Real-Time Feature Extraction for High Speed Networks

David Nguyen, Gokhan Memik, Seda Ogrenci Memik, and Alok Choudhary

Department of Electrical and Computer Engineering
Northwestern University
Evanston, IL 60201
{dnguyen, memik, seda, choudhar}@ece.northwestern.edu

Abstract

With the onset of Gigabit networks, current generation networking components will soon be insufficient for numerous reasons: most notably because existing methods cannot support high performance demands. Feature extraction (or flow monitoring), an essential component in anomaly detection, summarizes network behavior from a packet stream. This information is fed into intrusion detection methods such as association rule mining, outlier analysis, and classification algorithms in order to characterize network behavior. However, current feature extraction methods based on per-flow analysis are expensive, not scalable, and thus prohibitive for large-scale networks. In this paper, we propose an accurate and scalable Feature Extraction Module (FEM) based on sketches. We present the details of the FEM design on an FPGA and show that using FPGAs we can achieve significantly better performance compared to existing software and ASIC implementations. Specifically, the optimal FEM configuration achieves 20.18 Gbps throughput and 97.61% accuracy.

1 Introduction

Traditionally, intrusion detection techniques fall into two categories: signature (or misuse) detection or anomaly detection. Signature detection looks to find well-known patterns of attacks and intrusions by searching for pre-classified signatures either in network traffic or data patterns. Since general-purpose processors (i.e., software implementations) cannot meet the required performance limitations, numerous projects have investigated dedicated hardware (including reconfigurable hardware) for such tasks [1-4, 8, 17].

Anomaly detection, which is designed to capture behavior that deviates from the norm, is the counterpart to signature detection. These systems "predict" anomalous behavior. Hence, they can detect new/unknown intrusions. However, they suffer from false alarms (false positives) and also not sounding alarms when attacks do occur (false negatives) [19]. Since the number of new attacks is increasing and variations of old attacks are more prevalent, next generation IDSs must employ anomaly detection.

Regardless of the underlying algorithm, the first step in anomaly detection is the real-time network feature extraction. Due to the complexity of gathering detailed information in high speed links, existing techniques only monitor a small amount of features in a packet stream, limiting their effectiveness. Feature extraction mines more information than is readily available at the packet level. Besides the packet payload, a single packet does not offer much information. Yet by processing a series of packets, one can mine for additional characteristics of the network activity between hosts. Our architecture allows for characterization of a sudden increase network activity. This information is needed for anomaly detection algorithms such as rule mining, classification, and outlier detection.

In this paper, we propose a Feature Extraction Module (*FEM*). FEM accurately characterizes network behavior and always provides an up-to-date view of the network

environment. Depending on the application utilizing this information (e.g., rule mining, classification), different properties get monitored in the network. As we will describe in the following sections, our architecture can be easily configured to gather such different types of information. By utilizing the reconfigurable capabilities of FPGAs, these changes can be effectively performed. Because of such configuration and high performance requirements, FPGAs are an ideal implementation medium for their reconfigurability and inherent parallelism. Our simulation results prove the sketch data structure a viable alternative to expensive per-flow methods. In addition, our FEM implementation requires a constant amount of memory and achieves a guaranteed performance level, important characteristics for networking hardware design.

This paper is organized as follows. Section 2 presents a background of feature extraction measures and an introduction the types of attacks plaguing many networks. Section 3 presents the FEM architecture and its components. Section 4 demonstrates the applicability of our architecture. Simulations and FPGA implementation are diagrammed in Section 5. Then related work is shown in Section 6 with conclusions in Section 7.

2 Background

Our architecture is a necessary precursor for online anomaly detection. Although FEM can be configured to gather information for any anomaly detection scheme, in this paper we focus on detecting two of the most popular network attacks: denial-of-service (DoS) attacks and port scanning, a mechanism in worm propagation.

A majority of DoS attacks are SYN floods, which send connection requests faster than a machine can process them. The well-known TCP 3-way handshake is below.

1. Client sends a **SYN** segment with client's ISN (Initial Sequence Number).
2. Server responds with a **SYN-ACK** segment, which is Server's SYN segment with its own ISN and an acknowledgement to Client's SYN with Client's ISN+1.
3. Client sends **ACK** with server's ISN+1.

Initially, the attacker creates a random source IP for each packet with the SYN flag set to request a new connection. The victim responds with a packet having the SYN and ACK flags set and then waits for a confirmation packet, which will never arrive. Typically connection tables wait a period of time before dropping the entry. In this time, the victim is bombarded with SYN requests and the table fills up. As a result, the victim refuses any additional connection requests even if they are legitimate. Typical SYN flood behavior involves a large amount of packets with the SYN flag set directed at a victim. In section 4, we explain how the FEM architecture is configured to watch for this behavior.

Port scanning, on the other hand, is probably the most common and versatile type of intrusion mechanism. For example, with worm propagation, in order to distribute copies of itself, the worm must find other hosts vulnerable to it. Worms may target a specific host or search for any number of hosts. We classify three well-known port scan methods: vertical scan, horizontal scan, and block scan [11, 18]. Figure 1 is a visual abstraction of these port scanning methodologies. Horizontal scans are the most common, scanning a range of IPs on a particular port. The port number is often unique as it reflects the susceptibility the worm is exploiting. Vertical scans target a specific host and search for open ports on that host. The third scan type, block scan, is a combination of horizontal and vertical scans for different ports and machines. Fortunately, port scanning requires a real source IP address instead of a spoofed one.

Therefore, it is possible to track port scan behavior from the source IP address. Taking this into account, section 4 details how FEM will trace this type of activity.

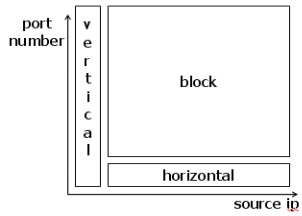

Figure 1: Port Scanning Methods

3 FEM architecture

In this section, we introduce the feature extraction module (FEM), which characterizes network behavior within an interval of time or specified interval of connections. Network behavior represented by the FEM sufficiently reflects the current state of the network. Thus, real-time profile of the network is always available for processing with intrusion detection schemes such as data mining, outlier analysis, statistical methods, etc [10].

The architecture's data storage component models the idea of sketches [13], which are used in data stream modeling for summarizing large amounts of information requiring a small constant amount of memory. Sketches are a probabilistic summary technique for analyzing large network streams without keeping per-flow state that make vector projections onto other sketches to infer additional information [13]. Our case study will show how the relationships between sketches aid in inferring additional network characteristics that are not explicitly monitored. To achieve fast execution and to achieve effective adaptation, we implement our architecture on an FPGA. The regular structure of sketches maps well onto an FPGA. We exploit the inherent parallelism in the sketch to increase throughput and obtain significant link speeds. Following is an explanation of the core component of FEM.

It is possible to model anomalous behavior associated with two general types of intrusions: time-based and connection-based. Time-based attacks cause an increase in network activity in a period of time, referred to as "bursty attacks." SYN floods are an example, where connection tables are flooded in a period of time disabling the victim machine to service new connection requests. Connection-based attacks do not have a recognizable temporal aspect. They are sometimes referred to as "pulsing zombie attacks." Port scans may release connection requests in the span of seconds or days. Therefore, intrusion detection methods focusing on large volumes of network activity are ineffective. Our architecture can capture both connection and time-based statistics.

3.1 Feature Extraction Functions

There are two main functions supported by the FEM:

- UPDATE (k, v) to change the value in the sketch
- ESTIMATE (k) to correctly retrieve a value from the sketch

Both functions take in a key k, which is input to H hash functions in the feature sketch. The key k, in this case, is any combination of the 5-tuple fields present in TCP/IP packet headers: {source IP, destination IP, source port, destination port, protocol}. The 6-bit flag field, also in a packet header, assists the control logic for intelligent hashing of the 5-tuple fields depending on what network characteristics are analyzed.

3.2 Architecture

Figure 2 highlights our architecture, consisting of a comprehensive feature controller (FC), hash functions (HF), feature sketch (FS), and a data aggregate (DA). The FEM architecture provides a fast, scalable, and accurate platform from which important network characteristics can be monitored and tracked in real-time. FEM can be configured to monitor a plethora of network characteristics by using the semantics of the TCP/IP protocol. Also, FEM requires a small memory footprint while maintaining a high level of accuracy, making it an attractive alternative to expensive per-flow methods.

The feature controller (FC) coordinates the inputs to the hash functions using the flags of a packet header. The reconfigurable aspects of FPGAs make reprogramming possible to monitor a variety of network statistics. Our case study in Section 4 focuses on open connection requests originating from or incoming to hosts by utilizing the SYN and ACK flags. Other possible statistics include the number of live connections, the flow size of active connections, amount of service-related traffic, or connection based statistics such the number of connections for specific services on a host. These measures would utilize the PSH (push), RST (reset), FIN (finish), and URG (urgent) flags.

For instance, a feature sketch monitoring web traffic at a particular host would use the source IP and destination port fields. Port 80 is designated as the port for http. Other destination ports such as 20 or 21 are designated for FTP traffic and 23 for telnet services. However, each FS monitors only one network characteristic. By using multiple FSs along with the relationships between FSs, we can infer additional network behavior information.

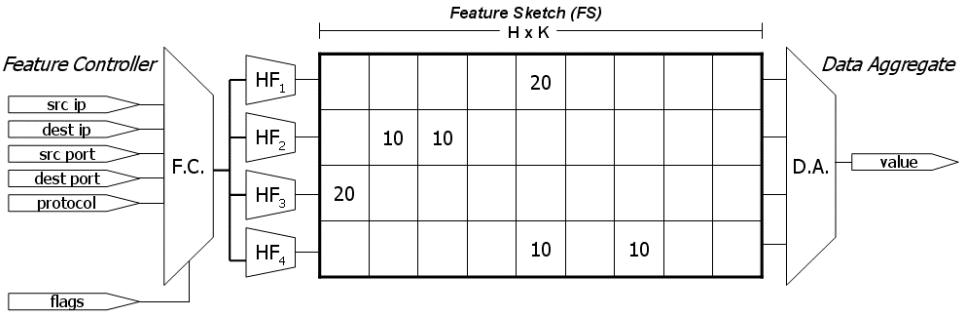

Figure 2. Feature Extraction Module with one FS

The feature sketch (FS) is an application of sketches used for data stream modeling. It uses a constant amount of memory and has constant per-record update and reconstruction cost. Each row in the FS is accessed in parallel with different hash functions. This favors FPGAs versus expensive per-flow methods. An FS contains H rows each of length K. When H>1, the accuracy of ESTIMATE queries improves. Section 5 presents the accuracy results.

This increased accuracy is achieved by addressing each row in the FS with a different hash function (HF). This way, the distribution of information varies for each row. We chose the Jenkins Hash for its speed and provable scatter properties. It is implemented in various Linux kernels as a component to the IPtables connection tracking module [9, 15]. With an FPGA, all hash functions are computed in parallel. Also, by pipelining the Jenkins Hash, FEM can accept a packet on every clock cycle, thus increasing throughput.

Lastly, the data aggregate (DA) component takes H values and estimates the actual value for a query. Using statistical estimation techniques, we show that ESTIMATE queries to the

FS are accurate. The heuristic we implement to estimate the value of a query takes the minimum of the H values in the FS. The minimum value suffers the least from collisions. Other estimation techniques are plausible [10, 13] but we found the minimum estimate usually gives the best results and the least hardware complexity. Minimum comparisons are performed in parallel such that this module is not on the critical path of FEM.

4 Case Study: Edge Router Level Application

In this section, we present an application of FEM at the router level for characterizing network behavior. Figure 3 is a simple diagram of network traffic occurring at any two nodes A and B. Node A represents outgoing traffic. The figure depicts different types of incoming traffic to node B through different ports. Port scans and SYN floods access any range of ports.

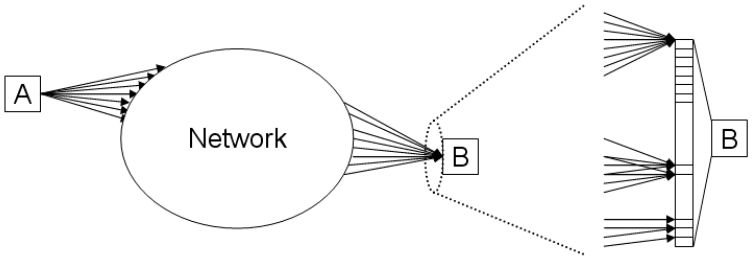

Figure 3. Incoming and Outgoing Packets

If the FEM is placed at the host level, for example at A, the architecture is simple. Each node is aware of its location when processing network packets so the feature controller FC easily preserves connection ordering. However, when placing FEM at a router, additional logic is needed to preserve connection ordering. For example, when A and B communicate with each other, the source IP/port and destination IP/port fields in a packet are not consistent with the particular node which started the connection.

This case study illustrates how to apply FEM to monitor network activity usually associated with SYN flood and port scans from a router's perspective. Each FEM consists of a number of FSs. For each FS, the key is denoted K and the feature value is denoted V. The source IP is designated SIP, destination IP DIP, source port SPORT, destination port DPORT, and protocol, PROTO. The flags applicable for this case study are the SYN and ACK flags**.**

We want to track the behavior associated with these two attacks.

First, it is known that SYN flood traffic is directed at a (DIP, DPORT) combination. Port scans are more flexible and use any combination of (DIP, DPORT). With an array of FSs, network behavior can be characterized for any given window of packets in a network stream. To monitor the behaviors of port scans and SYN floods, we propose the setup in Figure 4.

Four FSs are accessed and updated in parallel with a stream of packets. Each FS monitors a different network characteristic. Our architecture favors FPGA implementation since the feature controller can be reprogrammed and easily placed back into the network without any modification to the core architecture. Section 5.2 details the FPGA implementation and performance of a FEM module with one FS. Because multiple FSs are accessed in parallel, the width of the FEM has a minor impact on performance.

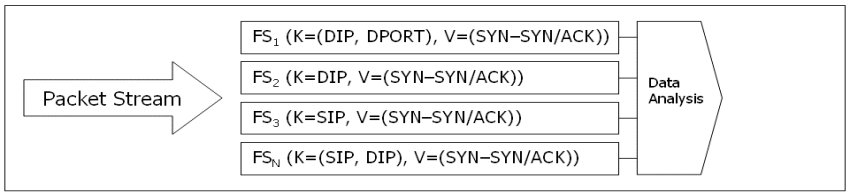

Figure 4. Case Study Example

FS_1 aids in SYN flood detection by monitoring the number of un-serviced SYN requests for specific services. When a machine services SYN requests, it responds with a packet having the SYN and ACK flags set. For a SYN packet, a count value is incremented. For a SYN/ACK response the count is decremented. By placing FS_1 at an edge router, connection ordering relative to the DIP is easily preserved by checking the flags in the packet. All connections in FS_1 are candidates for SYN floods and we denote this set $SYNFLOOD_{set}$.

FS_2 is monitors hosts with a large number of partially completed SYN requests. This activity indicates vertical scans or SYN floods. Notice FS_2 is a superset of FS_1. FS_2 contains all types of traffic at a particular IP. By querying both FSs with ESTIMATE, we can approximate the percentage of types of traffic at any DIP. Removing $SYNFLOOD_{set}$ from FS_2 leaves candidates for vertical scans, $VSCAN_{set}$.

FS_3 observes the traffic from any SIP that causes incomplete SYN requests. This measure includes vertical, horizontal, and block scans. To differentiate this activity, FS_N is implemented to oversee the amount of traffic between any two hosts.

For a $SIP_x \in FS_N$, if there is a $DIP_x \in VSCAN_{set}$ and FS_3 returns a value greater than a threshold (pre-determined by other intrusion detection algorithms), we claim SIP_x is vertically scanning DIP_x. If not, SIP_x may be horizontally or block scanning on the network. Using both FS_3 and FS_N, we are able characterize additional network behavior.

The main difference between each FS is how the FC coordinates addressing each FS. As described, the flags SYN and ACK are used to intelligent configure each FEM. Nonetheless our architecture is general enough to measure other network characteristics. Using SYN/FIN relationships for opening and closing network connections, it is possible keep an FS updated with traffic flow sizes.

FEM can be employed at both the edge routers or on specific hosts. Our example contains extra logic for router implementation (connection ordering). Host implementation would actually be simpler because the perspective of network traffic is narrower.

5 Results

5.1 Simulations

In this section, we investigate the accuracy of using feature sketches by testing different FS sizes. There are no known benchmarks specifically compiled for feature extraction, so we arbitrarily chose six days of traces from the 1999 DARPA Intrusion Detection Evaluation [14]. Half of the traces contain labeled attacks and the other half do not. Nonetheless, FS should accurately represent the network environment.

We simulate a FS (K= (SIP, DIP, DPORT, SPORT), K= (SYN – SYN/ACK)). Our test FS is more intensive because more connections are simultaneously being tracked. By virtue of design, FS is constantly updating; so we stream in 24 hours of network activity and query the FS afterwards to compare the FS estimate with exact per-flow results.

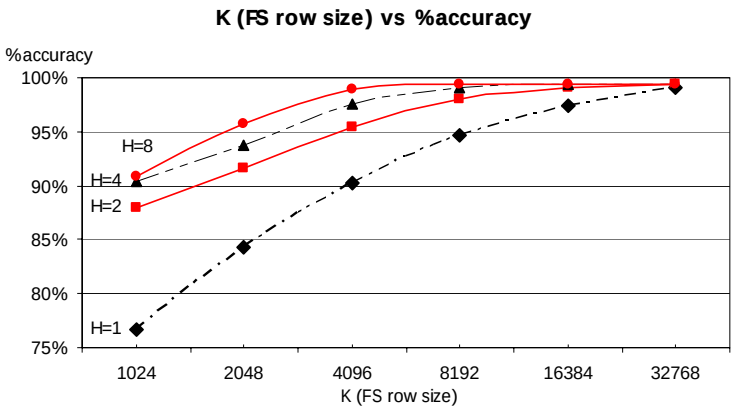

Figure 5. K(FS row size) vs. Accuracy

Figure 5 presents the accuracy of using a FS. H represents the number of rows in the FS and K represents the size of each row. The accuracy is measured as the percentage of precisely estimated flows (i.e., where the estimated value is equal to the actual value) out of all flows in the DARPA traces. The results of all six days are averaged together. For multiple hash function results (H > 1), we use the Jenkins Hash with different seed values.

When keeping K constant and increasing H, the accuracy also improves. For example, with H=1, K=2048, the accuracy is 84.3%. With H=2, K=1024, the accuracy increases to 87.8%. The 3.4% difference equates to 5586 more precisely estimated flows of the total 164,276 flows. However, in most cases increasing K boosts accuracy more than increasing H. This is attributed to hash function limitations, such as poor scattering or lack of variability between different hash functions, or unavoidable collisions in small row size K (ex. H=8, K=1024)).

Table 1 represents an example of this behavior. The accuracy improves when increasing the number of rows until H=8, at which point the small K value limits the accuracy. Overall, however, the FS data structure ably satisfies accuracy demands. In Section 5.2, we investigate how increasing H changes throughput and FPGA performance.

H	K	Accuracy
1	16384	97.4238%
2	8192	97.9699%
4	4096	97.6010%
8	2048	95.6835%

Table 1. Constant Total K = 16384 entries

Figure 6 reports another measure of the effectiveness of feature sketches, the average deviation of estimations from exact per-flow results. Clearly, increasing H improves estimation of, in this case, SYN–SYN/ACK values. This trend persists for other network behavior measures. As in Figure 5, the gap between H=1 and H=2 is the largest. It shows that our datasets result in mostly 2 collisions. This fact favors more balanced FS configurations versus a one row FS where collisions adversely affect the accuracy.

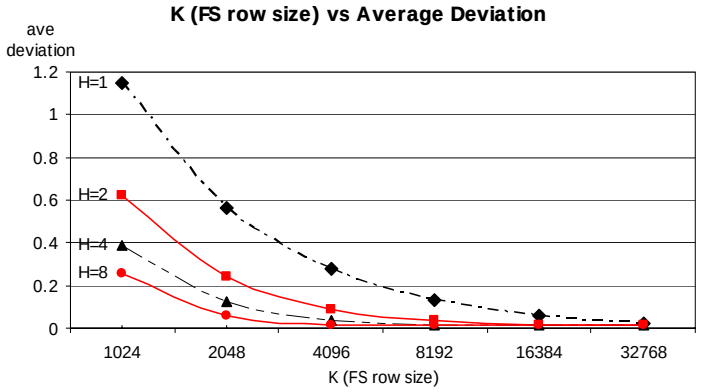

Figure 6. K (FS row size) vs. Average Deviation

5.2 Implementation Details

FEM was implemented on a Xilinx VirtexII xc2v1000 chip. This member of the Virtex II family contains 5120 slices and 40 16Kb Block RAM modules. We used Synplify Pro 7.2.1 for logic synthesis and the Xilinx ISE 5.2i suite for placement and routing. For our hash function, the Jenkins Hash was extensively pipelined to operate at 270.6 MHz.

FEM total K = 8192 entries				
H	**K**	**slices**	**Frequency (MHz)**	**Throughput (Gbps)**
1	8192	628	167.5	18.42
2	4096	1263	202.6	22.29
4	2048	2543	216.6	23.82
FEM total K = 16384 entries				
1	16384	634	169.3	18.62
2	8192	1265	190.1	20.99
4	4096	2543	183.4	20.18
FEM total K = 32768 entries				
1	32768	643	113.6	12.50
2	16384	1274	135.4	14.89
4	8192	2543	152.3	16.76

Table 2. Router-based FEM Place and Route Results

Table 2 contains the performance and area metrics for FEM implemented for edge routers. The performance results are similar for host-level implementation since the added logic in the feature controller (FC) is not on the critical path of the FEM. We test configurations for H=1, 2, and 4. Throughput, clock frequency, and slices are reported for three overall row sizes K=8192, 16384, and 32768. The throughput value is calculated from the 5-tuple data {source IP, destination IP, source port, destination port, protocol} and the 6-bit flag field used to configure the FC.

It is clear that for a given total memory size, increasing H increases throughput because it reduces the memory size and hence reduces the access times. Similarly, for a constant H, reducing the total memory amount (K) also increases the throughput. Among the simulated configurations, the best throughput of 23.81 Gbps is achieved for H=4 and K=2048. However,

note that this configuration has a relatively low accuracy of 94.1%. Hence, when one considers the "accuracy * throughput" product, the best configuration is H=4 and K=4096, which can extract information at 20.18 Gbps.

Note that the increase in number of slices is mostly a result of using multiple hash functions in parallel. Replicating the hash functions allows higher throughput and frequency at the expense of area. If there are area constraints, however, one could use one hash function implementation for multiple FS rows, providing the values to each of them at consecutive cycles. This would result in decreased throughput but also reduced area requirement. Since the Jenkins Hash is pipelined, mapping a hash function to multiple rows would not introduce long extra delays.

In conclusion, the simulations show that feature sketches are effective data structures for network behavior characterization. The simulation results demonstrate the gains in accuracy and estimation ability of feature sketches. FPGAs take advantage of multiple FS rows to satisfy Gigabit throughput demands. Consequently feature sketches, the main components of FEM, are attractive data structures for FPGAs to exploit parallelism.

6 Related Work

Many networking applications have found their way into hardware implementations [7]. With link speeds increasing and the multitude of network applications, future solutions place a premium on both performance and flexibility. FPGAs qualify for both these requirements. Current generation of FPGAs can operate at speeds ranging from 50 MHz to 250 MHz and have capacity on par with large ASIC designs. For example, FPGAs have been used in developing platforms for experimentation of active networks [4] for services such as detection of Denial-of-Service (DoS) attacks, real-time load balancing for e-commerce servers, real-time network based speed recognition servers for v-commerce, etc. Also, high speed front-end filters and security management applications for ATM firewalls have found their way onto FPGAs to reduce performance penalties at the IP level [12].

As for flow monitors, TCP/IP Splitter [17] has been implemented as part of the FPX (Field-Programmable Port Extender) project to perform flow classification, checksums, and packet routing. However, this implementation is limited to 3 Gbps monitoring. In our previous work, we implemented a flow size monitor similar to FEM [16]. However, the design was not updateable when connections were completed. This limitation prevented achieving an accurate representation of the network. In this paper, we modify the architecture for update information, which increases the accuracy by almost an order of magnitude for comparable configurations. Other studies [6] agree that per-flow methods will not suffice and propose both intelligent algorithms and multistage filters using multistage hash tables to increase accuracy over Cisco's NetFlow (which uses sampling to characterize network traffic).

Jupiter T-Series routers also implement a propriety flow monitoring mechanism. Although the router runs at 10 Gbps link speeds, the monitoring is limited to 250K packets per second for each physical interface card. It is also limited in the maximum number of flows (400K) and flow creation rate (12K new sessions per second). Before these dedicated hardware solutions, flow monitoring tools had been implemented in software, such as HTTPDUMP.

7 Conclusions

Real-time feature extraction is a core component for any intrusion detection system that claims to be truly real-time. Signature detection can be done live, but live anomaly detection

requires a comprehensive picture of the network environment. Our feature extraction module provides this functionality using feature sketches, which map well onto reconfigurable hardware. We took advantage of pipelining and inherent parallelism in FPGAs to increase throughput. Many network behavior parameters can be monitored using our architecture by making small modifications to the design. These characteristics include flow size, number of open connections, number of un-serviced connection requests, etc. The novelty of our design lies in modifying a single FS row into multiple FS rows to increase the accuracy up to 97.61%, reduce the estimation error to an average of 0.0365 packets, and achieve throughputs up to 20.18 Gbps for a 16K entry FEM.

References

[1] Michael Attig, Sarang Dharmapurikar, and John Lockwood. Implementation Results of Bloom Filters for String Matching. In Proceedings of: IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, CA, April 20-23, 2004.

[2] Z. Baker and V. Prasanna. Automatic Synthesis of Effcient Intrusion Detection Systems on FPGAs. In Proceedings of FPL'04, 2004.

[3] Z. Baker and V. Prasanna. Time and Area Effcient Pattern Matching on FPGAs. In Proc. of FPGA '04. 2004.

[4] Dollas, A. et al., Architecture and Applications of PLATO, a Reconfigurable Active Network Platform. 1999, Department of ECE, Technical University of Crete: Greece.

[5] Sarang Dharmapurikar, Praveen Krishnamurthy, Todd Sproull, John W. Lockwood. Deep Packet Inspection Using Parallel Bloom Filters. Symposium on High Performance Interconnects (HotI), Stanford, CA, USA, pp. 44-51, Aug. 20-22, 2003.

[6] Estan, C., and Varghese, G. New directions in traffic measurement and accounting. In Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (2002).

[7] Hauck, S., The roles of FPGAs in reprogrammable systems. IEEE, Apr.1998, 86: p. 615--638.

[8] B. L. Hutchings, R. Franklin, and D. Carver. Assisting Network Intrusion Detection with Recon¯gurable Hardware. In Proceedings of IEEE FCCM'02, 2002.

[9] B. Jenkins, Hash Functions and Block Ciphers, http://www.burtleburtle.net/bob/hash/index.html

[10] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch based change detection: methods, evaluation, and applications," in Proc. of ACM SIGCOMM IMC, 2003.

[11] Lazarevic, A., et. al: A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection, Proc. of 3rd SIAM Conference on Data Mining, San Francisco, May. 2003.

[12] McHenry, J. T., P. W. Dowd, F. A. Pellegrino, T. M. Carrozzi, and W. B. Cocks. An FPGA-based coprocessor for ATM firewalls. In IEEE Symposium on FCCM, April 1997. Napa Valley, CA.

[13] S. Muthukrishnan. Data streams: Algorithms and applications, 2003. Manuscript based on invited talk from 14th SODA. Available from http://www.cs.rutgers.edu/~muthu/stream-1-1.ps.

[14] MIT Lincoln Laboratory – Darpa Intrusion Detection Evaluation. http://www.ll.mit.edu/IST/ideval/

[15] NetFilter/IPtables: Firewalling, NAT and packet mangling for Linux 2.4. http://www.netfilter.org/

[16] D. Nguyen, J. Zambreno, and G. Memik. Flow Monitoring in High-Speed Networks with 2D Hash Tables. In Proc. the 14th Annual International Conference on FPL, 2004.

[17] David V. Schuehler, John W. Lockwood. TCP Splitter: A TCP/IP Flow Monitor in Reconfigurable Hardware. Proceedings of Hot Interconnects 10 (HotI-10), Stanford, CA, August 2002, pp. 127-131.

[18] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," Journalof Computer Security, vol. 10, no. 1-2, 2002.

[19] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich, "Internet intrusions: Global characteristics and prevalence," In Proceedings of ACM SIGMETRICS, 2003.