SYN/ACK`=100 represents the aggregated number of flows which have a `SYN-SYN/ACK`≥ 100.

Our empirical studies show that the knee points of these curves are good choices of the threshold values for administrators. The knee point refers to the critical point where a monotonic decreasing/increasing curve transit from dramatic changes to small changes. The intuition behind choosing the knee points of the aggregated flow curves as the threshold lies on the fact that normal network flows usually have small `SYN-SYN/ACK` values which can be well modeled by a Gaussian distribution with a small variance, while the anomaly flows usually have large numbers of `SYN-SYN/ACK` which are almost uniformly distributed in a large range. Our observation is that the flow data whose `SYN-SYN/ACK` are larger than the knee point usually indicate more serious attacks, and administrators should focus on those flows. For example, in Figure 9(a) we can see the knee point of the aggregated network flow on `(DIP, Dport)` for NUIT curve



(a) Hscan (b) Vscan

Fig. 10. The effect of different thresholds.

and LBL curve is around 300 (in 5 minutes interval), and for Fermi curve is about 100 (in 5 minutes interval). In this way we give administrators more reasonable threshold range. Table 4 shows the recommending thresholds for all the data traces obtained by the knee points in their curves.

Traces	SYN flooding	Hscan	Vscan
NUIT	300	500	100
CERNET	0	0	0
LBL	300	500	100
Fermi	100	500	100

Table 4. Threshold table.

Another observation is that in a network of reasonable large scale, those aggregated network flows which have small number `SYN-SYN/ACK` can usually be ignored, since they will not affect the network too much anyway. So system administrators can pay more attention to those more serious attacks. Certainly administrators can adjust threshold values based on their requirements and experiences. Figure 10 shows the change of the alerted attack number affected by choosing different thresholds using two-hour NUIT data.

4.2 Validation Metrics

Our metrics include *accuracy* (in terms of the real positive /false positive percentages), *execution speed*, and *the amount of memory access per packet*, as well as *the amount of memory used in recording stage*.

Accuracy As we all know, it is almost impossible to obtain the absolute ground truth of attacks from the real traces. In order to verify the accuracy of the results, we evaluate

it by two means. On one hand, we evaluate the error introduced by sketches. Here we implemented a similar algorithm but without sketches (we call it non-sketch method), so that program can use memory as much as it needs, and then use the attacks detected by it as the ground truth. On the other hand, we further compare our approach with several other the state-of-the-art network intrusion detection approaches, and check them by limited information we obtained, such as honeypot hosts.

Firstly, we compared the results of our approach with those of non-sketch method on all the four data sources. We used a measurement interval of five minutes. For the number of buckets used in the sketch, we chose two set of parameters according to the different memory requirements. The small one used 2^{12} buckets for 48 bits reverse sketch and 2^{14} buckets for its corresponding original verification sketch, 2^{16} buckets for 64 bits reverse sketch and 2^{14} buckets for its corresponding original verification sketch, and $2^{12} \times 64$ buckets for the two-dimensional sketch. The large one used 2^{18} buckets for 48bits reverse sketch and 2^{16} buckets for its corresponding original verification sketch, 2^{16} buckets for 64 bits reverse sketch and 2^{16} buckets for its corresponding original verification sketch, and $2^{14} \times 64$ buckets for the two-dimensional sketch. Table 5 and Table 6 show the comparison results using EWMA and the nonseasonal Holt-Winter method as the forecasting method, respectively. We chose threshold for these results in Table 4 according to knee point discussed in previous section.

Data traces	Attack type	Small memory				Large memory			
		Non-Sketch	Sketch	FP	FN	Non-Sketch	Sketch	FP	FN
NUIT	SYN Flooding	11	11	0	0	11	11	0	0
	Horizontal scan	30	30	0	0	30	30	0	0
	Vertical scan	3	3	0	0	3	3	0	0
CERNET	SYN Flooding	0	0	0	0	0	0	0	0
	Horizontal scan	0	0	0	0	0	0	0	0
	Vertical scan	0	0	0	0	0	0	0	0
LBL	SYN Flooding	8	8	0	0	8	8	0	0
	Horizontal scan	2171	2171	0	0	2171	2171	0	0
	Vertical scan	0	0	0	0	0	0	0	0
Fermi	SYN Flooding	8	8	0	0	8	8	0	0
	Horizontal scan	10	10	0	0	10	10	0	0
	Vertical scan	13	13	0	0	13	13	0	0

Table 5. Comparison results of non-sketch method and our approach forecasting by EWMA.

As we can observe in these tables, with either EWMA or Holt-Winter as the forecast method, the detection results of our approach are the same as non-sketch method. In other words, the bias introduced by our sketches can almost be ignored in real experiments. These results demonstrate that the proposed approach does achieve high accuracy as well as scalable capability. Note in order to compare the accuracy between non-sketch method and our sketch method more strictly, listed attacks have not been filtered, i.e. there are some repeated alerts (the same attack but reported several times in different time interval), although change detection has already reduced a lot.

Secondly, we applied the HRAID to various network flow data. To the best of our knowledge, the HRAID is the first such kind of IDS that can not only detect SYN

Data traces	Attack type	Small memory				Large memory			
		Non-Sketch	Sketch	FP	FN	Non-Sketch	Sketch	FP	FN
NUIT	SYN Flooding	11	11	0	0	11	11	0	0
	Horizontal scan	26	26	0	0	26	26	0	0
	Vertical scan	3	3	0	0	3	3	0	0
CERNET	SYN Flooding	0	0	0	0	0	0	0	0
	Horizontal scan	0	0	0	0	0	0	0	0
	Vertical scan	0	0	0	0	0	0	0	0
LBL	SYN Flooding	9	9	0	0	9	9	0	0
	Horizontal scan	2485	2485	0	0	2485	2485	0	0
	Vertical scan	0	0	0	0	0	0	0	0
Fermi	SYN Flooding	8	8	0	0	8	8	0	0
	Horizontal scan	10	10	0	0	10	10	0	0
	Vertical scan	13	13	0	0	13	13	0	0

Table 6. Comparison results of non-sketch method and our approach forecasting by Holt-Winter.

flooding, worm (horizontal scan) and vertical scan at the same time, but also provide detailed information of the detected attacks, such as victim IP, victim port, and attack source IP. Using four data sources, we show the statistical attack table according to explicit detected attack list in Table 7, where unlike Table 5 and Table 6, we filter some redundant attacks and count them as just one, thus some statistical values are not same as previous ones. Here the parameters we chose are the same as Table 4.

Data traces	Attack type	EWMA			Holt-Winter		
		DIP	SIP	Dport	DIP	SIP	Dport
NUIT	SYN flooding	7	\	7	7	\	7
	Horizontal scan	\	12	11	\	11	7
	Vertical scan	3	1	\	3	1	\
CERNET	SYN flooding	0	0	0	0	0	0
	Horizontal scan	0	0	0	0	0	0
	Vertical scan	0	0	0	0	0	0
LBL	SYN flooding	3	\	2	3	\	2
	Horizontal scan	551	555	551	555	551	555
	Vertical scan	0	0	0	0	0	0
Fermi	SYN flooding	7	\	4	7	\	4
	Horizontal scan	\	10	6	\	10	6
	Vertical scan	13	1	\	13	1	\

Table 7. Experiment results

We further compare our approach with several other related network intrusion detection approaches: the *Threshold Random Walk (TRW)* by Jaeyeon Jung for port scan detection, *Change-Point Monitoring (CPM)* by Haining Wang for SYN flooding detection, and *Backscatter (BS)* for spoof DoS detection.

Above all, Table 8 shows the functionality comparison to our approach and all mentioned approaches above from a high-level angle. Note that TRW only focus on horizontal scan, and Backscatter only focus on spoofed DoS detection. Although CPM can

detect both spoofed and non-spoofed DoS attack, it can cause false positives by mixed port scans. Compared with them, our approach realizes all these detect goals, and in the following part we further verify that our approach reaches these goals without losing accuracy.

Approaches	Spoofed DoS	Non-spoofed DoS	Horizontal Scan	Vertical Scan
HRAID	Yes	Yes	Yes	Yes
TRW	No	No	Yes	No
CPM	Yes*	Yes*	No	No
BS	Yes	No	No	No

Table 8. Functionality comparison of four approaches.

Table 9 shows the comparison results of our methods with TRW on horizontal scans. The numbers of detected horizontal scans by our sketch based approach and native infinite memory edition have been aggregated by the source IP in order to compare the result from TRW, in which the horizontal scans of same source IP but different ports are counted only by the source IP. From this table, we can observe that almost all the detected scans are the same for both algorithms expect some special cases. Two factors cause these exceptional cases. One is some attacks have both successful connection and unsuccessful connection. The main idea for TRW is to give them different weight and multiply them together in the process of detection. If the product value is higher than a certain threshold, it will be identified as port scan. In some cases, although the number of unsuccessful connections is fairly large, which is more than 500, they still have several successful connections, such as 50 to 100. Due to the disturbance of these successful connections, TRW is not sensitive to this kind of abnormal behaviors. In fact, we can consider these behaviors as abnormal ones, at least suspicious ones. The other factor is some of horizontal scans aim at different destination ports of a certain range of hosts, and the scan number on one destination port is relative small (less than our threshold), that is, this kind of behavior will not cause serious damage to network, so in our approach we just ignore those behaviors. But if the serious level of these behaviors increases, and the scan number will reach our threshold, we can detect them by then.

traces	TRW	HRAID (EWMA)	overlay number	HRAID (HW)	overlay number
NUIT	8	12	8	11	8
CERNET	0	0	0	0	0
LBL	555	551	547	555	551
Fermi	10	10	10	10	10

Table 9. Comparison hscan results of our method and TRW aggregated by source IP.

Next, we compared our method with Change-Point Monitoring for SYN flooding attacks. The results are showed in Table 10. It is weird that the results of these two

approaches have no intersection. The main reason is that CPM only uses normalized metrics for (SYN, SYN/ACK) pair and (SYN, FIN) pair, and it can not distinguish scans from SYN flooding, since both of them can cause the great difference between these two pairs. Unfortunately there are a lot of horizontal scans in LBL data, and it is distributed in every interval, so CPM reports SYN flooding attack for every interval as we expected. This case shows that CPM may cause high false positive in real network environment, which includes both SYN flooding attacks and port scans. Except this, in some intervals only one or two SYN flooding attacks exists, but normal flows account for the majority stream, thus abnormal behaviors will be flooded by large amount of normal behaviors. This may cause the false negative of CPM, e.g. in NUIT traces.

traces	CPM	HRAID(EWMA)	HRAID(HW)	overlay number
NUIT	0	6	6	0
CERNET	0	0	0	0
LBL	288	0	0	0
Fermi	15	8	8	0

Table 10. Comparison synflooding results of our method and CPM.

Speed Speed is another important factor for high-speed network intrusion detection, especially recording speed. We can adopt both hardware and software to implement the HRAID. For hardware implementation, we can do full parallel updating on several reverse sketches and two-dimensional sketches. Thus the total processing time is to update one reverse sketch. In our experiments it is about ?. For software implementation, we have to update all the sketches sequentially. Among them, the reverse sketches account for most of the computation. So we can almost ignore the running time of the two-dimensional sketches. In our experiments it is about ?.

Memory consumption It is very important to have small memory consumption for online traffic recording over high-speed links, since that is much easier and more effective when implementing in hardware. According to different level detection requirement, we can choose either small number buckets hash or large number buckets hash. For the best case, we only need 9Mbytes memory to record ll the coming flows. Table 11 shows the comparison results.

5 Related Work

5.1 Data Stream Computation

The ever-increasing link speeds and traffic volumes of the Internet make monitoring and analyzing network traffic a challenging but essential service for managing large ISPs. This service is especially important for anomaly/intrusion detection. There are

Methods(Bytes)	1sec	1min	5min	1sec	1min	5min
Our method with small sketch	9M	9M	9M	9M	9M	9M
Our method with large sketch	42.9M	42.9M	42.9M	42.9M	42.9M	42.9M
Method without sketch	93.75M	5.63G	28G	375M	22.5G	112.5G
TRW	31.3M	1.88G	9.38G	125M	7.5G	37.5G

Table 11. Memory comparison.

two key primitives in the analysis of a live network traffic stream: heavy hitter detection and heavy change detection. The former finds flows that constitute more than a given threshold fraction of the total traffic stream. The latter detects flows whose size changes significantly from one period to another. There is a significant amount of prior work on efficient and online heavy hitter detection [27, 37–39].

Efficient online heavy *change* detection, however, remains a challenging problem of significant interest because it is more general and powerful than heavy hitter detection. "Change" is a concept that ranges over a gamut from simple absolute or relative changes, to variational changes and the linear transformation changes for use with various time-series forecasting models [24, 40–43]. K-ary sketch [44] is the first heavy change detection approach, but it is not reversible, which means we cannot know the "key" inside the culprit flows. After that we proposed reversible sketch [23, 24] which targets this problem. Deltoids [45] is another approach which can get the "key" inside the culprit flows based on group testing.

PCF is recently proposed for scalable network detection [20]. It uses a similar data structures as an original sketch, thus not reversible. So even when attacks are detected, attacker or victim information is still unknown, making the mitigation impossible. Furthermore, they examine each individual time interval for detection, while we check the statistical behavior change point for detection which can detect the attacks more accurately and give concise report.

Some other existing high-speed network monitoring systems estimate the flow-level traffic through packet sampling [46–49], but this has two shortcomings. First, sampling is still not scalable, especially after aggregation; there are up to 2^{64} flows even defined only by source and destination IP addresses. Second, long-lived traffic flows, increasingly prevalent for peer-to-peer applications, will be split up if the time between sampled packets exceeds the flow timeout [46].

5.2 Network Intrusion Detection

Many network IDSs like Bro [11] and Snort [12] check packet payload for virus/worm signatures. However, such schemes are not scalable for high-speed network links. Recently, there are some works proposed to detect large scale attacks, like DoS attacks, port scans, *etc.*based on the statistical traffic patterns. They can roughly be classified into two categories: 1) detecting based on the overall traffic [16–19], thus tend to be inaccurate or cannot find real attack flows; and 2) flow level detection [11, 12, 21], but they are not scalable and thus themselves are vulnerable to attacks. In the first category, for instance, the-state-of-art stateless router-based SYN flooding detection techniques,

CPD, use the statistical behavior of SYN-FIN, or SYN-SYN/ACK packet pairs based on overall traffic for detection [17, 18]. However, it will suffer high false positives when there is port scan traffic, because port scan also can contribute lots of SYN without SYN/ACK or FIN. In the second category, For example, the state-of-the-art work using Threshold Random Walk for port scan detection can accurately and fast detect port scans [21], but it is very vulnerable to DoS attacks with randomly spoofed IP addresses, or worm breakout, which will cause it use too much memory to store the status of each source IP addresses.

6 Conclusion

Conclusion we get

7 Acknowledgement

Thanks to everyone

References

1. Mars, E., Jansky, J.D.: Email defense industry statistics. `http://www.mxlogic.com/PDFs/IndustryStats.2.28.04.pdf` (2004)
2. Axelsson, S.: Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Department of Computer Engineering, Chalmers University of Technology, Sweden (2000)
3. Alessandri, D., et al.: Malicious and accidental fault tolerance for Internet applications (MAFTIA): Towards a taxonomy of intrusion detection systems and attacks. Technical Report Research Report RZ 3366, IBM Research (2001)
4. Allen, J., Christie, A., Fithen, W., McHugh, J., Pickel, J., Stoner, E.: State of the practice of intrusion detection technologies. Technical Report CMU/SEI-99-TR-028, Carnegie Mellon University (1999)
5. Bace, R., Mell, P.: Intrusion detection systems. Technical Report SP 800-31, National Institute of Standards and Technology (2001)
6. Ryutov, T., Neuman, C., Kim, D., Zhou, L.: Integrated access control and intrusion detection for web servers. IEEE Trans. on Parallel and Distributed Systems **14** (2003) 841–850
7. Loyall, J.P., Pal, P.P., Schantz, R.E., Webber, F.: Building adaptive and agile applications using intrusion detection and response. In: Proc. of the Network and Distributed Systems Security Conference. (2000)
8. Hofmeyr, S., Forrest, S.: Intrusion detection using sequences of system calls. Journal of Computer Security **6** (1998)
9. Wagner, D., Dean, D.: Intrusion detection via static analysis. In: Proceedings of the IEEE Symposium on Security and Privacy. (2001)
10. Sekar, R., Bendre, M., Dhurjati, D., Bollineni, P.: A fast automaton-based method for detecting anomalous program behaviors. In: Proc. of the IEEE Symposium on Security and Privacy. (2001)
11. Paxson, V.: Bro: A system for detecting network intruders in real-time. Computer Networks **31** (1999) 2435–2463
12. Roesch, M.: Snort: The lightweight network intrusion detection system (2001) `http://www.snort.org/`.

13. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: The spread of the Sapphire/Slammer worm. http://www.caida.org (2003)
14. Staniford, S., Paxson, V., Weaver, N.: How to own the Internet in your spare time. In: Proceedings of the 11th USENIX Security Symposium. (2002)
15. Weaver, N., Paxson, V., Staniford, S., Cunningham, R.: Large scale malicious code: A research agenda. Technical Report DARPA-sponsored report, DARPA (2003)
16. Moore, D., Voelker, G., Savage, S.: Inferring Internet denial of service activity. In: Proceedings of the 2001 USENIX Security Symposium. (2001)
17. Wang, H., Zhang, D., Shin, K.G.: Detecting SYN flooding attacks. In: Proc. of IEEE INFOCOM. (2002)
18. Wang, H., Zhang, D., Shin, K.G.: Change-point monitoring for detection of DoS attacks. IEEE Transactions on Dependable and Secure Computing **1** (2004)
19. Barford, P., Kline, J., Plonka, D., Ron, A.: A signal analysis of network traffic anomalies. In: ACM SIGCOMM Internet Measurement Workshop. (2002)
20. Kompella, R.R., Singh, S., Varghese, G.: On scalable attack detection in the network. In: Proc. of ACM/USENIX IMC. (2004)
21. Jung, J., Paxson, V., Berger, A.W., Balakrishnan, H.: Fast portscan detection using sequential hypothesis testing. In: Proceedings of the IEEE Symposium on Security and Privacy. (2004)
22. Schweller, R., Gupta, A., Parsons, E., Chen, Y.: Reversible sketches for efficient and accurate change detection over network data streams. In: ACM SIGCOMM Internet Measurement Conference (IMC). (2004)
23. Schweller, R., Gupta, A., Parsons, E., Chen, Y.: Reverse hashing for sketch-based one-pass change detection for high-speed networks. Technical Report 2003-31, Northwestern University (2004)
24. Krishnamurthy, B., Sen, S., Zhang, Y., Chen, Y.: Sketch-based change detection: Methods, evaluation, and applications. In: Proc. of ACM SIGCOMM Internet Measurement Conference (IMC). (2003)
25. Cormode, G., Muthukrishnan, S.: Improved data stream summaries: The count-min sketch and its applications. Technical Report 2003-20, DIMACS (2003)
26. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. J. Comput. Syst. Sci. **31** (1985) 182–209
27. Cormode, G., Korn, F., Muthukrishnan, S., Johnson, T., Spatscheck, O., , Srivastava, D.: Holistic UDAFs at streaming speeds. In: Proceedings of ACM SIGMOD. (2004)
28. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.J.: QuickSAND: Quick summary and analysis of network data. Technical Report 2001-43, DIMACS (2001)
29. Mirkovic, J., Reiher, P.: Data streams: algorithms and applications. In: Proc. of ACM SODA. (2003)
30. Sikka, S., Varghese, G.: Memory-efficient state lookups with fast updates. In: Proc. of ACM SIGCOMM. (2000)
31. CERT Coordination Center: Denial of service attacks. `http://www.cert.org/tech_tips/denial_of_service.html` (1999)
32. Yegneswaran, V., Barford, P., Ullrich, J.: Internet intrusions: Global characteristics and prevalence. In: Proceedings of ACM SIGMETRICS. (2003)
33. Staniford, S., Hoagland, J.A., McAlerney, J.M.: Practical automated detection of stealthy portscans. Journal of Computer Security **10** (2002)
34. Checkpoint Software Technologies: TCP Flooding Attack and Firewall-1 SYNDefender (2000)
35. Bernstein, D.: Syn cookies. http://cr.yp.to/syncookies.html (1996)
36. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attacks and defense mechanisms. In: ACM CCR. (2004)

37. Estan, C., Varghese, G.: New directions in traffic measurement and accounting. In: Proc. of ACM SIGCOMM. (2002)
38. Manku, G.S., Motwani, R.: Approximate frequency counts over data streams. In: Proc. of IEEE VLDB. (2002)
39. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Finding hierarchical heavy hitters in data streams. In: VLDB 2003. (2003)
40. Tsay, R.S.: Outliers, level shifts, and variance changes time series. Journal of Forecasting **7** (1988) 1–20
41. Tsay, R.S.: Time series model specification in the presence outliers. Journal of the American Statistical Association **81** (1986) 132141
42. Chen, C., Liu, L.M.: Joint estimation of model parameters and outlier effects in time series. Journal of the American Statistical Association **88** (1993) 284297
43. Chen, C., Liu, L.: Forecasting time series with outliers. Journal of Forecasting **12** (1993) 1335
44. Krishnamurthy, B., Sen, S., Zhang, Y., Chen, Y.: Sketch-based change detection: Methods, evaluation, and applications. In: Proceedings of the 2003 ACM SIGCOMM Conference on Internet Measurement, ACM Press (2003)
45. Cormode, G., Muthukrishnan, S.: What's new: Finding significant differences in network data streams. In: Proc. of IEEE Infocom. (2004)
46. Duffield, N., Lund, C., Thorup, M.: Properties and prediction of flow statistics from sampled packet streams. In: Proc. of ACM SIGCOMM Internet Measurement Workshop (IMW). (2002)
47. Duffield, N.G., Lund, C.: Predicting resource usage and estimation accuracy in an IP flow measurement collection infrastructure. In: Proc. of ACM SIGCOMM Internet Measurement Conference. (2003)
48. Duffield, N., Lund, C., Thorup, M.: Estimating flow distributions from sampled flow statistics. In: Proc. of ACM SIGCOMM. (2003)
49. Duffield, N., Lund, C., Thorup, M.: Flow sampling under hard resource constraints. In: Proc. of ACM SIGMETRICS. (2004)