A DoS Resilient Flow-level Intrusion Detection Approach for High-speed Networks

Yan Gao, Zhichun Li, Yan Chen
Department of EECS, Northwestern University
{ygao, lizc, ychen}@cs.northwestern.edu

Abstract

Global-scale attacks like viruses and worms are increasing in frequency, severity and sophistication, making it critical to detect outbursts at routers/gateways instead of end hosts. In this paper we leverage data streaming techniques such as the reversible sketch to obtain HiFIND, a High-speed Flow-level Intrusion Detection system. In contrast to existing intrusion detection systems, HiFIND 1) is scalable to flow-level detection on high-speed networks; 2) is DoS resilient; 3) can distinguish SYN flooding and various port scans (mostly for worm propagation) for effective mitigation; 4) enables aggregate detection over multiple routers/gateways; and 5) separates anomalies to limit false positives in detection. Both theoretical analysis and evaluation with several router traces show that HiFIND achieves these properties. To the best of our knowledge, HiFIND is the first *online DoS resilient flow-level intrusion detection system for high-speed networks (approximately 10s of Gigabit/second), even for the worst case traffic of 40-byte-packet streams with each packet forming a flow.*

Key words: *high-speed networking, intrusion detection, statistical detection, data streaming*

1 Introduction

Identifying Traffic anomalies and attacks rapidly and accurately is critical for operators of large networks. With the rapid growth of network bandwidth and fast emergence of new attacks/viruses/worms, existing network intrusion detection systems (IDS) are insufficient due to lack of the following features.

First, scalability to high-speed networks. Today's fast propagating viruses/worms (*e.g.*, SQL Slammer worm) can infect most vulnerable machines within ten minutes [10]. Thus, it is crucial to identify such outbreaks in their early phases, which is achievable only in high speed routers. However, existing schemes are not scalable to the link speeds and number of flows for high-speed networks.

It is in general difficult for software-based data recording approaches in IDSes to keep up with the link speed in a high-speed router. Thus, the data recording of high-speed IDSes has to be hardware implementable, and it is strongly desirable to achieve the following three capabilities: 1) small memory usage; 2) sparse memory accesses per packet [3]; and 3) scalability to large key size.

Second, attack resiliency. To bypass an IDS, attackers can execute denial-of-service (DoS) attacks, or fool the IDS to raise many false positives to conceal the real attack. Thus, the attack resiliency of an IDS is very important. However, existing IDSes often keep per-flow states for detection, which is vulnerable.

Third, attack root cause analysis for mitigation. Accurate attack mitigation requires IDSes to pinpoint the attack type and flows. This advocates to detect intrusions at the *flow level* instead of the overall traffic. Furthermore, we want to differentiate different types of attacks to choose different mitigation schemes accordingly.

Fourth, aggregated detection over multiple vantage points. Most existing network IDSes assume detection be on a single router or gateway. However, as multihoming, load balancing based routing, and policy routing become prevalent, even for a connection between a certain source and destination, the packets may traverse different paths [2]. Thus, observation from a single vantage point is often incomplete and affects detection accuracy. Meanwhile, it is very hard to copy all traffic from one router to other routers/IDSes due to the huge data volume.

Fifth, separating anomalies from intrusions for false positive reduction. To detect unknown attacks and polymorphic worms, statistics-based instead of signature-based intrusion detections have been adopted widely. However, many network element faults, *e.g.*, router misconfigurations and polluted DNS entries, can lead to traffic anomalies that may be detected as attacks.

To meet the requirements above, we propose a new paradigm called DoS resilient High-speed Flow-level INtrusion Detection, HiFIND, leveraging recent work on data streaming computation and in particular, sketches [12]. Essentially, we want to detect as many attacks as possible. As the first step towards this ambitious goal, we aim to detect various port scans (which covers most large-scale worm propagation) and TCP SYN flooding.

While each of these attacks seems relatively easy to be detected, it is indeed very hard to detect a mixture

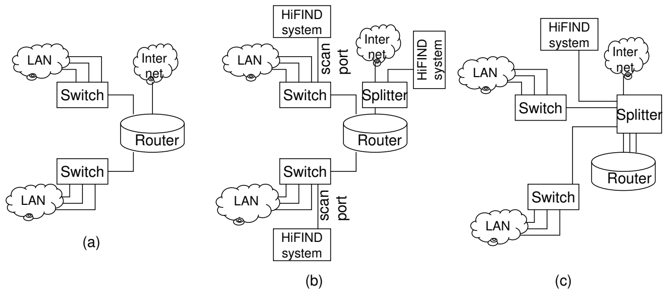

Figure 1. Attaching the HiFIND systems to high-speed routers. (a) original configuration, (b) distributed configuration for which each port is monitored separately, (c) aggregate configuration for which a splitter is used to aggregate the traffic from all the ports.

of attacks online at the flow-level. To the best of our knowledge, HiFIND is the *first* DoS resilient high-speed flow-level IDS for port scans and TCP SYN flooding for high-speed networks.

To this end, we leverage and improve sketches to record flow-level traffic as the basis for statistical intrusion detection. Firstly proposed in [11, 12], sketches have not been applied to building IDSes for the following challenges:

- Sketches can only record certain aggregated metrics for some given keys. Since it is not feasible to try all possible combinations of the metrics, what would be the minimal set of metrics for monitoring?
- Existing sketches are all one dimensional. However, various forms of attacks are often hard to identify with such single dimensional information.

In this paper, we address these two challenges and build the HiFIND prototype system to meet the aforementioned five requirements. We make the following contributions:

- We analyze the attributes in TCP/IP headers and select an optimal small set of metrics for flow-level sketch-based traffic monitoring and intrusion detection. Based on that, we build the HiFIND online high-speed flow-level IDS prototype which is DoS resilient.
- To analyze the attack root cause for mitigation, we design efficient two-dimensional (2D) sketches to distinguish different types of attacks.
- We aggregate the compact sketches from multiple vantage points (*e.g.*, routers) to detect intrusion in the face of asymmetric routing and multi-path routing caused by per-packet load balancing of routers.
- For false positive reduction, we propose several heuristics to separate SYN floodings from network/server congestions and misconfigurations.

As shown in Figure 1, HiFIND detection systems can be implemented as black boxes attached to high-speed routers (edge or backbone routers) of ISPs without affecting the normal operation of the routers.

For evaluation, we tested the router traffic traces collected at Northwestern University (NU) and Lawrence Berkeley National Labs (LBL). We validate the SYN flooding and port scans detected, and find the HiFIND system is highly accurate. The 2D sketches successfully separate the SYN flooding from port scans, and the heuristics effectively reduce false positives of SYN flooding. Our approach is also very fast in terms of data recording and detection.

Approaches	Spoofed DoS	Non-spoofed DoS	HScan	VScan
HiFIND	Yes	Yes	Yes	Yes
TRW(AC)	No	No	Yes	(Yes)
CPM	Yes, but with high FP with port scans		No	No
Backscatter	Yes	No	No	No
Superspreader	No	No	Yes	No

Table 1. Functionality comparison.

2 Related Work

2.1 Intrution Detection Systems

Some vendors claim to have multi-gigabit statistical IDSes [1], they usually refer to *average* traffic conditions and use packet sampling [4]. Recent work has proposed detecting large scale attacks, like DoS attacks, port scans, *etc.*, based on the statistical traffic patterns. They can roughly be classified into two categories: Detecting based on the overall traffic [9, 15] and flow level detection [6].

With the first approach, even when we can detect the attack, we still do not have any flow or port knowledge for mitigation. Moreover, attacks can be easily buried in the background network traffic. Thus, such detection schemes tend to be inaccurate; for example, CPM [15] will detect port scans as SYN floodings as verified in Section 5. For the second approach, such schemes usually need to maintain a per-flow table (*e.g.*, a per-source-IP table for TRW [6]) for detection, which is not scalable and thus provides a vulnerability to DoS attacks with randomly spoofed IP addresses, especially on high-speed networks. TRW was recently improved by limiting its memory consumption with approximate caches (TRW-AC) [16]. However, spoofed DoS attacks will still cause collisions in TRW-AC, and leave the real port scans undetected[1].

The existing schemes can detect specific types of attacks, but will perform poorly when facing a mixture of attacks as in the real world. People may attempt to combine TRW-AC and CPM to detect both scans and SYN flooding attacks. However, each of these two approaches can work properly only when the other one works well.

Table 1 shows the high-level functionality comparison

[1] As the authors mentioned in [16], when the connection cache size of 1 million entries reaches about 20% full, each new scan attempt has a 20% chance of not being recorded because it aliases with an already-established connection. Actually, during spoofed DoS attacks, such collisions can become even worse.

Functions	Descriptions	k-ary sketch	Reversible sketch
UPDATE(S, y, v)	Update the corresponding values of the given key into the sketch in the monitoring module	√	√
v = ESTIMATE (S, y)	Reconstruct the signal series for statistical detection for a given key in the anomaly detection module	√	√
S = COMBINE (c_1, S_1,..., c_k, S_k)	Compute the linear combination of multiple sketches $S = \sum_{k=1}^{l} c_k {\cdot} S_k$ (c_i is coefficient.) to aggregate signals in the anomaly detection module	√	√
Y = INFERENCE (S, t)	Return the keys whose values are larger than the threshold in the anomaly detection module		√

Table 2. Function of sketches (S-Sketch, v-Value, y-Key, Y-Set of keys, t-Threshold).

of our approach to the other methods. Backscatter detects the spoofed SYN flooding attacks by testing the uniform distribution of destination IPs to which the same source (potential victim) sends SYN/ACK [9]. We use this for validating the SYN flooding detected by HiFIND. Venkataraman *et al.*propose efficient algorithms to detect superspreaders, sources that connect to a large number of distinct destinations [13]. But they may have high false positives with P2P traffic where a single host may connect to many peers for download. PCF was recently proposed for scalable network detection [7]. They do not differentiate among various attacks.

2.2 Sketches for Network Monitoring

There is a significant amount of prior work on efficient and online heavy hitter detection [3, 17]. However, these approaches are limited in their applicability to online intrusion detection in that 1) they lack the ability to differentiate different types of attacks; 2) they cannot work with Time Series Analysis based detection algorithms; and 3) they cannot be applied to asymmetric routing environments.

To this end, we designed the original k-ary sketchs [8], and further enhanced them to be reversible sketches [11, 12], which allow us to have separate stages for update, combine and inference, so that we can easily solve the problems mentioned before. In Table 2, we summarize the functions supported by sketches.

3 Architecture of the HiFIND System

3.1 System Architecture

Figure 2 shows the architecture of the HiFIND system. First, we record the network traffic with sketches in each router. Based on linearity of the sketches, we summarize the sketches over multiple routers into an aggregate sketch, and apply time series analysis methods for aggregate sketches to obtain the forecast sketches for change detection. The forecast time series analysis method, *e.g.*, EWMA (exponentially weighted moving average), can help remove noise. 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 aggregate the 2D sketches in the same way and adopt them to further distinguish different types of attacks. We also apply other false positive reduction techniques as discussed in Section 3.4. Finally, we use the key characteristics of the culprit flows revealed by the reversible sketches to mitigate the attacks. Note that the streaming data recording process needs to be done continuously in real-time, while the detection process can be run in the background executing only once every interval (*e.g.*, every second or minute) with more memory (DRAM).

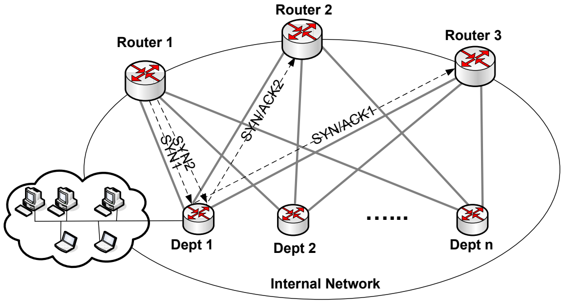

Figure 3. Sample network topology with asymmetric routing and multi-path routing.

To deal with asymmetric routing (in Figure 3), for most existing IDS systems, all the packet traces or all connection states have to be transported from one router to the other. Obviously this is very expensive. Moreover if the link is congested when an attack happens, transmission of this data can be very slow. Furthermore, some routers may use per-packet load balancing so that packets of the same flow may traverse different paths.

In contrast, for HiFIND, we summarize the traffic information with compact sketches at each edge router, and deliver them quickly to some central site. Then, with the linearity of the sketches, we can aggregate them and the resulting sketch has all the information as if all the traffics went through the same router.

3.2 The Threat Model

Ultimately, we want to detect as many different types of attacks as possible. As a first step, we focus on detecting the two most popular intrusions: TCP SYN flooding (DoS) attack and port scans/worm propagation, which include *horizontal scan (Hscan), vertical scan (Vscan),*

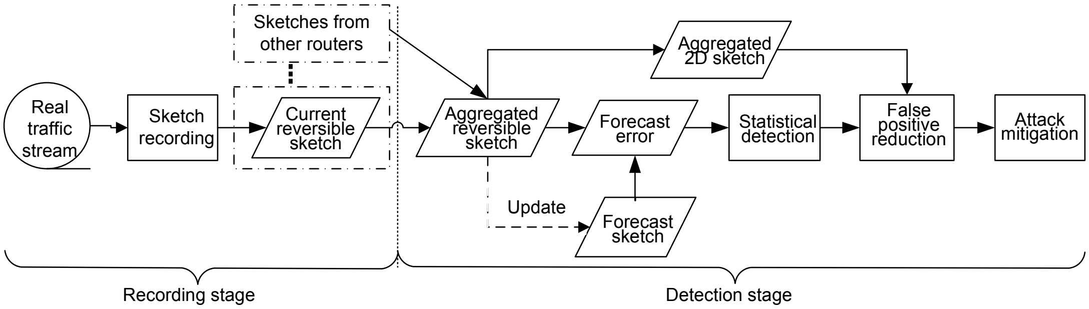

Figure 2. HiFIND System architecture.

Keys	SYN flooding	Hscan	Vscan	uniqueness
{`SIP`,`Dport`}	non-spoofed	Yes	No	1.5
{`DIP`,`Dport`}	Yes	No	No	1
{`SIP`,`DIP`}	non-spoofed	No	Yes	1.5
{`SIP`}	non-spoofed	Yes	Yes	2.5
{`DIP`}	Yes	No	Yes	2
{`Dport`}	Yes	Yes	No	2

Table 3. The uniqueness of different types of keys.

and *block scan* [14]. It is also crucial to distinguish them because network administrators need to apply different mitigation schemes for different attacks.

3.3 Sketch-based Detection Algorithm

We denote the key of a sketch as `K`, the feature value recorded as `V`, and the reversible sketch as `RS(K,V)`. We also denote the number of SYN packets as #SYN, and the number of SYN/ACK packets as #SYN/ACK.

Here, we only consider the attacks in TCP protocol, *i.e.*, the TCP SYN flooding attacks and TCP port scans. Normally, attackers can choose source ports arbitrarily, so `Sport` is not a good metric for attack detection. For the other three fields, we can consider all the possible combinations of them, but the key (`SIP`, `DIP`, `Dport`) can only help detect non-spoofed SYN flooding, so we do not use it in the detection process. Table 3 shows the other combinations and their uniqueness. Here, we define the *uniqueness* of a key as the capability of differentiating between different types of attacks. For example, the count of unsuccessful connections aggregated by {`SIP`} can be used to detect non-spoofed SYN flooding attacks (we count it as 0.5), horizontal scans and vertical scans, so its value of uniqueness is 2.5. The best key would ideally correspond to only one type of attack. Normally a key can be related to several types of attacks, so we need to use more than one dimension to differentiate these attacks as shown in Section 4. In this paper, we use the tree combinations of two fields as keys for the reversible 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 a 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 structures can be applied to detect any partial completion attacks [7]. The reversible sketch can further provide the victim IP and port number for mitigation as in Section 3.2. We denote this set of `DIPs` as $FLOODING\_DIP\_SET$.

Step 2, we use `RS({SIP, DIP}, #SYN-#SYN/ACK)` to detect any intruder trying to attack a particular IP address. The detected attacks 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`} is the attacker's IP and victim IP of a vertical scan.

Step 3, we use `RS({SIP, Dport}, #SYN-#SYN/ACK)` to detect any source IP which causes a large number of uncompleted 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.

Here we apply EWMA algorithm as the forecast models to do change detection. We denote $\mathbf{M}_0(t)$ as the current `#SYN-#SYN/ACK` at the time interval t, and $\mathbf{M}_f(t)$ as the forecasted `#SYN-#SYN/ACK` at the time interval t, we have

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

The difference between the forecasted value and the actual value, $\mathbf{e}_t = \mathbf{M}_0(t) - \mathbf{M}_f(t)$, is then used for detection.

3.4 Reducing False Positives(FP) for SYN Flooding Detection

There can be a number of factors other than SYN flooding that may cause a particular destination IP and port with a large number of unacknowledged SYNs. For instance, flash crowds, network/server congestions/failures, and even polluted or outdated DNS entries may cause a large number of SYNs without SYN/ACK at the edge routers. These may cause high false positives in our detection scheme. For the flash crowds, it is difficult, if not impossible, to differentiate it from the SYN flooding attacks without payload information as discussed in [6]. Thus we aim at reducing the false positives caused by the other two behaviors listed above.

First, we add filters to reduce false positives caused by bursty network/server congestions/failures based on the ratio of #SYN comparing with #SYN/ACK and the fact that attacks may last some time. Second, we add filters to reduce the false positives caused by misconfigurations or related problems based on the fact that DoS attacks attack some active IP addresses and services.

3.5 DoS Resilience Analysis

As we mentioned before the TRW is vulnerable to spoofed IP attack. The attacker can send a lot of SYN packets with spoofed source IP address and random destination IP address (within the edge network). This will cause the TRW to use too much memory and possibly crash, since the TRW needs to keep states for each source IP address. The TRW-AC uses an AC table with fixed memory to improve the scalability of the TRW, so the attack cannot crash the TRW-AC. However, as mention in the TRW-AC paper [16] itself, the more source spoofed packets, the more collision happen in the AC table. Hence, this makes the TRW-AC suffer high false negatives. For example, in their paper, they use the connection cache size of 1 million entries, and the $D_{conn} = 10$ minutes[2]. If the attacker periodically sends 1 million IP spoofed packets in 10 minutes (1667 packets/second, 533Kb/s for 40 bytes SYN packets), he can fully pollute the connection cache with half-open connections.

The HiFIND system is resilient to such attacks. If an attacker sends the source spoofed SYN packets to a fixed destination, our system will treat this as a SYN Flooding attack to the particular IP address. If the attacker sends the source spoofed packets to random destinations within the edge network, the SYN count will be evenly distributed in the buckets of each of the hash tables in sketches. Even if there is a real attack, the SYN count for that attack is still significant to be detected.

One possible attack is to introduce false positives or false negatives by creating collisions in the hash tables of sketches. To create collisions, the attacker needs to reverse engineer all the hash functions of sketches and search exhaustively offline. Since the interval parameters of hash functions used by sketches are independent from the functionality sketches archived, it is very difficult to infer the parameters solely from input and output of sketches. Therefore, such reverse engineering is impossible unless they can compromise the HiFIND system and obtain the intermediate execution results. Finally, even if the attacker can somehow create collisions for sketches, when we monitor both ingress and egress traffic for detection, they can at most create some false positives, but not false negatives. Overall, it is extremely difficult to attack the HiFIND system. For the detailed analysis please refer to our technical report [5].

4 Intrusion Classification with Two Dimensional Sketch

It is crucial to distinguish different types of attack to take the most effective mitigation scheme. However, one major challenge for intrusion detection is that the traffic anomalies are often multidimensional *i.e.*, they can only be identified when we examine traffic with specific combinations of IP addresses, port numbers, and protocols. For example, if the port distribution of a particular attack is unknown, it becomes very hard to distinguish non-IP-Spoofing SYN flooding attacks from vertical scans because both of them will exhibit a single (or a small number) of source IPs sending a large number of un-responded SYN packets to the destination IP. The key difference lies in whether the attacker sends to a small number, *e.g.*, one or two, of the ports (SYN flooding) or many different ports (vertical scan) on the destination. In other words, there is a bi-modal distribution for the number of unique ports visited when there are a large number of un-responded SYN packets from one source to one destination. One mode corresponds to the SYN flooding, and the other, vertical scans. This bi-modal assumption is verified with real network traffic shown in figure 4. We tested the bi-modal assumption with one-month edge router traffic from Northwestern University (NU) and Fermi Lab. Thus, it is essential to know the port distribution, given a specific source IP and destination IP pair {SIP,DIP}, to distinguish between these two attacks. However, existing sketches are all of single dimension. To address this challenge, we design a novel two-dimensional (2D) k-ary sketch and apply it for intrusion classification.

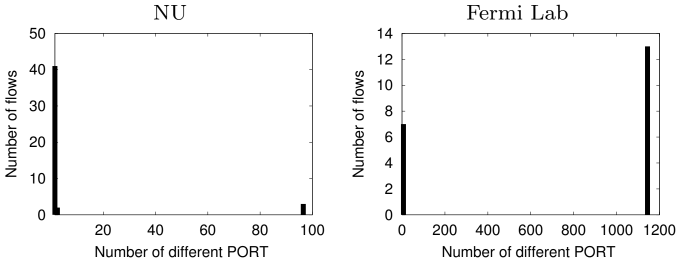

Figure 4. The distribution of the number of attacks with respect to the number of unique ports visited when there are more than 50 un-responded SYNs in 1-minute interval between one {SIP,DIP} pair.

For the 2D k-ary sketch, instead of using H independent one-dimensional hash tables, we use H independent 2D hash tables (matrices), as shown in Figure 5. Let K_x and K_y denote the number of buckets for each dimension respectively. For each 2D hash matrix, we hash two groups of fields into it. Consider the previous example of separating SYN flooding attacks from vertical scans. The x dimension represents the {SIP,DIP}, and the y dimension corresponds to Dport. For each packet, we locate its corresponding entry in the matrix by two independent hash mappings as shown in Figure 6, and update

[2] The TRW-AC uses a background process to remove any connection idle for more than D_{conn} minutes

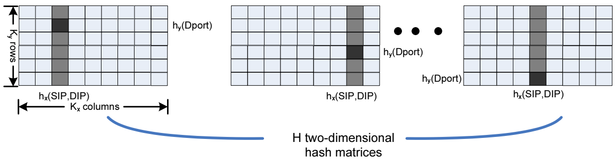

Figure 5. Diagram of the two-dimensional k-ary sketch

the bucket in the same manner as for reversible sketches in Section 3.3. Similarly, we can update all H matrices for data stream recording.

In the detection stage, after finding an attack by using reversible sketch or any other method (*i.e.*, the {SIP,DIP} is known), we can use the column of buckets in the hash matrix selected by the {SIP,DIP} to infer the distribution of Dport and pinpoint the type of attack, *e.g.*, a SYN flooding or a vertical scan. The algorithm is as follows. For each 2D hash matrix, the {SIP,DIP} pair selects a column of buckets. We define B to be the total sum of all buckets in the column. We then obtain the sum S_p of the top p buckets with the largest values (*e.g.*, 5 out of 64). If $S_p > \phi \times B$ for some $\phi < 1$, *e.g.*, 0.8., then we regard it as a SYN flooding. If the majority of the H hash matrices of the 2D sketch imply it is a SYN flooding, we conclude it is a SYN flooding attack; otherwise we conclude it is a vertical scan. Similarly, we can differentiate horizontal scans from the SYN flooding attacks.

In our technical report [5], we analytically proved that the 2D sketches are highly accurate.

5 Evaluation

5.1 Evaluation Methodology

In this section, we evaluate HiFIND with two dataset. One is the router traffic traces collected at the Lawrence Berkeley National Laboratory (LBL) which consists of about 900M netflow records. The other is the traffic traces of Northwestern University (NU, which has several Class B networks) edge routers. The router exports netflow data continuously which is recorded with sketches of HiFIND on the fly. The one day experiment in May 2005 consists of 239M netflow records, which comes 1.8T total traffic.

Unless denoted otherwise, the default time interval for constructing the time series is one minute. The data recording part of the HiFIND system consists of 1) three reversible sketches (RS), one for {SIP,Dport}, one for {DIP,Dport}, and the other for {SIP,DIP}, 2) one original sketch (OS) for {DIP,Dport}, and 3) two 2D sketches for {SIP,Dport} × {DIP} and {SIP,DIP} × {Dport}. For all the RS and 2D sketches we update #SYN - #SYN/ACK as the value, and only for the OS, we use #SYN as the value.

The following parameters are chosen based on systematic study as in [8, 12]. We adopt 6 stages for each RS and OS, and 5 stages for each 2D sketch in our system. We use 2^{12} buckets for each stage in 48-bit RS, 2^{16} buckets for each stage in the 64-bit RS, and 2^{14} buckets for all their verification sketches. 2^{14} buckets are applied for each stage in OS. We also use $2^{12} \times 64$ buckets for each stage of the 2D sketches. Therefore, the total memory is 13.2MB.

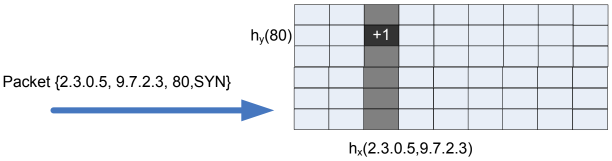

Figure 6. An example of UPDATE operation for two-dimensional sketch

Traces	Attack type	Phase1: Raw results	FP reduction	
			Phase2: Port scan	Phase3: Flooding
NU	SYN flooding	157	157	32
	Hscan	988	936	936
	Vscan	73	19	19
LBL	SYN flooding	35	35	0
	Hscan	736	699	699
	Vscan	40	1	1

Table 4. Detection results under three phases.

Both NU and LBL have a large amount of traffic, so we set the detection threshold to be one un-responded SYN packet per second.

5.2 Sketches Highly Accurate in Recording Traffic for Detection

Table 4 shows the three phases of our detection results. We first detect attacks using reversible sketches with algorithms described in Section 3.3. The results are shown as "Raw results"("Phase 1") in Table 4. 2D sketches reduce the false positives for port scans introduced by SYN flooding attacks ("Phase 2") of Table 4. The heuristics in Section 3.4 reduce false positives of SYN flooding attacks("Phase 3").

To evaluate the errors introduced by sketches, we compare the results obtained from the same detection algorithm but with two different types of traffic recording: 1) sketches; 2) accurate flow table to hold per-flow information (we call it non-sketch method). We find that we detect exactly the same attacks for the two configurations with very different amounts of memory (see memory consumption discussion in Section 5.5). There is no false positive in our results. This shows sketches are highly accurate in recording the traffic for detection.

5.3 HiFIND Outperforms Other Existing Network IDSes

5.3.1 Detection Over a Single Router

We compare the HiFIND with other state-of-the-art work as introduced in Section 2: the TRW [6] for port scan detection and the CPM [15] for SYN flooding detection.

Data	TRW	HiFIND	Overlap number
NU	497	512	488
LBL	695	699	692

Table 5. Horizontal scans detection comparison of HiFIND and TRW aggregated by source IP.

Anonymized SIP	Dport	#DIP	Cause
204.10.110.38	1433	56275	SQLSnake scan
109.132.101.199	22	45014	Scan SSH
95.30.62.202	3306	25964	MySQL Bot scans
162.39.147.51	6101	24741	Unknown scan
15.192.50.153	4899	23687	Rahack worm

Table 7. Five major senarios of the top 10 Hscans in NU experiment.

Data	CPM	HiFIND	Overlap number
NU	1422	1427	1422
LBL	1426	0	0

Table 6. TCP SYN flooding detection comparison of HiFIND and CPM.

Anonymized SIP	Dport	#DIP	Cause
98.198.251.168	135	64	Nachi or MSBlast worm
3.66.52.227	445	64	Sasser and Korgo worm
2.0.28.90	139	64	NetBIOS scan
98.198.0.101	135	64	Nachi or MSBlast worm
165.5.42.10	5554	62	Sasser worm

Table 8. 5 major scenarios of the bottom 10 Hscans in NU experiment.

For TRW experiments, we choose similar parameters as those in their paper. We apply the TRW on both datasets with the same threshold. Repeated alerts are removed from the results of both methods. Table 5 shows the comparison results of our methods with TRW for Hscan detection. We observe that the scans detected by these two methods have very good overlap, except for a few special cases. There are a small number of Hscans detected by HiFIND but not TRW, because some attacks have both successful and unsuccessful connection attempts, but TRW cannot detect those suspicious ones in this category. There are also a very small number of Hscans detected by TRW but not HiFIND, because they are the combination of multiple small scans, which are too stealthy to be captured by our threshold. It is our future work to further investigate it.

Next, we compare our method with CPM for SYN flooding attack detection. The results are shown in Table 6. In the LBL traces, there is no SYN flooding, but a very large number of scans. CPM cannot differentiate them. On the other hand, CPM and HiFIND have very similar results for the NU data because most time intervals contain SYN flooding. Meanwhile, there is a small number of intervals in which SYN flooding is buried in the rest of the normal traffic, so CPM cannot detect them.

5.3.2 Aggregated Detection over Multiple Routers

In this section, we consider the network topology of Figure 3 discussed in Section 3 and evaluate the performance of HiFIND and TRW under such scenarios. To simulate asymmetric routing and multi-path routing caused by per-packet load balancing on routers, we split the packet level trace from a Northwestern University edge router into three routers randomly, for both inbound and outbound packets. For each packet, we randomly select an edge router to deliver, *i.e.*, for any single connection, the incoming SYN packet and the outgoing SYN/ACK packet have 2/3 probability to go through different routers.

For HiFIND , we obtain the same results as those when the traffic goes through the same router, *i.e.*, the results in Section 5.3.1. In comparison, we apply TRW to the data on each router for detection and then sum the result up. We found their approach had high false positives or negatives in this case.

5.4 Detected Intrusions Successfully Validated

In this section we manually examine a certain number of attacks for validation.

SYN Flooding We validate our SYN flooding detection results with backscatter [9]. Among the 32 SYN floodings detected, there are 21 matched with backscatter results. For the other 11 attacks, three are due to threshold boundary effect.

5.4.1 Horizontal Scans

We manually validate horizontal scans, in particular, the top 5 and bottom 5 attacks in terms of their change difference. Due to limited space, Table 7 shows the top 5 Hscans, and Table 8 shows the bottom 5 Hscans from the NU experiment. Detailed evaluation can be found in our technical report [5].

5.4.2 Vertical Scans

We also manually validate vertical scans. In the LBL trace, we found one vertical scan. It scanned some well-known service ports, such as HTTPS(81), HTTP-Proxy(8000,8001,8081). In the NU experiment, we found in total 19 vertical scans. We manually checked all of them and found the vertical scans are mostly interested in the well known service ports and Trojan/Backdoor ports.

Methods	2.5Gbps		10Gbps	
	1min	5min	1min	5min
HiFIND w/ sketch	13.2M		13.2M	
HiFIND w/ complete info	10.3G	51.6G	41.25G	206G
TRW	5.63G	28G	22.5G	112.5G

Table 9. Memory comparison(bytes).

5.5 Evaluation Results for Online Performance Constraints

5.5.1 Small Memory Consumption

In our experiments, we only use a total memory of 13.2MB for traffic recording. Note that such settings work well for a large range of link speeds.

On the other hand, if hash tables are used to record every flow, much larger memory is required as shown in Table 9. We consider the worst-case traffic of all-40-byte packet streams with 100% utilization of the link capacity. There is a spoofed SYN flooding attack with a different source IP for each packet. For the method without sketch, it needs at least three hash tables corresponding to the three reversible sketches in our detection methods.

5.5.2 Small Memory Access per Packet

There are 15 memory accesses per packet for 48 bit reversible sketches and 16 per packet for 64-bit reversible sketches (see [12] for details). For each two-dimensional sketch, we only need 5 memory accesses per packet, one for each 2D hash matrix. Thus, when recording these sketches in parallel or in pipeline, the HiFIND system has a very small number of memory accesses per packet and is capable of online monitoring.

5.5.3 High Speed Traffic Monitoring

In HiFIND system, the speed of 2D sketches is much faster than that of the reversible sketches. Thus, the speed is dominated by the latter. With our prototype single FPGA board implementation, we are able to sustain 16.2 Gbps throughput for recording all-40-byte packet streams (the worst case) with a reversible sketch.

We can also use multi-processors to record multiple sketches simultaneously in software. We record 239M items with one reversible sketch in 20.6 seconds, *i.e.*, 11M insertions/sec. For the worst case scenario with all 40-byte packets, this translates to around 3.7 Gbps. These results are obtained from code that is not fully optimized and from a machine that is not dedicated to this process.

For the on-site NU experiments, the HiFIND system used 0.34 seconds on average to perform detection for each one-minute interval, and the standard deviation is 0.64 seconds. The maximum detection time (for which the interval contains the largest number of attacks) is 12.91 seconds, which is still far less than one minute. In order to show the scalability of HiFIND, we further do some stress experiments. We compress the NU data by the factor of 60, and detect the top 100 anomalies in each interval. The HiFIND system used 35.61 seconds on average in detection for each interval. The maximum detection time is 46.90 seconds.

6 Conclusion

In this paper, we propose, implement and evaluate a DoS resilient High-speed Flow-level Intrusion Detection system, HiFIND, leveraging recent data streaming techniques such as reversible sketches. Experiments with several router traces show that HiFIND is highly accurate, efficient, uses very small memory, and can effectively detect multiple types of attacks simultaneously.

References

[1] Arbor Networks. Intelligent Network Management with Peakflow Traffic. http://www.arbornetworks.com/download.php.

[2] Cisco Inc. Per-Packet Load Balancing, 2003. http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newf%t/120limit/120s/120s21/pplb.pdf.

[3] G. Cormode and S. Muthukrishnan. What's new: Finding significant differences in network data streams. In *Proc. of IEEE Infocom*, 2004.

[4] N. Duffield, C. Lund, and M. Thorup. Flow sampling under hard resource constraints. In *Proc. of ACM SIGMETRICS*, 2004.

[5] Y. Gao, Z. Li, and Y. Chen. Towards a high-speed router-based anomaly/intrusion detection system. http://list.cs.northwestern.edu/hpnaidm.html.

[6] J. Jung et al. Fast portscan detection using sequential hypothesis testing. In *Proc. of the IEEE Symposium on Security and Privacy*, 2004.

[7] R. R. Kompella, S. Singh, and G. Varghese. On scalable attack detection in the network. In *Proc. of ACM/USENIX IMC*, 2004.

[8] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation, and applications. In *Proc. of ACM SIGCOMM Internet Measurement Conference (IMC)*, 2003.

[9] D. Moore et al. Inferring Internet denial of service activity. In *Proceedings of the 2001 USENIX Security Symposium*, Aug. 2001.

[10] D. Moore et al. The spread of the Sapphire/Slammer worm. http://www.caida.org, 2003.

[11] R. Schweller, A. Gupta, E. Parsons, and Y. Chen. Reversible sketches for efficient and accurate change detection over network data streams. In *IMC*, 2004.

[12] R. Schweller, Z. Li, Y. Chen, et al. Reverse hashing for high-speed network monitoring: Algorithms, evaluation, and applications. In *Proc. of IEEE Infocom*, 2006.

[13] S. Venkataraman, D. Song, P. Gibbons, and A. Blum. New streaming algorithms for superspreader detection. In *the Annual Network and Distributed System Security Symposium (NDSS)*, 2005.

[14] Vinod et al. Internet intrusions: Global characteristics and prevalence. In *Proc. of ACM SIGMETRICS*, 2003.

[15] H. Wang, D. Zhang, and K. G. Shin. Detecting SYN flooding attacks. In *Proc. of IEEE INFOCOM*, 2002.

[16] N. Weaver et al. Very fast containment of scanning worms. In *USENIX Security Symposium*, 2004.

[17] Q. G. Zhao, A. Kumar, and J. J. Xu. Joint data streaming and sampling techniques for detection of super sources and destinations. In *Proc. of ACM/USENIX Internet Measurement Conference*, 2005.