HRAID: a High-speed Router based Anomaly/Intrusion Detection System

Yan Gao, Zhichun Li, Robert Schweller, Yan Chen

Northwestern University {ygao, lizc, schwellerr, 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. We propose a High-speed Router-based Anomaly and Intrusion Detection system, HRAID, leveraging our recent work on data streaming computation and in particular, sketches. 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. To overcome the limitations of existing single dimension sketches, we design an efficient and accurate two-dimensional sketch to distinguish different types of attacks for mitigation. We further propose several heuristics to reduce the false positive rate for SYN flooding detection. Simulation with several router traces shows that HRAID is highly accurate, efficient, uses very small memory, and can effectively detect multiple types of attacks simultaneously. To the best of our knowledge, HRAID is the *first* online flow-level anomaly/intrusion detection system for high-speed networks, even for the worst case traffic of 40-byte-packet streams with each packet forming a flow.

I. 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]. With the rapid growth of network bandwidth and fast emergence of new attacks/viruses/worms, existing network intrusion detection systems (IDS) [2–5] are insufficient for the following two reasons.

First, they are mostly host-based and not scalable to high-speed networks. Most existing IDSes reside on single host or low-end routers, examining application-level [2], system-level [3] logs, or the sniffed network packets [4, 5] because such detailed information can identify attacks on individual machines. However, today's fast propagation of viruses/worms (*e.g.*, SQL Slammer worm) can infect most of the vulnerable machines in the Internet within ten minutes [6] or even less than 30 seconds with some highly virulent techniques [7, 8]. 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 [9]. Also, the high speed backbone is suggested as a good vantage point for worm containment [10], for which worm detection on high speed backbone is a crucial prerequisite. Furthermore, for DDoS attacks, although the end hosts can detect and block them, they still can consume much of the network bandwidth or capacity causing the network to become unusable. In fact, it is very hard to implement certain detection techniques, such as those for port scanning, at end hosts. However, the existing schemes are not scalable to the link speeds and number of flows for high-speed networks.

Given a high-speed router with, *e.g.*, an OC-192 link (10Gbps), each 40-byte TCP packet only has 32 ns to proceed [11]. Therefore, three performance features are strongly desirable for high-speed network monitoring and intrusion detection systems: 1) a small amount of memory usage (to be implemented in SRAM); 2) a small number of memory accesses per packet [12, 13]; and 3) scalabilty to a large key space size. The last constraint is especially important for the coming decade: IPv6 with its 128 bit IP address is being adopted, especially in Asia. Thus, the system should scale to a key space of 2^{128} or 2^{256}.

Second, most of the existing approaches are signature based, which cannot detect unknown network attacks. Also, attackers can easily evade such IDSes by changing the attack signatures with polymorphism. Statistical IDSes are therefore proposed to detect anomalous behaviors. Current systems are mostly designed to detect based on the overall traffic [14–18]. They therefore 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 [4, 5, 19], but they are not scalable and thus are themselves vulnerable to attacks. We compare these schemes with ours in Section II.

To address these challenges, we propose a new paradigm called High-speed Router-based Anomaly and Intrusion Detection system, HRAID, leveraging the recent work on data streaming computation and in particular, sketches [20, 21]. To the best of our knowledge, HRAID is the *first* online flow-level anomaly/intrusion detection system for tens of Gigabit links, even for the worst case traffic of 40-byte-packet streams with each packet forming a flow. Essentially, we want to detect as many attacks as possible. As the first step towards this ambitious goal, our threat model includes various port scans (which covers most large-scale worm propagation) and TCP SYN flooding. Our goal is to identify and distinguish them in real-time on high speed networks, and to obtain the attacks' key characteristics for mitigating the attacks. Note that while each of these attacks seems relatively easy to detect separately, or in an offline setting, it is in fact very hard to detect a mixture of attacks online at flow-level for high-speed networks. As far as we know, none of the previous work is able to achieve that as discussed in Section II.

To this end, we leverage and improve sketches, an efficient tool for data streaming computation, to record flow-level traffic as the basis for statistical intrusion detection. Although proposed in [20, 21], sketch has not been applied to building intrusion detection systems for the following challenges.

- Sketch can only record certain aggregated metrics for some given keys. For each flow, there are numerous possible keys: source/destination IP addresses, source/destination ports, source/destination prefixes, protocols, *etc.*, and any of these combinations. Since it is not feasible to try all possible combinations of the metrics, given the threat model, what would be the minimal set of metrics for monitoring?
- Existing sketches are all one dimensional, *i.e.*, they can only record the values for a specific metric. However, various forms of attacks are often hard to detect or identify with such single dimensional information. For example, both horizontal scans and un-spoofed SYN flooding exhibit large number of unsuccessful connections aggregated with the (source IP, destination port) pair. However, it is very hard to differentiate them unless the distribution of the attacks on the destination IPs are also considered.
- Many common network element faults, *e.g.*, congestion/failures, routing misconfigurations, and polluted DNS entries, can cause the erroneous detection of traffic anomalies through the statistical algorithms. How can these be separated from attacks to reduce false positives?

In this paper, we address these challenges with the following approaches.

- We analyze the attributes in TCP/IP headers and select an optimal small set of metrics for flow-level sketch-based traffic monitoring. Time series analysis algorithms like exponentially-weighted moving average (EWMA) are applied to detect anomalies and intrusions.
- We design efficient two-dimensional sketches to distinguish different types of attacks for mitigation. Both analytical and empirical results show the effectiveness of the 2-dim sketches.
- We propose several heuristics to separate SYN floodings from network/server congestions and misconfigurations.

As shown in Figure 1, HRAID detection systems can be implemented as black boxes to attach to high-speed routers (edge network routers or backbone routers) of ISPs without affecting the normal operations of the routers. They can enable the early detection of global scale attacks, which is crucial because such attacks can subvert the networking infrastructure and many hosts instantly with exponential growth.

Detection on edge networks is particularly critical, powerful and efficient (without deploying IDSes on all the edge hosts), according to a recent research agenda for large scale malicious code by a recent DARPA research agenda [9].

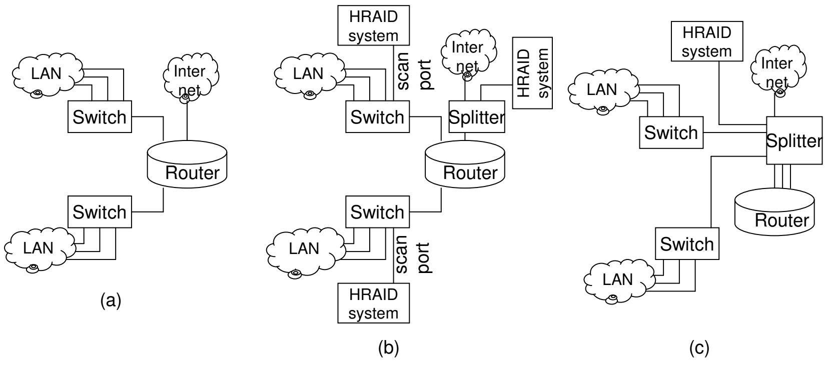

Fig. 1. Attaching the HRAID systems to high-speed routers. (a) original configuration, (b) distributed configuration for which each port is monitored separately, (c) aggregate configuration for which a splitter is used to aggregate the traffic from all the ports.

It can be naturally extended for response, especially at the edge networks. That is, when alerted, filters can be automatically constructed to filter classes of traffic, and isolate infected machines. Again, such a scheme is considered as "powerful and flexible" by [9].

For evaluation, we first use the router traffic traces collected at Lawrence Berkeley National Labs. We then apply HRAID for on-site detection at the Northwestern University (NU) edge routers: we record each minute of traffic with reversible sketches on the fly. At the end of each minute, we use the recorded sketches for online detection. In particular, the one day experiment data consists of 239M network flows of 1.8TB total traffic. We validated the SYN flooding and port scans detected, and found the HRAID system is highly accurate. The 2-dim sketches successfully separate the SYN flooding from port scans, and the heuristics effectively reduce false positives of SYN flooding. The evaluation demonstrates that HRAID significantly outperforms existing approaches like Threshold Random Walk (TRW) [19] and Change-Point Monitoring (CPM) [15, 16]. Compared with statistical detection with complete flow-level data logs, we use much less memory, but have almost the same detection accuracy.

The HRAID system runs in real-time, and requires a small number of memory accesses per packet. With a Pentium Xeon 3.2 GHz machine with normal DRAM memory, we record 239M flows with one reversible sketch in 20.6 seconds, *i.e.*, 11.6M insertions/second. For the worst case scenario with all 40-byte packets, this translates to around 3.7 Gbps. Even when we run the three reversible sketches of HRAID in serial, we can still archive 3.8M insertions/second. Our prototype single FPGA board for reversible sketches can achieve a throughput of over 16Gbps for all 40-byte-packet streams. For the NU on-site experiments over a total of 1430 minutes, HRAID on average used only 0.34 seconds to detect intrusions for each minute of traffic, and the standard deviation was 0.64 seconds.

The organization of this paper is as follows. First we survey related work in Section II. In Section III, we introduce the sketches and reversible sketches as the basis for high-speed network monitoring. We then introduce the HRAID architecture in Section IV, and discuss the threat model and flow-level monitoring and detection design in Section V. The two-dimensional sketches are presented in Section VI, and evaluation methodology and results are in Section VII.

Approaches	Spoofed DoS	Non-spoofed DoS	Horizontal Scan	Vertical Scan
HRAID	Yes	Yes	Yes	Yes
TRW(AC)	No	No	Yes	Yes(AC)
CPM	Yes, but with high FP with port scans		No	No
Backscatter	Yes	No	No	No
Superspreader	No	No	Yes	No

TABLE I
FUNCTIONALITY COMPARISON OF FOUR APPROACHES.

Finally, we conclude in Section VIII.

II. Related Work

Although some vendors claim to have multi-gigabit statistical IDSes (*e.g.*, Arbor Networks' Peakflow Traffic [22, 23] and Symantec's Manhunt [24]), they usually refer to *average* traffic conditions and use packet sampling [25, 26] which 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 [25]. In contrast, HRAID records *every* packet in the traffic summary, and targets *worst-case* performance of tens of gigabits, when all the packets are small, *e.g.*, 40-byte packet streams as in TCP SYN flooding attacks. In a similar spirit, Alcatel, Cisco and Enterasys Networks recently combined IDSes with their switches to defend against worms and DoS attacks [27]. They mostly only use signature-based detection, and all their techniques are proprietary.

Many network IDSs like Bro [4] and Snort [5] check packet payload for virus/worm signatures. However, such schemes are not scalable for high-speed network links. Recently, some works have 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: 1) Detecting based on the overall traffic [14–17] and 2) flow level detection [4, 5, 19], *e.g.*, TRW.

With the first approach, attacks can be easily buried in the background network traffic. Thus, such detection schemes tend to be inaccurate or cannot find real attack flows; For example, CPM [15, 16] will detect port scans as SYN floodings as verified in Section VII. For the second type, such schemes usually need to maintain a per-flow table (*e.g.*, a per-source-IP table for TRW [19]) 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 (AC) [28]. However, spoofed DoS attacks will still cause collisions in AC, and make the real port scans undetected[1].

[1]As the authors mentioned in [28], 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.

In short, 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. Thus, it is like a chicken-and-egg problem. TRW and AC are vulnerable to spoofed DoS attacks, such as SYN flooding, unless CPM can detect DoS attacks accurately and remove them from the traffic. However, with port scans in the traffic, CPM will always detect them as SYN floodings. As a result, this combination cannot work.

Table I shows the high-level functionality comparison of our approach to the other methods. Backscatter detects the SYN flooding attacks by testing the uniform distribution of destination IPs to which the same source (potential victim) sends SYN/ACK [14]. Thus, it is another flow-level scheme and can only detect spoofed DoS attacks when the source IP addresses are randomly spoofed. We use that for validating the SYN flooding detected by HRAID.

Venkataraman *et al.*propose efficient algorithms to detect superspreaders, sources that connect to a large number of distinct destinations [29]. They can detect horizontal scans and worm propagation, but 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 [18]. It uses similar data structures as an original sketch, and thus is not reversible. So even when attacks are detected, the attacker or victim is still unknown, making mitigation impossible. Similar to the approaches discussed before, they do not differentiate from various attacks.

For intrusion classification, Lakhina *et al.*recently examine the traffic feature distribution with entropy, for all OD flows between a pair of point of presence (POPs) [30]. Their scheme, however, cannot give the key of culprit flows for mitigation even when spotting anomalies. Similarly, other recent work [31, 32] use traffic feature distributions or patterns to classify network flows for anomaly detection, but their methods require the complete flow tables which are often unavailable for high-speed networks of 10s of Gigabits.

III. Background on Sketches for High-speed Network Monitoring

A. k-ary Sketch

The sketch, a recently proposed data structure, has proven to be useful in many data stream computation applications [33]. We have designed, implemented and evaluated a variant of the sketch, namely the k-ary sketch [34], and displayed how to detect heavy changes in massive data streams with small memory consumption, constant update/query complexity, and provably accurate estimation guarantees [34]. It is also flexible and can be applied with various forecast models to detect anomalies.

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

TABLE II
FUNCTION OF SKETCHES (S-SKETCH, v-VALUE, y-KEY, Y-SET OF KEYS, t-THRESHOLD).

The input stream arrives sequentially, item by item. Each item consists of a key and an update $value$. The k-ary sketch has three basic functions: UPDATE, ESTIMATE and COMBINE. Among them, UPDATE is most frequently used and has the most stringent real-time requirement. Table II shows the definition of these functions. Suppose there is a sketch with a source IP as the key and packet size as the value. When a 68-byte packet arrives, the UPDATE function will increase the value of the corresponding source IP in the sketch data structure by 68. The ESTIMATE function will return an unbiased estimation of the value (total traffic volume) given a source IP and a sketch. The COMBINE function can linearly combine several sketches into a single one. The key feature of the COMBINE function is to support aggregate queries over multiple data streams, *i.e.*, to find the top heavy hitters and their keys from the linear combination of multiple data streams, for temporal and/or spatial aggregation. Many statistical approaches, such as Time Series Analysis (TSA), need this functionality for anomaly/trend detection. Moreover, the linearity of reversible sketches enables traffic aggregation over multiple routers to facilitate distributed detection. With sketches, we can record hundreds of millions of flows with only a few hundred kilobytes of memory. See [34] for more details.

B. Reversible k-ary Sketch

Although the sketch data structures have good linear properties and can accurately estimate the value for any given key, they have one major drawback: they are not *reversible*. That is, a sketch cannot efficiently report the set of all keys that have large value estimates in the sketch. This means that to compare two streams, we would have to know which keys to query and the streams with the largest changes. Naively, this would require either exhaustively testing all possible keys, or recording and testing all data stream keys and corresponding sketches. Unfortunately, neither option is scalable.

To address these problems, we propose a novel framework for efficiently *reversing* sketches [20, 21]. 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 intact, such as big changes in a given category, without sacrificing detection accuracy. We also use a second verifier sketch with 2-universal hash functions to reduce false positives. In fact, we obtain analytical bounds

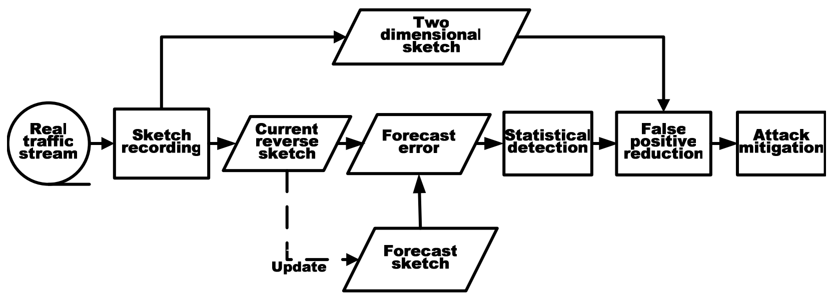

Fig. 2. HRAID System architecture.

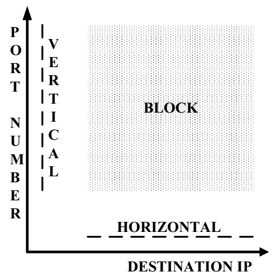

Fig. 3. Three types of scans.

on the false positives with this scheme. In addition to the three basic functions of sketches, reversible sketches also support the INFERENCE function, which will return the keys whose values are larger than a given threshold. Given the example before, we may want to get all source IPs whose total traffic volume are larger than 1MB in a certain time period. We just need to record the traffic of that period with reversible sketches, and call the INFERENCE function to give the list of all such source IPs. Compared with original k-ary sketches, we only add negligible extra memory consumption (4KB - 8KB) and few (4 to 8) additional memory accesses per packet for the UPDATE function, but we achieve efficient and accurate INFERENCE function. The other two functions are unchanged. For more details, please refer to our technical report [21].

IV. Architecture of the HRAID System

Figure 2 shows the architecture of the HRAID system. First, we record the network traffic with sketches using the UPDATE function. We then apply different time series analysis methods to obtain the forecast sketches for change detection by the COMBINE function, based on the linearity properties of sketches. The forecast time series analysis method, *e.g.*, EWMA (exponentially weighted moving average) and Holt-Winter algorithm [35], 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 adopt the two-dimensional sketches to further distinguish different types of attacks, and apply other false positive reduction techniques as discussed in Section V-C. Finally, we use the key characteristics of the culprit flows revealed by the reversible sketches to mitigate the attacks.

Here, we apply the EWMA algorithm as the forecast model to do change detection. The following characteristic we used for detection for EWMA 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 $\mathbf{M}_f(t)$ as the forecasted `#SYN−#SYN/ACK` at the time interval t, we have

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

In HRAID, we use COMBINE(α, $S_0(t-1)$, $1-\alpha$, $S_f(t-1)$) for implementation, where $S_0(t)$ and $S_f(t)$ are the current sketch and the forecast sketch at the time interval t respectively. The difference between the forecasted value and the actual value is then used for detection. That is, we use the INFERENCE function to output the set of keys whose forecast error $\mathbf{y}_t = \mathbf{M}_0(t) - \mathbf{M}_f(t)$ is larger than some given threshold for detection.

V. The Threat Model and Detection Algorithms

A. The Threat Model

Ultimately, we want to detect as many different types of attacks as possible. As a first step, we focus on arguably the two most popular intrusions for detection: TCP flooding denial of service (DoS) attack and port scan/worm propagation. Actually, our work also can be easily extended to detect UDP or ICMP flooding by using the number of packets or traffic volume instead of `#SYN-#SYN/ACK`. We focus on TCP flooding here because it is reported that more than 90% of DoS attacks are TCP SYN flooding attacks [15, 16].

Scans are probably the most common type of intrusion and many of them are caused by worm propagation. According to statistics data by CERT [36], most existing worms are TCP worms. Most TCP worm propagation starts with massive TCP connection requests, *i.e.*, port scans. Our system detects generic port scans. Thus, it can detect propagation for both known and *unknown* worms. Based on source/destination IP and the port number combinations, there are three well known types of scans: *horizontal scan, vertical scan,* and *block scan* [37, 38] as illustrated in Figure 3.

Unlike DoS attacks, for port scans, the attacker needs to use a real source IP address, since he/she needs to see the result of the scan in order to know what ports are actually open [37, 38]. Although the "idle scan" allows completely blind scans, it is based on predictable IP-ID sequence numbers and recent versions of operating systems, such as OpenBSD, Solaris and Linux, have made the IP-ID sequences less predictable and rendered the attack obsolete [39]. Among the three types of scans in Figure 3, horizontal scans are the most common type of scan, and scan a given port on IP addresses in some range of interest. The port number is often unique because it reflects the vulnerability attackers (or the virus/worm) try to exploit. A vertical scan is a scan of some or all ports on a single host. The attacker is interested in this particular host, and wishes to characterize its active services to find which exploits to attempt [38]. The third type of scan, a block scan, is a combination of horizontal and vertical scans over numerous services on numerous hosts [38]. Block scan is not very common in practice, thus we focus on horizontal and vertical scans.

B. Sketch-based Detection Algorithm

The challenge then 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 in order to mitigate them efficiently.

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 the IP header, the possible fields we can use are shown in Figure III.

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 detection. Figure IV 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, which is measured by the types of attacks that the key metric is related to and can help detect. The best key would ideally correspond to only one type of attack. Nevertheless, normally a key can be related to several types of attacks, so we need use more dimensions to differentiate these attacks as shown in Section VI.

the destination IP	`DIP`
the source IP	`SIP`
the destination port	`Dport`
the source port	`Sport`

TABLE III
THE FIELDS IN IP HEADER USED IN DETECTION

Types of Keys	SYN flooding	horizontal scan	vertical scan	block scan
{`SIP`,`Dport`}	non-spoofed	Yes	No	Yes
{`DIP`,`Dport`}	Yes	No	No	No
{`SIP`,`DIP`}	non-spoofed	No	Yes	Yes
{`SIP`}	non-spoofed	Yes	Yes	Yes
{`DIP`}	Yes	No	Yes	Yes
{`Dport`}	Yes	Yes	No	Yes

TABLE IV
THE UNIQUENESS OF DIFFERENT TYPES OF KEYS.

From Figure IV we can tell that the combinations of two fields have better uniqueness than single fields, so we use the 3 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 by one, while for each outgoing SYN/ACK packet, the sketch will be updated by decrementing by one. In fact, similar structures can be applied to detect any partial completion attacks [18]. The reversible sketch can further provide the victim IP and port number for mitigation as in Section V-D. We denote this set of `DIP`s 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. 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`} is the attacker and victim 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` $\in FLOODING\_SIP\_SET$,

Attack types	{`DIP, Dport`}	{`SIP, DIP`}	{`SIP, Dport`}
SYN flooding	Yes	Yes	Yes
Vertical scans	No	Yes	No
Horizontal scans	No	No	Yes

TABLE V
DIFFERENT ATTACKS ARE DETECTED IN DIFFERENT REVERSIBLE SKETCHES.

it is a non-spoofed SYN flooding; otherwise, it is a horizontal scan.

Thus altogether, we need three reversible sketches to record the traffic characteristics. The classification rules are depicted in Table V. 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 [15, 16]. After the detection, we use the heuristics described in Section V-C and 2-dim sketches in Section VI to reduce the false positives.

C. Reducing False Positives for SYN Flooding Detection

While port scans are relatively easy to identify with the algorithms above, 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 unresponded SYNs. For instance, flash crowds, network/server congestions/failures, and even polluted or outdated DNS entries may exhibit a large number of SYNs without SYN/ACK at the edge routers. These will cause relatively high false positives in our detection scheme. For the flash crowds, it is very hard, if not impossible, to differentiate it from the SYN flooding attacks without payload information as discussed in [40]. Thus we target reducing the false positives caused by the other two behaviors listed above.

a) Filters to reduce the false positives caused by bursty network/server congestions/failures: We believe a real SYN flooding should deny service to a certain degree, *e.g.*, the number of successful connections is less than the unsuccessful ones. That is, $\frac{SYN-SYN/ACK}{SYN/ACK}$ is greater than one. Moreover, the typical TCP SYN flooding attacks observed in the Internet last for a certain period, such as 10 minutes [16, 41]. On the other hand, most network congestions/failures are bursty, which may only last a few seconds or minutes. Based on these two observations, we introduce two filters to reduce the false positives of SYN flooding detection. The first one is $\frac{SYN-SYN/ACK}{SYN/ACK} > 1$, and the second one is $lifetime > Threshold_{life}$ (*e.g.*, 10 mins). The system administrators can adjust their threshold to get the tradeoff between sensitivity and the number of alerts.

To implement the first filter, we added an original sketch `OS({DIP, Dport}, SYN)` to record the number of SYNs for each {`DIP, Dport`} at each time interval. When a SYN flooding suspect is detected, we estimate the number of its SYNs from the original sketch. Then, based on $\frac{SYN-SYN/ACK}{SYN/ACK} > 1 \iff \frac{SYN-SYN/ACK}{SYN} > \frac{1}{2}$, we use the equivalent condition to decide whether the attack should be filtered.

To implement the second filter, we put any found SYN flooding suspect key into a suspect array of counters. Then, for each subsequent interval, we query the `SYN` and `SYN-SYN/ACK` value from the related sketches, and increment the counter if the above inequality holds.

b) Filter to reduce the false positives caused by misconfigurations or related problems: Usually, a SYN flooding attack will be launched towards some server which provides certain (popular) services. However, we found that some victims of detected SYN flooding are non-existing IPs or some hosts without any services on the ports being flooded. Moreover, the size of packets is small, so they are not bandwidth consumption attacks. Such anomalies look more like mistakes than real attacks. These may be caused by misconfigrations. For example, some administrators may misconfigure the DNS entry for a popular server as a non-existing IP or some IP which does not run the particular service. Some automatic connection software then may keep trying to connect to that server. Another possibility is that a DNS entry is updated, but the TTLs for the DNS entries are set/misconfigured to be very long, so the clients may still try to connect to the old IP which may not have the service open anymore.

When evaluating HRAID with the LBL and NU data as discussed in Section VII, we did find several such scenarios. For instance, in LBL data, we found there was a SYN flooding like anomaly towards a certain IP with large `#SYN-#SYN/ACK`. The anomaly lasted for 3 hours, but there were no good connections towards that IP during the day, before or after the anomaly. Thus, this is unlikely to be a real SYN flooding attack. Based on such observations, we can design another false positive reduction heuristic by filtering those anomalies for which the victim does not have any response to SYN requests during the recent 24 hour period. Due to the linearity of sketches, we can add up the sketches in the past 24 hours, and then estimate the number of SYN/ACK for the suspect {`DIP, Dport`}. If the number is less than a given threshold, *e.g.*100, it will not be considered as an attack. Note that we already have sketches to record SYN and (SYN-SYN/ACK), so no extra sketches are needed to record SYN/ACK for the monitoring phase.

D. Intrusion Mitigation

The HRAID system can instruct routers or system administrators to mitigate the attacks based on the attack type and their key characteristics as obtained from reversible sketches. For distributed SYN flooding, we can start the *SYN defender* [42], *SYN proxy* and/or *SYN cookies* [43] for the particular victim machines and applications to alleviate the DoS effects. Other DoS attack mitigation schemes [44] can also be applied. For port scans and point-to-point SYN flooding, we can use an 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 can pay particular attention to any suspicious internal or outgoing traffic with the same port number since the compromise of any internal machine means it will probably try to infect more machines within the network.

VI. Intrusion Classification with Two Dimensional Sketch

One major challenge for anomaly 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 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/ACK packets from one source to one destination. That is, one mode corresponds to the SYN flooding, and the other, vertical scans. This bi-modal assumption is verified with real network traffic in Section VI-A. Thus it is essential to know the port distribution, given a specific source IP and destination IP pair$\{\texttt{SIP,DIP}\}$, 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 in Section VI-B.

A. *Verification of the Bi-modal Distribution for SYN flooding vs. Port Scans*

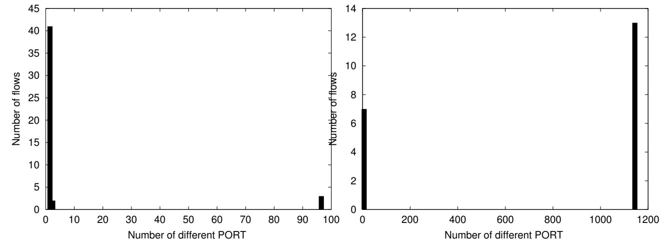

Fig. 4. The distribution of the number of attacks with respect to the number of unique ports visited when there are more than 50 un-responded SYNs in 1-minute interval between one $\{\texttt{SIP,DIP}\}$ pair. Left: NU, right: Fermi Lab

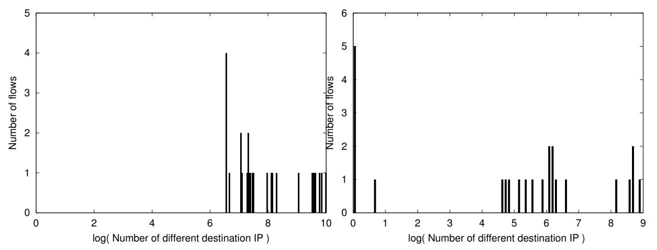

Fig. 5. The distribution of the number of attacks with respect to the number of unique destination IPs visited when there are more than 100 un-responded SYNs in 1-minute interval with one $\texttt{SIP,Dport}$ pair. Left: NU, right: Fermi Lab

We tested the bi-modal assumption with one-month edge router traffic from Northwestern University (NU) and Fermi Lab. With samples of two-hour NU data and one-day Fermi data, Figure 4 shows the distribution of the number of attacks with respect to the number of unique ports visited when there is more than 50 un-responded SYNs in a 1-minute interval from one source to one destination. There is a clear bi-modal distribution to distinguish SYN flooding attacks from vertical scans. The former usually have the number of ports visited at less than three. Similarly, there is also a bi-modal distribution between SYN flooding attacks and horizontal scans. That is, SYN flooding attacks have a very small number of destination IPs, while horizontal scans have a much larger amount. Figure 5 shows the distribution of the number of attacks with respect to the number of unique destination IP addresses visited when there are more than 100 un-responded SYNs in a 1-minute interval from one source to one destination port. Note that in the graph for NU, there is only one mode because there is no SYN flooding attack in the two hour NU traffic.

B. *Two-dimensional Sketch Design and Intrusion Classification*

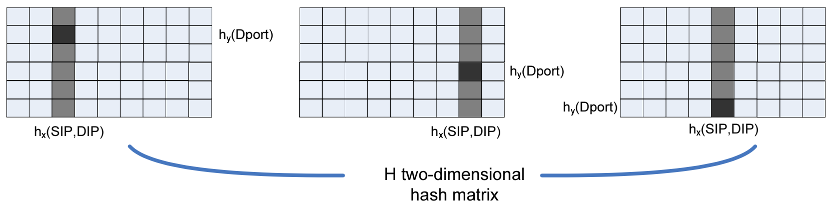

Fig. 6. Diagram of the two-dimensional k-ary sketch

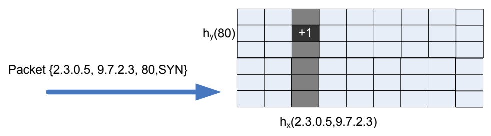

Fig. 7. An example of UPDATE operation for two-dimensional sketch

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 6. 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 is the concatenation of `SIP` and `DIP`, and the y dimension is `Dport`. For each packet, we locate its corresponding entry in the matrix by two independent hash mappings as shown in Figure 7, and update the bucket in the same manner as for reversible sketches in Section V-B. 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 then obtain the sum of the top p (*e.g.*, 5 out of 64) buckets with the largest values. If the sum is larger than ϕ ($\phi < 1$, *e.g.*, 0.8) times the total sum in all buckets B, 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.

C. *Accuracy Analysis*

An important issue with using the bi-modal distribution to separate SYN flooding attacks versus vertical scan attacks is the possibility of misclassification. The concern is that, by chance, a vertical scan attack could be hashed among the K_y buckets of a given column in a hash matrix such that a large number of attacked ports hash to just a single bucket, while the remaining attacked ports are spread out among many different buckets. Such a clumping would cause the detection algorithm to report a SYN flooding attack rather than the correct vertical scan attack (note that the reverse problem of misclassification should not be an issue). Alternatively, it is possible that a SYN attack key and a vertical scan attack key happen to hash to the same column. In such a case, since the hash table can only return SYN or vertical scan for the given column, it will be wrong. To address these issues, we first bound the probability of the first type of misclassificiation assuming that the given column has been hit by only one type of attack. We will then bound the likelihood of such a column being hit by two separate attacks to get a final bound on the probability of a misclassification.

To bound the probability of the first type of misclassification, we use the *Generalized Increment-Decrement Counter Model* [18]. If one SYN packet hashes to the bucket (counter), the counter increases by one; if one SYN/ACK packet hashes to the counter, the counter decreases by one. At the end of a sufficiently large amount of time called a *measurement interval*, T (*e.g.*, 1 minute), the counter should accumulate all the packets hashed to it to get the outcome value N. Now, for all normal traffic, there will be an equal number of SYN's and SYN/ACK's, thus contributing a total of zero to the hash matrix (this is not always true, but given a large enough measurement window and high volume traffic, it is approximately true).

Now, consider a column of the hash matrix that is hashed to by one single vertical scan key. We will assume that a vertical scan sends exactly c SYN's (for simplicity, assume $c = 1$) to B different ports. Thus, the total traffic for the target column will be B, and our algorithm will report the column as a SYN attack if the sum of the top p buckets S_p exceeds ϕB. Assuming uniformly random hash functions, we get the following bound on misclassification.

Lemma 1: Given a column of a hash matrix whose total SYN value comes from a single vertical scan key, the probability of misclassification is

$$Pr[S_p > \phi B] \leq K_y e^{\frac{-2\delta^2}{B}}$$

where $\delta = \frac{B}{p}(\phi - \frac{p}{K_y})$.

Proof: Let X_i be a random variable denoting the total traffic in bucket i of a given column. As each X_i is a binomial variable with mean $\frac{B}{K_y}$, from the Chernoff bound we get that

$$Pr[X_i > \frac{B}{K_y} + \delta] < e^{\frac{-2\delta^2}{B}}$$

For the largest X_i, X_{max}, we thus get

$$Pr[X_{max} > \frac{B}{K_y} + \delta] < K_y e^{\frac{-2\delta^2}{B}}$$

Further, we know that if the sum of the top p buckets S_p exceeds $p(\frac{B}{K_y} + \delta)$, then by the pigeon hole principle at least one X_i must exceed $\frac{B}{K_y} + \delta$, yielding the following inequality.

$$Pr[S_p > \frac{B \cdot p}{K_y} + p\delta] < K_y e^{\frac{-2\delta^2}{B}}$$

By setting $\delta = \frac{B}{p}(\phi - \frac{p}{K_y})$ we get the desired inequality. ■

Having bounded the possibility of misclassification assuming only one attack for a given bucket, we now consider the probability of actually getting this scenario. Let f and v denote the total number of SYN flood attacks and vertical scan attacks within a measurement interval T respectively. First, we will bound the probability of misclassifying a vertical scan. For a given key that constitutes a vertical scan, the key will be correctly categorized in any given hash matrix according to Lemma 1 unless a SYN flood attack key j also hashes to the same bucket (multiple vertical scans in a single bucket will still register as a vertical scan). Combining the probability of such a collision with Lemma 1 yields the following lemma.

Lemma 2: The probability that a given hash matrix h correctly classifies a vertical scan key k is

$$P[correct_h] \geq S = (\frac{K_x - 1}{K_x})^f (1 - error)$$

where $error = K_y e^{\frac{-2\delta^2}{B}}$ as stated in Lemma 1.

Proof: The probability of one or more of the f SYN flood keys hashing to a single given bucket is $(\frac{K_x-1}{K_x})^f$. Combining this event with Lemma 1 yields the result. ■

We can now use this lemma to derive the probability of failure for our algorithm that takes the majority result of the H hash tables.

Theorem 1: Given a key k that constitutes a vertical scan, the majority of the H hash tables will classify k as a vertical scan attack with probability at least

$$\sum_{r=\lfloor \frac{H}{2} \rfloor + 1}^{H} \binom{H}{r} S^r (1 - S)^{H-r}$$

where $S = (\frac{K_x-1}{K_x})^f (1 - error)$ and $error$ is the value stated in Lemma 1.

Proof: The result follows from observing that the total number of correct hash matrices is a binomial variable with probability of success equal to S by Lemma 2. ■

Theorem 2: Given a key k that constitutes a SYN flooding, the majority of the H hash tables will classify k as a SYN flooding attack with probability at least

$$\sum_{r=\lfloor \frac{H}{2} \rfloor + 1}^{H} \binom{H}{r} S^r (1 - S)^{H-r}$$

where $S = (\frac{K_x-1}{K_x})^{f+v}$.

Proof: The number of correct hash matrices is a binomial variable, so it is sufficient to show that the probability of successful classification for any given matrix is at least S. As discussed previously, A SYN attack can only be misclassified if there is a collision among attack keys. Thus, this constitutes a lower bound on the correct classification for a given matrix. ■

As an example of the bounds provided by Theorems 1 and 2, consider the following typical values. For $p = 5$, $\phi = 0.8$, $K_x = 4096$, $K_y = 64$ (these configurations are also adopted in the evaluation of Section VII), $B = 200$, $f = 5$, and $v = 5$, we get that a vertical scan will be detected with probability at least 99.9956% and a SYN attack will be classified correctly with probability at least 99.99999%. Note that the larger B is, the larger these probabilites are. We can achieve similar misclassification bounds for reporting a horizontal scan as a SYN flooding and vice versa.

VII. Evaluation

A. Evaluation Methodology

In this section, we evaluate HRAID with both simulation and on-site experiment.

- We use the router traffic traces collected at the Lawrence Berkeley National Laboratory (LBL) for simulation. The one-day trace consists of about 900M netflow records. Unfortunately, the sampling rate is unknown.
- We apply HRAID for on-site detection at the Northwestern University (NU, which has several Class B networks) edge routers. The router exports netflow data continuously which are recorded with sketches of HRAID on the fly. At the end of each minute, HRAID detects intrusions online as shown in Figure 2. We also record the complete netflow records for detection comparison. The one day experiment in May 2005 consists of 239M netflow records. The total traffic is 1.8TB. 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.

Unless denoted otherwise, the default time interval for constructing the time series for detection is one minute.

The evaluation metrics include *detection accuracy* (in terms of the true positive and false positive percentages), *execution speed*, *the amount of memory access per packet*, and *the amount of memory used in the recording stage*.

The HRAID 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`} $\times$ {`DIP`} and {`SIP,DIP`} $\times$ {`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 [21, 34]. 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.

Both NU and LBL have a large amount of traffic, so we set the detection threshold to be one scan per second for both horizontal and vertical scans, and one un-responded SYN packet per second for SYN flooding attacks. The thresholds can be adjusted by network administrators.

To evaluate the accuracy of the HRAID system, we take the following steps. First, we solely evaluate the error introduced by the sketches (Section VII-B). Second, we compare the HRAID system with several existing network intrusion detection approaches (Section VII-C). Finally, we manually validate the attacks we detected in Section VII-D.

B. Sketches Highly Accurate in Recording Traffic for Detection

Table VI shows the results when using EWMA as the forecasting method. There are three phases. We first detect attacks using reversible sketches with algorithms described in Section V-B. The results are shown as “Raw results” in Table VI. We then use 2D sketches to reduce the false positives for port scans introduced by SYN floodings, and get results in the column of “Phase 2” of Table VI. Finally, we use heuristics in Section V-C to reduce false positives of SYN flooding and the results are presented as “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 found that we detect exactly the same attacks for the two configurations with very different amount of memory (see memory consumption discussion in Section VII-E). This shows that sketches are highly accurate in recording the traffic for detection.

Traces	Attack type	Phase1: Raw results	False positive 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 VI
DETECTION RESULTS UNDER THREE PHASES.

C. HRAID Outperforms Other Existing Network IDSes

We compare our approach with other state-of-the-art network intrusion detection approaches as introduced in Section II: TRW [19] for port scan detection and CPM [15, 16] for SYN flooding detection.

Table VII shows the comparison results of our methods with TRW for horizontal scan detection. In TRW, the horizontal scans from the same source IP but to different ports are counted as only one attack. Thus, we aggregate the horizontal scans detected by HRAID with the source IP address. 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 horizontal scans detected by HRAID but not TRW due to the following reasons. Some attacks have both successful connection attempts and unsuccessful connection attempts. TRW gives each attempt different weights and multiplies them together for detection. If the product value is higher than a certain threshold, it will be identified as a port scan. In some cases, although the number of unsuccessful connections is fairly large, more than 500, they still have several successful connections, such as 100 to 150. TRW is not sensitive to this kind of abnormal behavior. In fact, we should consider these behaviors as abnormal ones, or at least suspicious ones.

Meanwhile, there are a very small number of scans detected by TRW but not HRAID. They are caused by some scenarios as follows. The attackers scan multiple hosts, and for different hosts, they scan at different ports. When aggregated by either {`SIP, DIP`} or {`SIP, Dport`}, the number of scans is relatively small and less than our threshold. But TRW aggregates the scan by source IP and counts the number of unique destination IP's for detection, and thus will detect such behavior. We are not aware of any worms using such scans for propagation. It seems more like a combination of multiple small scans. Since the number of scans to each port and the number of scans to each host are small, the overall effect of such scans is still relatively small, *i.e.*, not a major attack. It is part of our future work to further investigate such problems.

Next, we compared our method with CPM for SYN flooding attack detection. The results are shown in Table VIII. Note that CPM detection is based on overall traffic, but not on each flow, so it can only report that for each interval whether there is an attack or not. For a fair comparison, we also count the number of intervals that SYN flooding detected by HRAID spans on (not the real number of attacks) and include them in Table VIII.

In the LBL traces, there is no SYN flooding, but a very large number of scans. As discussed before, CPM cannot differentiate them, so it treats all the scans as SYN flooding and has very high false positives. On the other hand, CPM and HRAID have very similar results for the NU data because the port scans are mixed with SYN flooding for

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

TABLE VII
HORIZONTAL SCANS DETECTION COMPARISON OF HRAID AND TRW AGGREGATED BY SOURCE IP.

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

TABLE VIII
TCP SYN FLOODING DETECTION COMPARISON OF HRAID AND CPM.

each interval, and thus the port scans do not cause any false positives for CPM. Meanwhile, there is a small number of intervals in which SYN flooding is buried in the rest of the normal traffic, which have large numbers of normal SYN and SYN/ACK packets from the traffic of other hosts, and thus for the total traffic $\frac{SYN-SYN/ACK}{SYN/ACK} < 1$, meaning CPM cannot detect them.

D. Detected Intrusions Successfully Validated

In the previous section, we compare HRAID with other statistical detection methods. But, it does not show whether the attacks detected are the real ones. In this section we manually examine a certain number of attacks for validation.

1) SYN Flooding: We validate our SYN flooding detection results with backscatter [14] because the Anderson-Darling A2 uniformity test in backscatter is a very strong one and can hardly reach the significance level 0.05 if they are not uniformly distributed. However, backscatter has one limitation: it only checks the reply traffic from the victim, and assumes that each attack packet is replied by the victim, which is not valid in many cases (observed from our netflow data as well). Furthermore, our evaluation is to detect attacks in the incoming traffic. Thus, we tweak backscatter a bit, to check the incoming SYN traffic, and test the uniform distribution of the source IPs if they are all towards the same destination IP and the total number of such source IPs are large enough. Among the 32 SYN floodings detected, there are 21 matched with backscatter results. For the other 11 attacks, three have uniformity values very close to the threshold, Anderson-Darling A2 test with significance level 0.05 [14]. Another three have IP's spoofed in some regular manner. For instance, one has the IP address spoofed as: 1.0.0.*, 1.0.1.*, 1.0.255.*, 2.0.0.*, *etc.*. But the distribution of such addresses does not satisfy the uniformity test. The remaining five attacks are hard to validate with only packet headers. But nowadays many DoS attacks are launched with botnet using un-spoofed IP's because ISPs tend to drop randomly spoofed packets through ingress filtering.

2) Horizontal Scans: We manually validate horizontal scans, in particular, the top 10 and bottom 10 attacks in terms of their change difference. Table IX shows the top 10 horizontal scans for the NU experiment, whose SYN-SYN/ACK change count ranges from 24,000 to 60,000. Table X shows the bottom 10 horizontal scans, whose SYN-SYN/ACK change count ranges from 60 to 64. Table XI and Table XII demonstrate the top 10 and bottom 10 horizontal scans for the LBL trace. Through these evaluation table, we find that they are all possible attacks, including Nachi or MSBlast worm, SQLSnake worms, W32.Sasser, SSH scans, *etc.*. There are even some unknown worm scans detected by us, and also confirmed in the Dshield [45], the largest worldwide intrusion log repository.

3) Vertical Scans: We also manually validate vertical scans for both the LBL trace and the NU experiment. In the LBL trace, we found one vertical scan. It scanned some well-known service ports, such as HTTPS(81), HTTP-

Anonymized SIP	Dport	#unique DIP	Cause
204.10.110.38	1433	56275	SQLSnake scan
5.4.247.103	1433	54788	SQLSnake scan
15.38.124.150	1433	46586	SQLSnake scan
5.64.27.226	1433	45981	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
15.82.184.106	4899	19794	Rahack worm
5.55.96.69	1433	19217	SQLSnake scan

TABLE IX
TOP 10 OF HORIZONTAL SCANS IN NU EXPERIMENT.

Anonymized SIP	Dport	#unique DIP	Cause
98.198.251.168	135	64	Nachi or MSBlast worm
3.66.52.227	445	64	Sasser and Korgo worm
20.128.114.246	445	64	Sasser and Korgo worm
2.0.28.90	139	64	NetBIOS scan
91.115.92.212	445	64	Sasser and Korgo worm
98.198.0.101	135	64	Nachi or MSBlast worm
162.8.171.169	139	62	NetBIOS scan
82.203.230.86	135	62	Nachi or MSBlast worm
165.5.42.10	5554	62	Sasser worm
189.75.216.218	139	62	NetBIOS scan

TABLE X
BOTTOM 10 OF HORIZONTAL SCAN IN NU EXPERIMENT.

Anonymized SIP	Dport	#unique DIP	Cause
352841	1433	16014	SQLSnake scan
251293	21	10360	FTP scan
199707	80	7893	HTTP scan
295205	80	7536	HTTP scan
346376	80	6980	HTTP scan
347333	80	6935	HTTP scan
312475	21	5892	FTP scan
447394	1433	5875	SQLSnake scan
285091	4000	5775	SkyDance worm
227176	554	5660	RTSCP scan

TABLE XI
TOP 10 HORIZONTAL SCANS IN THE LBL TRACE.

Anonymized SIP	Dport	#unique DIP	Cause
191136	135	67	Nachi or MSBlast worm
4698	135	67	Nachi or MSBlast worm
432940	135	66	Nachi or MSBlast worm
219400	135	66	Nachi or MSBlast worm
253855	135	66	Nachi or MSBlast worm
445247	135	66	Nachi or MSBlast worm
81248	135	66	Nachi or MSBlast worm
218603	135	64	Nachi or MSBlast worm
98802	135	64	Nachi or MSBlast worm
208650	139	26	NetBIOS scan

TABLE XII
BOTTOM 10 HORIZONTAL SCANS IN THE LBL TRACE.

Proxy(8000,8001,8081). In the NU experiment, we found in total 19 vertical scans. We manually checked the top 5 and bottom 5 vertical scans in terms of the ports they scanned. We found the vertical scans are mostly interested in the well known service ports and Trojan/BackDoor ports. For the well know services ports, most of them scanned: HTTP(80), FTP(21), RTSP(554), Cache(3128) *etc.*. Moreover, most attacks are also interested in Trojan/BackDoor ports. For example, WinHole(808,1082), Slapper(1978), SubSeven 2.1(7000), Tini(7777), OpwinTRojan(10000), *etc.*. All these characteristics drive us to believe that these vertical scans are real scans.

E. Evaluation Results for Online Performance Constraints

1) Small Memory Consumption: It is very important to have small memory consumption for online traffic recording over high-speed links, to make use of the fast SRAM and for potential implementation in specialized hardware *e.g.*, FPGA or ASIC. In our experiments, we only use a total memory of 13.2MB for traffic recording. Note that such settings works well for a large range of link speeds, as we tried on different network traces. Sketch can report heavy changes and heavy hitters with the statistical accuracy bounds given in [20, 34].

On the other hand, if hash tables are used to record every flow, much larger memory is required as shown in Table XIII. 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 (and maybe even different destination IP) for each packet. For the method without sketch, it needs at least three hash tables corresponding to the three reversible sketches

Methods	2.5Gbps			10Gbps		
	1sec	1min	5min	1sec	1min	5min
HRAID with sketch	`13.2M`			`13.2M`		
HRAID with complete info	`171.875M`	`10.3G`	`51.6G`	`687.5M`	`41.25G`	`206G`
TRW	`93.75M`	`5.63G`	`28G`	`375M`	`22.5G`	`112.5G`

TABLE XIII
MEMORY COMPARISON(BYTES).

in our detection methods. For each packet, at least a new entry will be added to both the $\{\texttt{SIP,DIP}\}$ table and the $\{\texttt{SIP,Dport}\}$ table. Every entry of the hash table needs 6 to 8 bytes to store the key (*e.g.*, SIP and DIP) and another 4 bytes to store the value. For example, when the bandwidth is 2.5Gbps, for one second, the total memory consumption is $(2.5G/(8 \times 40)) \times (12 + 10)$=171.875MB. For TRW, for each $\{\texttt{SIP,DIP}\}$ pair, it maintains a likelihood ratio which is updated based on observed SYN and SYN/ACK. Thus, for the example above, the total memory consumption is $(2.5G/(8 \times 40)) \times$ 12=93.75MB. Thus, both of the methods will run out of memory very quickly in these scenarios.

2) Small Memory Access per Packet for Online Monitoring: There are 15 memory accesses per packet for 48 bit reversible sketches and 16 per packet for 64-bit reversible sketches (see [21] for details). For each two-dimensional sketche, 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 HRAID system has a very small number of memory accesses per packet and is capable of online monitoring.

3) Traffic Monitoring and Intrusion Detection with High Speeds: The HRAID system is composed of the three reversible sketches and two 2D sketches. The speed of 2D sketches is much faster than that of the reversible sketches. Thus, the speed is dominated by the latter. For a real HRAID system, we will implement it in hardware so that multiple sketches can be updated in parallel. 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. With a Pentium Xeon 3.2 GHz machine with normal DRAM memory, we record 239M items (1.8TB, 1-day NUIT data) 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. If we update the three reversible sketches serially, we can still archive 3.8M insertions/sec.

For the on-site NU experiments covering a total of 1430 minutes, HRAID 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.

VIII. Conclusion

In this paper, we propose a High-speed Router-based Anomaly and Intrusion Detection system, HRAID, leveraging the recent work on sketches. Experiments with several router traces show that HRAID is highly accurate, efficient, uses very small memory, and can effectively detect multiple types of attacks simultaneously.

References

[1] E. Mars and J. D. Jansky, "Email defense industry statistics," `http://www.mxlogic.com/PDFs/IndustryStats.2.28.04.pdf`.

[2] T. Ryutov, C. Neuman, D. Kim, and L. Zhou, "Integrated access control and intrusion detection for web servers," *IEEE Trans. on Parallel and Distributed Systems*, vol. 14, no. 9, pp. 841–850, 2003.

[3] S. Hofmeyr and S. Forrest, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, 1998.

[4] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[5] Marty Roesch, "Snort: The lightweight network intrusion detection system," 2001, `http://www.snort.org/`.

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

[7] S. Staniford et al., "How to own the Internet in your spare time," in *Proceedings of the 11th USENIX Security Symposium*, 2002.

[8] S. Staniford, D. Moore, V. Paxson, and N. Weaver, "The top speed of flash worms," in *Proc. of ACM CCS WORM Workshop*, 2004.

[9] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "Large scale malicious code: A research agenda," Tech. Rep. DARPA-sponsored report, 2003.

[10] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage, "Internet quarantine: Requirements for containing self-propagating code," in *Proceedings of the IEEE Infocom*, 2003.

[11] S. Sikka and G. Varghese, "Memory-efficient state lookups with fast updates," in *Proc. of ACM SIGCOMM*, 2000.

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

[13] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proc. of ACM SIGCOMM*, 2002.

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

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

[16] H. Wang, D. Zhang, and K. G. Shin, "Change-point monitoring for detection of DoS attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 4, 2004.

[17] P. Barford et al., "A signal analysis of network traffic anomalies," in *ACM SIGCOMM IMC*, November 2002.

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

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

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

[21] R. Schweller, Y. Chen, E. Parsons, A. Gupta, G. Memik, and Y. Zhang, "Reverse hashing for sketch-based one-pass change detection for high-speed networks," Tech. Rep. NWU-CS-2004-45, Northwestern University, 2004.

[22] Arbor Networks, "Intelligent Network Management with Peakflow Traffic," `http://www.arbornetworks.com/download.php` (registration required).

[23] Arbor Networks, "Service Provider Infrastructure Security: Detecting, Tracing, and Mitigating Network-Wide Anomalies with Peakflow DoS," `http://www.arbornetworks.com/download.php` (registration required).

[24] Symantec Inc., "Symantec ManHunt," `http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=%156&EID=0`.

[25] N. Duffield, C. Lund, and M. Thorup, "Properties and prediction of flow statistics from sampled packet streams," in *Proc. of ACM SIGCOMM Internet Measurement Workshop (IMW)*, 2002.

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

[27] E. Messmer, "Switches taking on new security roles," `http://www.nwfusion.com/news/2004/0614switchsecurity.html?docid=2441`.

[28] N. Weaver et al., "Very fast containment of scanning worms," in *USENIX Security Symposium*, 2004.

[29] 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.

[30] Christophe Diot Anukool Lakhina, Mark Crovella, "Mining anomalies using traffic feature distributions," in *Proc. of ACM SIGCOMM*, 2005.

[31] Michalis Faloutsos Thomas Karagiannis, Dina Papagiannaki, "Blinc: Multilevel traffic classification in the dark," in *Proc. of ACM SIGCOMM*, 2005.

[32] Supratik Bhattacharya Kuai Xu, Zhi-Li Zhang, "Profiling internet backbone traffic: Behavior models and applications," in *Proc. of ACM SIGCOMM*, 2005.

[33] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin. J. Strauss, "QuickSAND: Quick summary and analysis of network data," Tech. Rep. 2001-43, DIMACS, 2001.

[34] 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.
[35] J. D. Brutlag, “Aberrant behavior detection in time series for network service monitoring,” in *Proc. of the USNEIX Systems Administration Conference (LISA)*, 2000.
[36] CERT Coordination Center, “Current scanning activity,” `http://www.cert.org/current/scanning.html`, 2004.
[37] Vinod et al., “Internet intrusions: Global characteristics and prevalence,” in *Proc. of ACM SIGMETRICS*, 2003.
[38] S. Staniford et al., “Practical automated detection of stealthy portscans,” *Journal of Computer Security*, vol. 10, no. 1-2, 2002.
[39] Insecure.org, “Idle scanning and related ipid games,” 2004, `http://www.insecure.org/nmap/idlescan.html`.
[40] J. Jung et al., “Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites,” in *Proc. of WWW*, 2002.
[41] D. Moore et al., “Inferring Internet denial-of-service activity,” in *Proceedings of the USENIX Security Symposium*, 2001.
[42] Checkpoint Software Technologies, “TCP Flooding Attack and Firewall-1 SYNDefender,” `http://checkpoint.com/press/1996/synattack.html`.
[43] D.J. Bernstein, “SYN Cookies,” http://cr.yp.to/syncookies.html.
[44] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attacks and defense mechanisms,” in *ACM CCR*, April 2004.
[45] SANS Institute, “Dshield.org: Distributed intrusion detection system,” `http://www.dshield.org`.