

SIFT: Snort Intrusion Filter for TCP *

Michael Attig and John Lockwood
Department of Computer Science and Engineering
Washington University in St. Louis
Saint Louis, MO 63130
Email: {mea1, lockwood}@ar1.wustl.edu

Abstract

Intrusion rule processing in reconfigurable hardware enables intrusion detection and prevention services to run at multi Gigabit/second rates. High-level intrusion rules mapped directly into hardware separate malicious content from benign content in network traffic. Hardware parallelism allows intrusion systems to scale to support fast network links, such as OC-192 and 10 Gbps Ethernet.

In this paper, a Snort Intrusion Filter for TCP (SIFT) is presented that operates as a preprocessor to prevent benign traffic from being inspected by an intrusion monitor running Snort. Snort is a popular open-source rule-processing intrusion system. SIFT selectively forwards IP packets that contain questionable headers or defined signatures to a PC where complete rule processing is performed. SIFT alleviates the need for most network traffic from being inspected by software.

Statistics, like how many packets match rules, are used to optimize rule processing systems. SIFT has been implemented and tested in FPGA hardware and used to process Internet traffic from a campus Internet backbone with live data.

1. Introduction

Network intrusion detection systems (NIDS) enable real-time detection of network attacks. Open-source tools, like Snort [16], allow network administrators to compose and apply intrusion rules that monitor and protect their network. Intrusion rules can specify the processing of packet headers, the matching of patterns in packet payloads, and the action to take when a rule matches. A typical Snort rule is given below.

```
alert tcp any 110 → any any (msg:"Virus -  
Possible MyRomeo Worm"; flow:established;
```

*This work was supported by a grant from Global Velocity.
<http://www.globalvelocity.com>

```
content:"I Love You"; classtype:misc-activity;  
sid:726; rev:6;)
```

The first portion of the rule indicates that the packet header can match a wildcard value for the source IP address, the destination IP address, and the destination port. However, the source port must have the value of 110. The rule also specifies that the protocol should be TCP. The second portion of the rule specifies to search for the string "I Love You" over an established TCP/IP connection. Flow reconstruction must be performed to detect a signature segmented across multiple TCP packets.

The Snort Intrusion Filter for TCP (SIFT) operates as a preprocessor to reduce the processing load on an intrusion PC running Snort. IP packets without keywords or questionable packet headers need not be processed by software. Packets containing matching rule criteria are forwarded to software for complete rule processing.

1.1 Motivation

Intrusion rule databases expand as new threats emerge. However, as the number of headers and signatures to match increases, the CPU on a PC running Snort becomes overloaded, and not all of the packets are processed. To provide complete protection, the intrusion system must process all packets.

Further, the complexity of rules defined is outpacing the ability of hardware designers to map these rules to hardware in a scalable way. As a result, software tools still provide more flexibility to process complex features, albeit at slower speed. By accelerating some aspects of Snort rule processing in hardware, the load on software can be limited, allowing the IDS to operate on higher speed links.

Schaelicke et. al showed that Snort inadequately acts as a sensor on higher speed links [17]. Even on a dual Pentium-4 Xeon running at 2.4 GHz, the system could only support up to 543 rules with no packets loss. Furthermore, only two of the authors' test systems could support saturated 100

Mbps links. This is troublesome because Gigabit links are common today.

Lee et. al described how coverage, cost, and resilience to system attacks must be considered when deploying an intrusion detection system [9]. They used Snort with a subset of the available rules to show how it can be overloaded. In their experiments, Snort was overloaded when traffic exceeded 40 Mbps. During the periods of packet loss, Snort was only able to find 10% of generated attacks.

Kruegel et. al proposed a complex set of multiple sensors to provide intrusion detection services on high-speed network links [8]. Their approach partitions traffic into small slices, each processed by a particular sensor. This requires the use of numerous sensors to implement full functionality.

Most network traffic does not need to be processed by software at all. SIFT supports most Snort rules, provides the capability to expand to support future rules, and scales for use on high-speed network links. By limiting the amount of traffic to be processed by software, far more data can be processed with less equipment. Along with a single PC, SIFT implements a complete IDS system.

1.2 Features

The SIFT architecture has a number of features to provide intrusion detection and prevention services:

- Support for case-insensitive string matching across TCP packet boundaries
- Support for up to 5000 signatures
- Ability to scan payloads for 5 different signature lengths between 2 and 32 bytes
- Ability to reconfigure hardware to change the amount of hardware allocated to scan for different signature lengths
- Support for header rules to be processed in parallel
- Throughput of 2.5 Gbps on the FPX platform [11]
- Ability to achieve total throughput of 20 Gbps using a Virtex 4 FPGA

1.3 Contributions

This work makes several contributions. First, a signature detector that operates on TCP streams was developed. SIFT supports up to 5000 signatures, each of which can be loaded into the system via commands sent over a network. Second, traffic analysis of real traffic on an university network was obtained. Intuition on what assumptions can be made and what to expect when designing rule processing elements was gained. Over 85% of the traffic monitored was

benign, containing no matching search criteria from version 2.2 of the Snort rule database. This suggests that general purpose processors with traffic off-loaders could be used today to monitor networks far more efficiently than software systems alone.

2 Related Work

There are three main aspects of Snort rule processing in reconfigurable hardware: TCP flow reconstruction, packet header classification, and payload string matching.

2.1 Flow Reconstruction

Flow reconstruction orders TCP stream data so that signatures can be discovered even when they are not contained in a single packet. Multi-context TCP processors allow signatures to be detected in multiple flows that are interleaved. Nguyen, Zambreno, and Memik created hardware-based flow monitors to process very high packet rates for a limited number of TCP flows [15]. Necker, Contis, and Schimmel implemented a single TCP-stream assembler in FPGA technology capable of operating at 3.2 Gbps [14]. Li, Torresen, and Soraasen implemented a state-based inspection technique for eight TCP flows in FPGAs to be used as an add-on to Snort systems [10].

Schuehler developed a TCP Processor as a protocol processing wrapper implemented in FPGA logic that annotates control information onto incoming IP packets [18, 19]. The TCP Processor is capable of simultaneously tracking state for eight million TCP flows while operating at 2.9 Gbps on the FPX platform. The circuits can operate at rates of over 10 Gbps by using faster FPGA devices. SIFT leverages this technology.

2.2 Packet Classification

Packet classification circuits inspect the fields inside packet headers, including the IP addresses, protocol, and port fields. This information is used to determine what intrusion rules match.

Yu and Katz used ternary content addressable memories (TCAMs) to return multiple matching packet headers for rule processing applications [23]. Gokhale et. al performed header processing and content matching using content addressable memories (CAMs) [7]. Lockwood et. al created a reconfigurable firewall that performed header processing using TCAMs implemented in FPGA hardware [12].

2.3 String Matching

String matching is the most complex aspect of rule processing in today's intrusion detection systems. In general,

strings can appear anywhere within the payload of a packet or TCP/IP traffic flow. Most techniques developed to date require re-synthesis of FPGA logic to add new signatures, operate on a packet rather than flow basis, and cannot hold the entire Snort signature database in a single FPGA.

Sidhu and Prasanna mapped nondeterministic finite automata (NFA) into FPGA logic [20]. Clark and Schimmel reduced the redundant logic generated with NFAs [5]. Moscola et. al automatically generated deterministic finite automata (DFA) optimized with JLex, a lexical analyzer that optimizes the number of states needed to detect regular expressions [13]. Sugawara, Inaba, and Hiraki implemented a string-matching method using trie-based table look-ups [22]. Their system supported a limited number of TCP streams.

Baker and Prasanna developed a technique to partition signature databases into independent pipelines to more efficiently utilize FPGA resources by reducing redundancy [2]. Cho, Navab, and Mangione-Smith created automated techniques to generate highly parallel comparator structures [4]. Sourdis and Pnevmatikatos created unique VHDL instances for each signature to process so signatures can be added or removed only by changing the instance [21].

3 Architecture

3.1 Overview

After analyzing version 2.2 of the Snort rule database, several observations were made regarding how to separate harmful traffic from safe traffic. There are 292 unique header rules, 2,107 fixed-length signatures, and 233 regular expressions. A fixed-length signature is a unique sequence of bytes while a regular expression looks for a pattern of bytes that can contain wildcards. Of the 292 header rules, 168 do not have a signature associated with them. Thus, for all but 168 of the 2,464 rules, signature detection alone can be used as SIFT criteria. If one of the 2,107 fixed-length signatures is detected, the data is forwarded to software for further examination. Similarly, if one of the 168 “header-only” rules match, it is also forwarded to software for further inspection. Rules that contain regular expressions also contain separate fixed-length signatures. In this implementation of the system, the fixed-length signatures detected in rules are used as SIFT criteria.

SIFT is configured to monitor traffic from a network tap, as shown in Figure 1. IP traffic passes through a Gigabit Ethernet line card to a TCP Processor [19], where TCP traffic flows are reconstructed. From the TCP Processor, the IP traffic passes to a Port Tracker [18], where statistics are obtained. The IP traffic then passes to SIFT to scan for Snort signatures and header rules. IP traffic containing matching

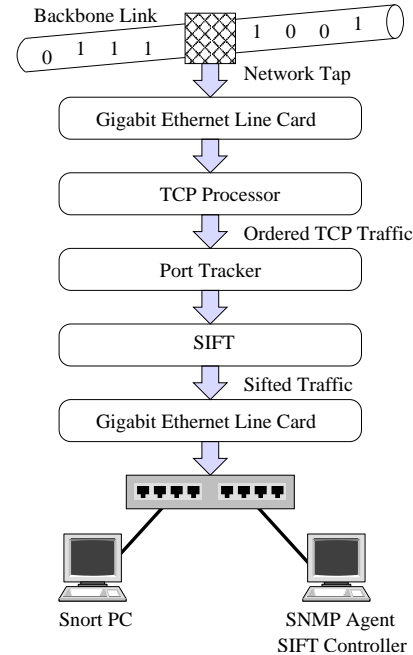


Figure 1. IP traffic flows from a network tap through the SIFT system to a Snort PC. Only IP traffic that could match a Snort rule is passed to software.

Snort header-only rules or signatures is sent from SIFT to a PC running Snort software.

3.2 SIFT

A block diagram of SIFT is shown in Figure 2. TCP data enters the system via the TCP deserialize wrapper [18]. Control signals mark specific locations in the packet, such as the beginning and ending of TCP headers. This data is sent to a header check component to determine if the packet matches a header-only rule. The payload is sent through an 8-stage word pipeline where each byte offset is searched for signatures by separate Bloom filters [1]. If a match is detected, the match decoder determines the string ID. This ID is sent to the action retriever to determine the action that should be taken on the packet. The default case is to pass the packet along to software. If no criteria matches, the packet is not inspected further. Once a questionable packet fully arrives, it is sent to software using the outbound side of the TCP deserialize wrapper.

The system is incrementally programmable. Signatures can be added or deleted via User Datagram Protocol (UDP) control packets that are sent through the communication wrapper (shown at the top of Figure 2) to a control FSM. New signatures are then forwarded to a Quad Bloom Filter

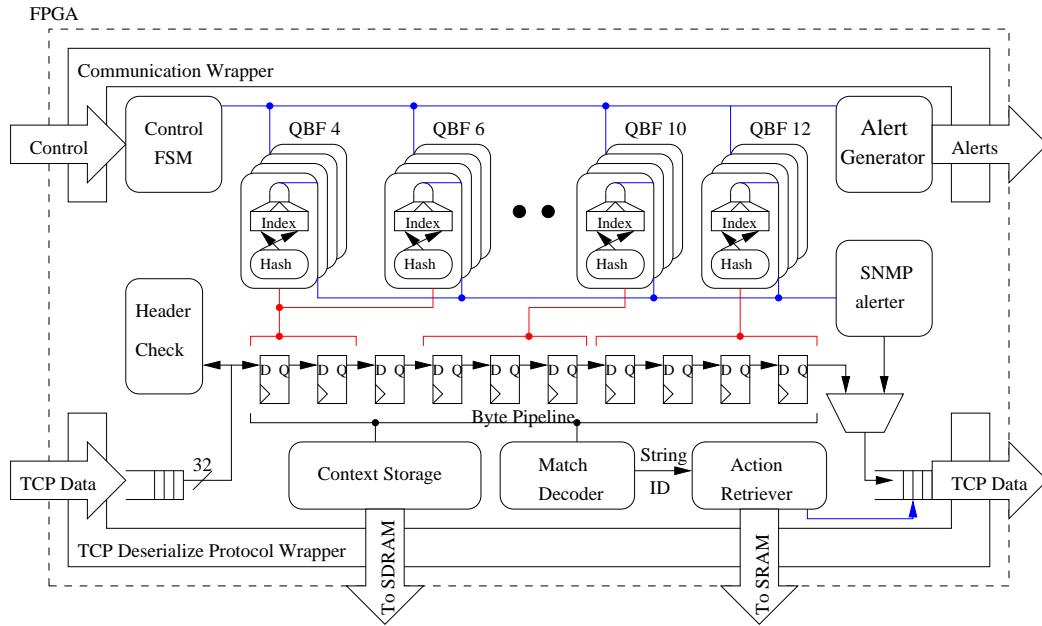


Figure 2. SIFT uses Bloom filters for string matching and custom logic for header classification. External memory is used to store flow context. System statistics are periodically reported out of the system to control software.

(QBF). An alert generator sends alert messages to a software controller if a match occurs. Statistics are sent out of the system periodically via a SNMP alerter.

To accommodate string matching across TCP packets, a context storage component holds the state of the last 32 bytes of the current TCP flow being scanned. When a different flow enters, the new flow’s previous 32 bytes are loaded from SDRAM, and the current flow’s last 32 bytes of data are placed in SDRAM. This allows a history for every TCP flow to be maintained.

3.3 Quad Bloom Filters

In order to scan for strings at a high data rate, it is necessary to process multiple different strings in each clock cycle. A Quad Bloom Filter (QBF) instantiates four Bloom filters for this purpose [3]. The Bloom filter designed for this system searches for signatures of a pre-defined length by computing eight hash functions that index a 16 Kbit vector. If all eight hashed locations are set, the input signature matches with high probability. Input strings of a fixed length plus three additional bytes are passed to each QBF in order to scan four offsets of the signature from the pipeline. By using QBFs, SIFT can process four bytes of data per clock cycle.

3.4 Header Check

A header check component instantiates logic blocks for the header-only rules based on protocol: TCP, UDP, ICMP, and other IP. Most of the 168 header-only rules are used to check for specific TCP/UDP port numbers or protocol-specific features like a sequence number. Each of the 168 header-only rules are checked in parallel, and a rule match is declared if any of the headers match.

3.5 Context Storage

A context storage unit interfaces with one SDRAM to retrieve and store the pipeline state. When a new packet arrives, the flow ID associated with the packet is used to retrieve 32 bytes of flow context from SDRAM. Flow context is fetched while the previous packet is processed, preventing the pipeline from stalling. Using a 64 MB SDRAM, context for two million simultaneous TCP flows can be tracked.

3.6 Action Retriever

An action retriever interacts with SRAM to determine the programmable action to perform when a signature matches. Each signature has a 16-bit ID number. This component uses the signature ID to look up the action to take. Actions can be performed to generate an alert, to drop a

packet, or to return the data in a packet to a monitoring host for storage.

3.7 SNMP Alerter

Hardware and software tools have been assembled to track traffic statistics. The SNMP (Simple Network Management Protocol) alerter periodically reports statistics to a software-based SNMP agent. The SNMP agent is queried by MRTG (multi router traffic grapher) to graphically display statistics. SIFT tracks 18 events that include the number of total TCP packets, sifted TCP packets, matching headers, string matches, and dropped packets.

4 Results

4.1 Implementation Results

The implementation of SIFT in a Xilinx Virtex XCV2000E-8 FPGA utilized 84% of the logic slices, 58% of the look-up tables, and 96% of the block RAMs. SIFT operated at 80 MHz, providing a throughput of 2.5 Gbps for all packet sizes by processing 32 bits of data per cycle.

4.2 Testing Configuration

SIFT was configured to use QBFs to scan for 4, 6, 8, 10, and 12 byte signatures. Signatures beyond 12 bytes occur infrequently, so it was deemed appropriate to truncate them in the hardware scanner. By truncating signatures, SIFT errs on the side of caution by sending slightly more traffic than necessary to software for further inspection.

Both 4 and 5 byte signatures are stored in the 4 byte engine, both 6 and 7 byte signatures are stored in the 6 byte engine, and so on. Table 1 summarizes how signatures from Snort version 2.2 were loaded into the hardware. Non-unique signatures refers to the truncated signatures that were already programmed into a scanner. The QBF that scanned for 12 byte signatures had the highest false positive probability because it held the most strings [6]. As implemented, the system supported 92% of Snort signatures. The remaining signatures were not supported because SIFT did not scan for 1, 2, or 3 byte signatures. After analyzing the 206 rules with signatures no longer than 3 bytes, all but 18 had specific header rules that could be mapped into the header check component.

SIFT was tested with live traffic to monitor all traffic entering or leaving Washington University's 19,000-node campus network. It was programmed to process data using the configuration of Figure 1. Two Gigabit Ethernet line cards and three FPX cards were used to implement the system. Traffic was passed from a Gigabit Ethernet line card

Table 1. A summary of the statistics for each Bloom filter engine.

Scan Range	Signatures Loaded	Non-unique Signatures	False Positive Probability
4-5	262	18	4.3E-8
6-7	163	2	1.2E-9
8-9	227	4	1.5E-8
10-11	244	2	2.5E-8
12+	872	137	2.1E-4

to another FPX with the TCP Processor, where stream re-assembly was performed. TCP traffic was then sent to a third FPX that implemented a Port Tracker [18]. Finally, the traffic was passed to SIFT, where content scanning and header processing occurred. Sifted traffic was passed to a PC running Snort via a second Gigabit Ethernet line card. The campus tap provided a high-volume of network traffic that varied with time. Network throughput peaked near 350 Mbps in the mornings and afternoons. The PC that ran the Snort software was an AMD Athlon MP 2600+ with 3 GB of RAM running Red Hat Linux.

4.3 Observations

To illustrate the type of results collected, a 30-second average number of matching header-only rules per second transmitted through the network throughout the day on February 15th, 2005 is shown as the solid line in Figure 3. As can be seen, the number of matching headers significantly drops off near mid-day, even though the number of packets traversing through the system increased at this time. The number of matching header-only rules ramps back up early in the afternoon, only to decline again until after midnight. This pattern repeated periodically on multiple days of observation.

The average number of matching 4-byte signatures per second is also plotted in Figure 3 as the dashed line. Although it might be expected that the number of matching signatures would scale with the traffic load, it was found to be invariant with the time of day. This was true for all Bloom engines. Also, the number of matches for each QBF engine is approximately the same, even the 12-byte QBF. This indicates that longer signatures do not occur frequently.

Figure 4 shows the average number of incoming and sifted TCP packets per second. As can be seen, SIFT reduced the amount of TCP traffic that needed to be examined by a Snort PC by 90% on average. The traffic reduction results based on protocol are summarized in Table 2.

Snort software running on the PC kept up with the average 40 Mbps of traffic that was passed to it from SIFT.

Table 2. A summary of the suspect packets sifted through the system to a Snort PC during the week of February 13-20, 2005.

Protocol	Total Packets	Suspect Packets	Reduction
TCP	28.2 K	2.71 K	90%
UDP	0.50 K	0.07 K	86%
ICMP	1.03 K	0.87 K	15%
IP	0.34 K	0.01 K	96%

An average of 5% of the sifted traffic resulted in completely matching Snort rules. This indicates that more traffic than necessary was sent to software. Of the matching rules, 78%, 16%, and 6% belonged to ICMP, TCP, and UDP packets, respectively. This is a curious fact since TCP traffic is the dominant protocol in use in the campus network. Our measurements indicated that only 0.5% of all traffic resulted in a rule match. Rule processing on benign traffic is a waste of software resources. By performing signature detection and header classification for header-only rules, SIFT eliminated 85% of traffic. Further degrees of filtering with SIFT can decrease the amount of traffic sent to software.

4.4 Device Performance

The circuits described in this paper were implemented with a Xilinx XCV2000E FPGA device. The relative improvements available by targeting the design to newer Xilinx FPGAs are shown in Figure 5. The critical resource for string matching with Bloom filters is the on-chip block RAM. The Virtex4 100 has 240 18-Kbit block RAMs, while the Virtex XCV2000E had only 160 4-Kbit block RAMs. By quadrupling the number of QBF engines used, 16 bytes can be processed per clock cycle in the Virtex4, as compared to 4 bytes in the Virtex XCV2000E. Additionally, the size of the bit-vector used by each Bloom filter engine increases to 72 Kbits from 16 Kbits. Thus, each Bloom engine can support 4500 signatures in the Virtex4 at the same false positive rate as 1000 signatures in the Virtex XCV2000E. The frequency improvement in the Virtex4 is 2x that of the Virtex XCV2000E. As a result, SIFT can support 4.5x as many signatures while operating at 20 Gbps.

5 Conclusion

In this paper, an architecture to preprocess network traffic destined for an intrusion monitor was presented. FPGA logic was used to perform header classification to determine if header-only Snort rules match, and string matching circuits were used to search for signatures in TCP flows. By

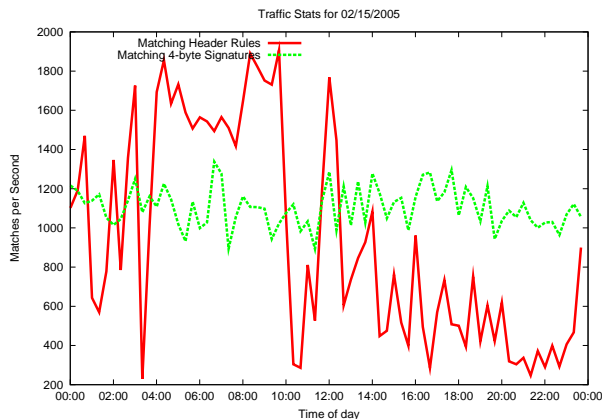


Figure 3. The number of matching header-only rules and matching 4-byte signatures per second versus time on Feb 15, 2005.

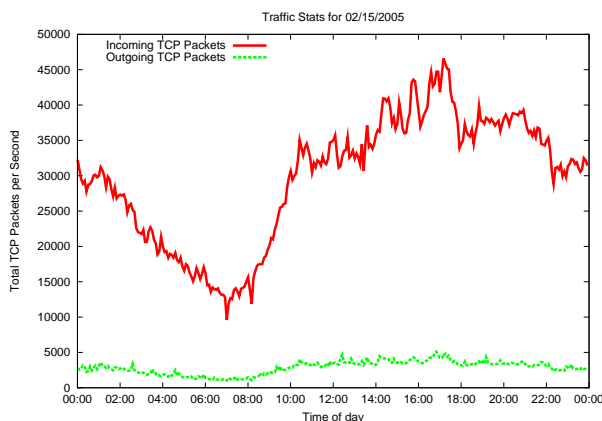


Figure 4. The number of incoming and sifted TCP packets per second versus time on Feb 15, 2005.

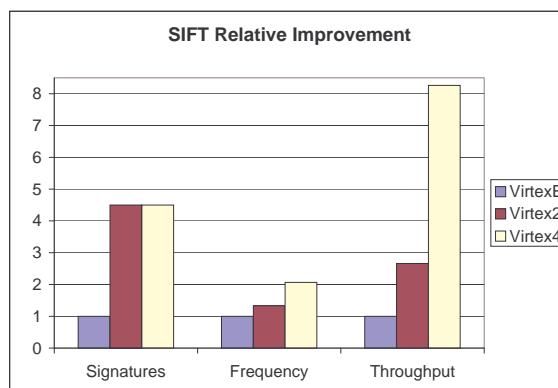


Figure 5. The relative improvements of the SIFT design using newer FPGAs.

using SIFT to preprocess traffic, the amount of traffic that needed to be inspected by a PC running Snort was reduced to 15% of all traffic. This offloaded the task of scanning non-matching content data from the PC.

Using live traffic monitored from a campus network, SIFT reduced the total amount of traffic forwarded to a PC running Snort to only 15% of the input load. Live traffic was processed by SIFT that consisted of all data passing into and out of the Washington University campus network - a large network that serves 19,000 hosts. By using SIFT to pre-filter traffic, a single PC running Snort software on the reduced traffic load was not overloaded. This type of traffic reduction provides an enormous benefit for intrusion monitors running on Gigabit links. By implementing the SIFT circuit on a Virtex 4 FPGA, SIFT could pre-filter traffic on a 20 Gbps link.

References

- [1] M. E. Attig, S. Dharmapurikar, and J. Lockwood. Implementation results of Bloom filters for string matching. In *IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM)*, Napa, CA, Apr. 2004.
- [2] Z. K. Baker and V. K. Prasanna. A methodology for synthesis of efficient intrusion detection systems on FPGAs. In *IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM)*, Napa, CA, Apr. 2004.
- [3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.
- [4] Y. Cho and W. Mangione-Smith. Deep packet filter with dedicated logic and read only memories. In *IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM)*, Napa, CA, Apr. 2004.
- [5] C. R. Clark and D. E. Schimmel. Scalable multi-pattern matching on high-speed networks. In *IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM)*, Napa, CA, Apr. 2004.
- [6] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. W. Lockwood. Deep packet inspection using parallel Bloom filters. In *Hot Interconnects*, pages 44–51, Stanford, CA, Aug. 2003.
- [7] M. Gokhale, D. Dubois, A. Dubois, M. Boorman, S. Poole, and V. Hogsett. GranidT: Towards gigabit rate network intrusion detection technology. In *12th Conference on Field Programmable Logic and Applications*, pages 404–413, Montpellier, France, 2002. Springer-Verlag.
- [8] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer. Stateful intrusion detection for high-speed networks. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 285. IEEE Computer Society, 2002.
- [9] W. Lee, J. B. Cabrera, A. Thomas, N. Balwalli, S. Saluja, and Y. Zhang. Performance adaptation in real-time intrusion detection systems. In *Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, Lecture Notes in Computer Science, Zurich, Switzerland, Oct. 2002. Springer-Verlag.
- [10] S. Li, J. Trresen, and O. Sraasen. Exploiting stateful inspection of network security in reconfigurable hardware. In *Field Programmable Logic and Applications (FPL)*, Lisbon, Portugal, Sept. 2003.
- [11] J. W. Lockwood, N. Naufel, J. S. Turner, and D. E. Taylor. Reprogrammable Network Packet Processing on the Field Programmable Port Extender (FPX). In *ACM International Symposium on Field Programmable Gate Arrays (FPGA'2001)*, pages 87–93, Monterey, CA, Feb. 2001.
- [12] J. W. Lockwood, C. Neely, C. Zuver, J. Moscola, S. Dharmapurikar, and D. Lim. An extensible, system-on-programmable-chip, content-aware Internet firewall. In *Field Programmable Logic and Applications (FPL)*, page 14B, Lisbon, Portugal, Sept. 2003.
- [13] J. Moscola, J. Lockwood, R. P. Loui, and M. Pachos. Implementation of a content-scanning module for an Internet firewall. In *FCCM*, Napa, CA, Apr. 2003.
- [14] M. Necker, D. Contis, and D. Schimmel. TCP-stream reassembly and state tracking in hardware. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Napa, CA, Apr. 2002.
- [15] D. Nguyen, J. Zambreno, and G. Memik. Flow monitoring in high-speed networks with 2D hash tables. In *Field Programmable Logic and Application: 14th International Conference, FPL 2004, Leuven, Belgium, August 30-September 1, 2004. Proceedings*, pages 1093–1097, Antwerp, Belgium, Aug. 2004. Springer-Verlag.
- [16] M. Roesch. SNORT - lightweight intrusion detection for networks. In *LISA '99: USENIX 13th Systems Administration Conference on System Administration*, pages 229–238, Seattle, Washington, Nov. 1999.
- [17] L. Schaelicke, T. Slabach, B. Moore, and C. Freeland. Characterizing the performance of network intrusion detection sensors. In *Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, Lecture Notes in Computer Science, Berlin-Heidelberg-New York, September 2003. Springer-Verlag.
- [18] D. V. Schuehler. Techniques for processing TCP/IP flow content in network switches at gigabit line rates. PhD dissertation, Washington University in St. Louis, 2004.
- [19] D. V. Schuehler and J. Lockwood. TCP-Splitter: A TCP/IP flow monitor in reconfigurable hardware. In *Hot Interconnects*, pages 127–131, Stanford, CA, Aug. 2002.
- [20] R. Sidhu and V. Prasanna. Fast regular expression matching using FPGAs. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Apr. 2001.
- [21] I. Sourdis and D. Pnevmatikatos. Pre-decoded CAMs for efficient and high-speed NIDS pattern matching. In *IEEE Symposium on Field-Programmable Custom Computing Machines, (FCCM)*, Napa, CA, Apr. 2004.
- [22] Y. Sugawara, M. Inaba, and K. Hiraki. Over 10gbps string matching mechanism for multi-stream packet scanning systems. In *Field Programmable Logic and Application: 14th International Conference, FPL 2004, Leuven, Belgium, August 30-September 1, 2004. Proceedings*, pages 484–493, Antwerp, Belgium, Aug. 2004. Springer-Verlag.
- [23] F. Yu and R. Katz. Efficient multi-match packet classification and lookup with TCAM. In *12th Annual Proceedings of IEEE Hot Interconnects*, Stanford, CA, Aug. 2004.