Novel Hardware-Based Approaches for Intrusion Detection

SLOBODAN BOJANIĆ[1], VLADIMIR MILOVANOVIĆ[2], ZORANA BANKOVIĆ[2], CARLOS CARERRAS[1] AND OCTAVIO NIETO-TALADRIZ[1]

[1]Department of Electronic Engineering
ETSI de Telecomunicación
Technical University of Madrid
Ciudad Universitaria, 28040 Madrid
SPAIN

[2]Department of Electronics
Faculty of Electrical Engineering
University of Belgrade
Bulevar Kralja Aleksandra 73, 11000 Beograd
SERBIA AND MONTENEGRO

http://www.die.upm.es

Abstract: In this paper we present novel approaches in hardware deployment for intrusion detection systems (IDS). As far as we know there are no published FPGA implementations of genetic programming and bioinformatics algorithms used for IDS. It is shown in the paper that the use of hardware can efficiently exploit the inherent parallelism of algorithms and reach Gigabit data processing rates that are required for current communications. Each processing unit can be replicated many times on deployed FPGA device and in dependence of the capacity of the device, almost proportionally increase the throughput performance.

Key-Words: security, intrusion detection, bioinformatics, genetic programming, FPGA, rule generation, pattern matching

1 Introduction

The move to hardware based systems allows the introduction of more parallelism than might be possible in software based systems and hence alternative algorithms. Hardware based solutions are probably the only approach currently practical for intrusion detection on high-speed backbone networks running at speeds of around 10 Gbps [1].

The paper elaborates the implementation and use of extra hardware in computer security The novel hardware implementation in FPGA devices gives new possibilities which can further lead to better results and more reliable security system. As dicussed in the paper, the results yields a promising future for these systems and overcome many speed drawbacks and disadvantages of already existing algorithms as well as platform for development the new, original ones.

In this work we tackle two intrusion detection problems: the masquerade detection and generation of more efficient intrusion detection rules that are based on bioinformatics and genetic programming approach respectively [2], [3]. As far as we know there is now published work on the hardware deployment for above mentioned cases.

The target technology for implementation is Filed Programmable Gate Arrays (FPGA) that are now feasible for a broad range of applications, including those required for intrusion detection systems. FPGAs have long been used for a number of integer and fixed-point applications, such as the signal processing applications. However, with the rapid advances in technology, current FPGAs contain much more configurable logic blocks (CLBs) than their predecessors. Some researchers have suggested that FPGAs have become highly competitive with microprocessors in both peak performance and sustained performance [4]. Besides high computing performance, the current FPGA fabrics also provide large amounts of on-chip and off-chip memory bandwidth to I/O-bound applications.

The use of Genetic Programming (GP) to detect unknown attacks is based on the belief that new rules will have better performance than initial ones based on known attacks [2]. Better performance means the new rules obtained after evolving the initial ones using GP will not only cover known attacks, but also possibly detect the novel ones.

Masquerade is a security attack in which an intruder assumes the identity of a legitimate user. The detection of the attack can be carried out using bioinformatics algorithm [3]. The Smith-Waterman algorithm [5] was originally applied in bioinformatics for purposes of gene alignment but efficient hardware implementation widen its use to other fields [6]. We demonstrated its applicability on the cryptanalysis of stream ciphers in [7].

The rest of the paper is organized as the following. Section 2 is dedicated to Genetic Programming approach with corresponding subsections about background on GP, its use for rule generation, FPGA implementation and obtained results. The Section 3 is devoted to bioinformatics approach with corresponding subsections on the Smith-Waterman algorithm, its application to intrusion detection, the

FPGA implementation and corresponding results. The conclusions are drawn in Section 4.

2 Genetic Programming Approach

In this Section the Genetic Programming approach for Intrusion Detection Systems is presented. It is based on the on the use of Genetic Programming for efficient generation of intrusion detection rules presented in [2]. Due to our hardware implementation the data processing performances are significantly increased.

2.1 Genetic Programming

Genetic Algorithm (GA) has been used in different ways in Intrusion Detection Systems (IDS). The different machine learning techniques, such as finite state machine, decision tree, and GA were used to generate artificial intelligence rules for IDS in [8]. An IDS was implemented using autonomous agents (security sensors) and applied AI techniques to evolve genetic algorithms. Agents are modeled as chromosomes and an internal evaluator is used inside every agent [9].

Genetic Programming is extension of Genetic Algorithm [10]. GP randomly generates an initial population of solutions. Then, the initial population is manipulated using various genetic operators to produce new populations. These operators include reproduction, crossover, mutation, dropping condition, etc. The whole process of evolving from one population to the next population is called a generation.

Fitness functions ensure that the evolution is toward optimization by calculating the fitness value for each individual in the population. The fitness value evaluates the performance of each individual in the population.

2.2 Rule generation

Initial rules are selected based on background knowledge from known attacks and can be represented as parse trees. GP will evolve these initial rules to generate new rules. New rules are used to detect novel or known attacks [2].

GP generates new rules in two phases. In the first step, temporary new rules are composed of new rules generated by four operators including mutation, reproduction, crossover, and dropping condition and additional rules directly generated from previous populations. Thus the number of temporary new rules is doubled.

An initial population of 40 rules was selected that cover a series of network-based attacks. The fitness value for each rule is calculated on the basis of the training dataset from DARPA Intrusion Detection Evaluation Program [11]. The dataset includes almost all known network-based attacks, namely *land*, *synfloodping of death* (*pod*), *smurf*, *teardrop*, *back*, *neptune*, *ispweep*, *portsweep*, and *UDPstorm* attacks.

Eleven parameters defined in DARPA dataset are used to describe the attacks in the training dataset. Table 1 describes these parameters and their meaning. The initial and new rules are composed of attribute descriptors.

Table 1. Representation of parameters

Parameters	*Meaning*
protocol_type	Type of protocol
Land	Flag to identify whether connection is from/to the same host/port
Wrong_fragment	Number of wrong fragments in the connection
synflood	Connections that have “SYN” errors
num_compromised	Number of compromised conditions
same_srv_rate	Percentage of connections to the same services
diff_srv_rate	Percentage of connections to the different services
count	Number of connections from the same source host to the same destination host
srv_count	Number of connections from the same source service to the same destination service
dst_host_count	Number of connections from the same destination host to the same source host
dst_host_srv_count	Number of connections from the same destination service to the same source service

The GP-based approach can detect smurf and UDPstorm attacks which are absent from the training dataset. The average false negative rate (FNR) for each rule is 5.04% and the average false positive rate

(FPR) is 5.23%. The average rate of detecting unknown attacks for each rule is 57.14% [2].

2.3 FPGA Implementation

In order to speed up the algorithm, the massively parallel model in hardware can be used [12]. The fine-grained (diffusion) model is presented like the one being most VLSI-friendly. It consists of a large number of independent processing nodes, connected through the X-net topology, that evolve a large number of small, overlapping subpopulations. The nodes are simple, regular and mainly use local communications. Every node has its own memory that has a linear machine code representation of the individuals and its own embedded CPU that executes that code (in this case does the pattern comparing) and the part that does GA.

One node consists of four major parts, a CPU, a memory, a control block and a simple router (Fig. 1).

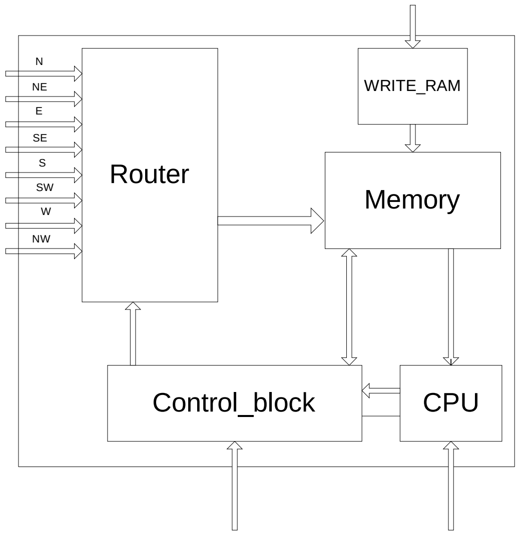

Fig. 1 Block scheme of the node

CPU consists of fast CAM (Content Addressable Memory) that contains the input-representation made according to the parameters given at the end. CAM performs comparing of one word from the memory that represents the terminal (meaning that it starts with “00”) with its contest and has the latency of one clock period to decide if there is matching or not. During the testing whether a connection is an intrusion or not, the input equal to “10000100” is following the connection.

The part that decides if the whole match is achieved consists of two shift registers. If the connection that represents intrusion is marked as intrusion by the node’s rule, the number of all detected intrusions is incremented by one, and similarly, if the connection that does not represent intrusion is marked as non-intrusion, the number of non-intrusions is incremented by one.

After the end of training data is signaled, the calculation of the fitness value is being performed according to the next formula:

The mutation randomly replaces one instruction with a randomly generated one, as described above. Control block also performs sending the whole code to another nodes if the request is issued. Genetic algorithm is not performed during the code sending.

2.3 Results and Discussion

The results are given in Table 2. It can be seen that a gigabit throughput is achieved. XST option for the forward retiming is set. Area needed for one neighborhood of 9 cells is 3208 slices while area needed for one cell 364 slices

Table 2. Performances on Xilinx Virtex-II FPGA

Speed grade	5	6
Clock frequency [MHz]	121	138
Throughput [Gbps]	0.97	1.1

Possible hardware improvements depend on the particular network speed. In the case of the high speed networks, it is possible to reconfigure design to take more than one byte at the same time, considering that the percentage of taken I/O blocks is low which would multiple the maximum throughput.

It is possible to improve genetic algorithm by adding new genetic operators that would accomplish in higher diversity of solutions and probably boost the possibility of detecting novel attacks. Considering the existing problem of ‘local maximum’ in genetic algorithm, design of router can be changed to take the highest fitness value as the result of the tournament with some predefined possibility.

3 Bioinformatics approach

In this section the bioinformatics approach in deployment of intrusion detection system is described. It is based on the use of Smith-Waterman algorithm for the detection of masquerading attack [3]. Since the algorithm can be significantly accelerated exploiting inherent parallelism we deployed hardware implementation to reach Gigabit rates that are required in contemporary data processing.

3.1 Masquerading attack

One of the most devastating attacks in computer security is the masquerading, in which an attacker assumes the identity of a legitimate user. Masquerade attacks often occur when an intruder obtains a legitimate user's password or when a user leaves their workstation unattended without any sort of locking mechanism in place. It is very difficult to discover this break of security because attacker appears to be a normal user with valid authority and privileges. The level of damage that can be performed via masquerader attacks (stolen/destroyed documents, data, e-mail) makes them one of the most serious threats to computer and network infrastructure.

To fulfill the task of detecting a masquerader, somehow we need to make a contrast between a real legitimate user and intruder. Every single user has its own specific behavior. If we make a so called user signature and compare the current behavior of logged user with it, in the case of legal user they should match well and in the case of an intruder they should differ. User signature usually contains a sequence of commands, but it could also contain a user style of typing on a keyboard or specific mouse movements, biometric features. There are many ways to create a user signature depending on how complex it is going to be or what type of operating system do we use. When the user signature is created it needs to be compared with current user behavior in a session. The algorithm checks for similarities between two command sequences and/or between styles of keyboard and mouse use. The problem is that the user behavior changes over time. In a small time period user can react in different manner and in long time periods user can change his behavior fundamentally. We can overcome this problem by updating a signature frequently.

As said before there have been numerous attempts to successfully detect masquerade attacks and to minimize false positives and negatives without degrading the quality of a user's session. In [13] various masquerade detection techniques were analyzed and performance were presented. At the moment, the best results are achieved with a bioinformatics approach that uses the slight modification of Smith-Waterman algorithm, originally published for the purpose of gene alignment.

3.2 Smith-Waterman algorithm

Sequence alignment is already well-studied tool used to quantify and visualize similarity between two or more sequences. It is originally developed for application of comparison of genetic material, such as DNA. Specifically DNA composes of only four elements, nucleotides: adenine (A), thymine (T), guanine (G) and cytosine (C).

The Smith-Waterman (SW) algorithm is a database search algorithm developed for use in bioinformatics, and based on an earlier model appropriately named Needelman-Wunsch after its original creators [14]. The SW algorithm implements a technique called dynamic programming, which takes alignments of any length, at any location, in any of two input sequences, and determines whether an optimal alignment can be found. Based on these calculations, scores or weights are assigned to each character-to-character comparison (positive for exact matches and substitutions, negative for insertions and deletions) and the highest scoring alignment is reported.

Simply, dynamic programming finds solutions to smaller pieces of the problem and then puts them all together to form a complete and optimal final solution to the entire problem. Because of its complexity, many heuristic methods were developed. Original SW algorithm is superior to the BLAST and FASTA algorithms because it searches a larger field of possibilities, making it a more sensitive technique, however, individual pair-wise comparisons between letters slows the process down significantly. Instead of looking at an entire sequence at once, the SW algorithm compares multi-length segments, looking for whichever segment maximizes the scoring measure. The algorithm itself is recursive in nature. It can be described by the following equation.

$$M_{ij} = \max\{(M_{i-1,j-1} + s_{ij}), (M_{i,j-1} + g_v), (M_{i-1,j} + g_h), 0\}$$

where: M_{ij} is a weight matrix element in *i*-th row and j-th column., s some positive reward for match interpreting similarity, g_h and g_v are usually negative horizontal and vertical gap penalties, respectively

For example, if we try to align the following two gene sequences using Smith-Waterman algorithm:

$$\text{Seq 1}: \{\text{C, A, G, C, C, T, C, G, C, T, T, A, G}\}$$
$$\text{Seq 2}: \{\text{A, A, T, G, C, C, A, T, T, G, A, C, G, G}\}$$

it will produce local alignment:

$$\begin{array}{ccccccc} \text{G} & \text{C} & \text{C} & - & \text{T} & \text{C} & \text{G} \\ \updownarrow & \updownarrow & \updownarrow & & \updownarrow & \times & \updownarrow \\ \text{G} & \text{C} & \text{C} & \text{A} & \text{T} & \text{T} & \text{G} \end{array}$$

It is very easy to expand and generalize this algorithm for use with any number of different elements and with any scoring function. By adjusting the scoring function the algorithm can be applied in many different fields, like masquerade detection.

Smith-Waterman algorithm is used as a detection algorithm in the following sense: user signature corresponds to one sequence and the test block corresponds to another. Matches should positively influence the score of an alignment, and should be chosen that matches are preferred to gaps. Mismatches are kept as a constant zero score. Using the above criteria, we chose scores of +1 for a match between two aligned commands, -2 for a gap placed in the tested block, -3 for a gap placed in the user's signature, and 0 for a mismatch between aligned commands.

The scores are taken from already mentioned work [3], but any other scoring can be applied without any degradation in performance. As the goal is to align characteristic groups of commands in a tested block with similar groups in the user's signature we want to heavily penalize any gaps within the signature itself, because we do not want commands in the tested block to be aligned with gaps in the user's signature.

3.3 FPGA Implementation and Results

The main drawback of the Smith-Waterman algorithm is its slowness. As the complexity of the algorithm was too high, many heuristic methods were developed; the best among them are FASTA and BLAST. Although the results were not bad, the sector of security requests great accuracy and they were not able to compete with original algorithm. The only way to reduce the complexity is bringing a parallelism in the calculations.

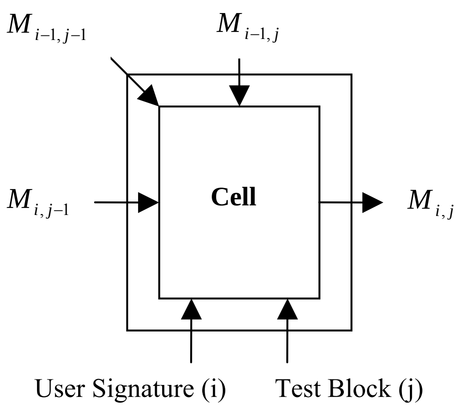

Fig. 2 Bioinformatics processing element

The parallelism can not be applied on general purpose machines where processor poses only one arithmetic-logic unit. But hardware devices like FPGA have capabilities to bring the necessary parallel calculations and reduce the complexity and set free other hardware parts that were used in processing user signature and test block.

The parallelization method for a matrix size $m \times n$ is realized using a systolic array of at least $\min\{m,n\}$ processing elements. Although user signature is by the rule almost always longer than the test block, it is better to assign number of elements that correspond to a signature's length for the reasons that will be later explained. This is a feasible solution having in mind area that this hardware occupies. The same processing element calculates distances in the column in which it is placed. This is shown on Figure 2.

Continuing this, it is not hard to conclude that whole matrix will be calculated in $m+n$ clock cycles giving this method a linear complexity of $O(m+n)$ (Fig. 3). These two properties: linear increase of time complexity and linear increase of area occupation when the length of sequences rise prevail in determination which method to choose.

0	1	2	3
1	$D_{1,1}$ $PE_{1(T1)}$	$D_{2,1}$ $PE_{2(T2)}$	$D_{3,1}$ $PE_{3(T3)}$
2	$D_{1,2}$ $PE_{1(T2)}$	$D_{2,2}$ $PE_{2(T3)}$	
3	$D_{1,3}$ $PE_{1(T3)}$		

Fig. 3 Parallelized matrix calculation

There is no need to load both, the user signature and the test block, in the device concurrently. While user is logging on, after the correct password is imported, the user signature is load to FPGA as a part of loading process that occurs until system is ready for work. This eliminates demands for storing user signatures in FPGA whose memory is often very small and can have other purposes but in some other memory like flash or hard disk drive. After session starts all the operations of interest are monitored and rewritten creating the test block suited for comparison with user signature. The test block is compared with user signature as the commands are inserted by the user and/or the specific movements of mouse and keyboard occurs giving that some sort of pipelining in the system, yielding the very fast detection, just after the last needed command for decision whether or not the legitimate user is logged to the system.

We implemented a general Smith-Waterman algorithm where the score can be any integer of fixed point decimal value. We used nowadays the most popular family of FPGA devices, Virtex 4 from

Xilinx. In structures like FPGAs the best way to gain a speed is to create highly optimized regular cells and then simply replicate them. We assumed that there is 256 used commands, which is in operating systems like, UNIX and Linux more than enough. We also assumed that the characteristic mouse movements and keyboard typing styles can be represented with eight bits each, although its expansion would not dramatically decrease performance. The design using given parameters was implemented in Xilinx Virtex 4 device with clock frequencies up to 250MHz. One cell takes 100 logic cells, what is 50 slices or 25 configuration logic blocks. This means that more than 2000 processing elements can be implemented in the greatest FPGA chip from Xilinx. Considering the bus width of eight bits the equivalent throughput is 250MB/s or 2Gb/s. These days de facto standard in computer network communications is 1Gb/s so our unit can be used without any slowness or delay. Existence of self equipped devices based on microprocessor and FPGA combining a network support gives the possibility of implementing a design independent from personal computer, a design that could monitor network independently and so free the server totally form this activity and what is also important to be placed somewhere else.

4 Conclusion

We presented two novel hardware architectures that are designed for intrusion detection systems. The genetic programming approach was applied for the generation of more efficient intrusion detection rules and bioinformatics Smith-Waterman algorithm was deployed to detect masquerading attack. The architectures were deployed in Virtex II and 4 FPGA devices respectively. Each hardware unit reaches Gigabit processing rates thus satisfying contemporary data flow requirements. Due to optimized area design the units can be massively replicated on FPGA devices and achieve significant throughput increase.

Acknowledgment:
The work was supported by Ministry of Science and Technology of Spain through the project TIC2003 - 09061-C03-02 and “Ramon y Cajal” program as well as EU Tempus project CD_JEP-17028-2002.

References:

[1] Tripp, G. An intrusion detection system for gigabit networks – architecture and an example system. *Technical Report 7-04*, Computing Laboratory, University of Kent, April 2004.

[2] Lu, W. and I. Traore. Detecting New Forms of Network Intrusion Using Genetic Programming, *Computational Intelligence,* Volume 20, Number 3, pp. 470-490, 2004.

[3] S. Coull, J. Branch, B. Szymanski, E. Breimer, Intrusion Detection: A Bioinformatics Approach, *Proc. 19th Annual Computer Security Applications Conference*, p. 24, 2003.

[4] Underwood, K. D. and K. S. Hemmert. Closing the gap: CPU and FPGA trends in sustainable floating-point BLAS performance. *Proc. 2004 IEEE Symposium on field-programmable custom computing machines (FCCM’04),* USA 2004.

[5] T. F. Smith, M. S. Waterman, Identification of common molecular subsequences, *Journal of molecular biology*, 147:195-197, 1981.

[6] G. Caffarena, S. Bojanić, J.A. López, C. Pedreira and O. Nieto-Taladriz, Parallel Computatuin of Gene Sequence Matching, *IADIS International Conference Applied Computing*, 2004.

[7] S. Bojanić, G. Caffarena, S. Petrović, O. Nieto-Taladriz, *FPGA for pseudorandom generator cryptanalysis*, to appear in Microprocessors and Microsystems, Elsevier.

[8] Sinclair, C., L. Pierce and S. Matzner. An Application of Machine Learning to Network Intrusion Detection. *Proc. 1999 Anual Computer Security Applications Conf. (ACSAC), pp. 371-377.* Phoenix, USA, Dec. 1999.

[9] Crosbie, M. and G. Spafford. Applying Genetic Programming to Intrusion Detection. Proc. 1995 AAAI Fall Symposium on Genetic Programming, pp. 1-8. Massachussets, USA, Nov. 1995.

[10] Koza, J. R. Genetic Programming, MIT Press, 1992.

[11] Lippmann, R. The 1999 DARPA off-line intrusion detection evaluation. Computer Networks, 34(4):579-595, 2000.

[12] Elkund, S. E. A Massively Parallel Architecture for Linear Machine Code Genetic Programming, Proc. 4th Inter. Conf. on Evolvable Systems: From Biology to Hardware, Lecture Notes In Computer Science, Vol. 2210, pp. 216–224, 2001.

[13] W. DuMouchel, W. H. Ju, A. F. Karr, M. Schonlau, M. Theusen and Y. Vardi, Computer Intrusion: Detectiong Masquerades, *Statistical Science*, 16, no. 1:58-74, 2001.

[14] S. B. Needleman and C. D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of molecular biology*, 48:443-453, 1970.