User Profiling in FPGA for Intrusion Detection Systems

Slobodan Bojanić1, Vukašin Pejović1, Gabriel Caffarena1, Vladimir Milovanović2, Carlos Carreras1 and Jelena Popović2

1Universidad Politécnica de Madrid, ETSI Telecomunicación, Ciudad universitaria s/n, 28040 Madrid, Spain {slobodan, vule, gabriel, carreras}@die.upm.es
2University of Belgrade, Faculty of electrical Engineering, Bul. Kralja Aleksandra 73, 11000 Belgrade, Serbia {vlada, jelena}@el.etf.bg.rs

Abstract: This work elaborates the implementation and the use of dedicated hardware in computer network security. The novel FPGA implementation for user profiling in intrusion detection systems, provides new potential which can further lead to better performances and more reliable security systems. The results yield 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.

Keywords: behavioural biometrics, user profiling, intrusion detection, pattern recognition, FPGA, Smith-Waterman algorithm.

1. Introduction

User Profiling in the field of Intrusion Detection consists of observing someone interacting with a computer, creating a model of such behaviour and using it as a template for what is considered a normal behaviour for that particular user [1]. If the behaviour of supposedly the same user is significantly different we can speculate that perhaps it is a different user masquerading as the user whose profile is stored in our security system as a template.

In this work we tackle the intrusion detection using the behavioural biometrics approach [2]. The approach is based on bioinformatics’ Smith-Waterman algorithm that is generally used for the DNA sequence alignment [3]. We implemented a general Smith-Waterman algorithm with the matching score that can be any integer of fixed point decimal value. In the case of the standard bioinformatics application, only fixed values are applied and optimized.

A security attack usually possesses some kind of signature that identifies it and can be described as a Boolean relation called rule. The detection technique basically looks for one specific signature in the data flow. This signature often consists in one or more specific binary patterns found in a given file or network packet. Pattern matching is the most widespread attack detection technique [4]. It generally means matching text or binary sequences against known elements often referred to as signatures.

Since the Smith-Waterman algorithm can be significantly accelerated exploiting its inherent parallelism, we deployed a hardware implementation to reach Gigabit rates that are required in contemporary data processing [5]. As far as we know there is no published work on the hardware deployment in IDS of the above mentioned algorithm.

The deployment of hardware based systems offers the possibility to exploit much more parallelism than might be the case of software based systems. The consequence is that the alternative, highly optimized parallel algorithms then can be applied. Therefore hardware based solutions are probably the most promising practical approach for intrusion detection on high-speed backbone networks [5].

This work elaborates the implementation and the use of dedicated hardware in computer network security. The novel FPGA implementation provides new potential which can further lead to better performances and more reliable security systems. The results yield 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.

The target technology in this work is Field Programmable Gate Array (FPGA). This technology is suitable for a broad range of applications as current FPGAs contain much more configurable logic blocks (CLBs) than their predecessors. Some researchers suggest that FPGAs have become highly competitive with microprocessors in both peak and sustained performance [6]. Besides high computing performance, the current FPGAs also provide large amounts of on-chip and off-chip memory bandwidth to I/O-bound applications.

Therefore the user profiling is aimed in this work for prevention of the masquerading attack. This is a security attack in which an intruder assumes the identity of a legitimate user. In the detection of this attack, it is shown that using a bioinformatics algorithm as behavioural biometrics tool provides the best rates in terms of false positive and false negative alarms [7]. The Smith-Waterman bioinformatics algorithm was originally applied in gene alignment but efficient hardware implementations widen its use to other fields [8], [9].

The rest of the paper is organized as follows. Section 2 gives a background on user behavioural profiling to detect masquerading attacks. Section 3 explains the matching
algorithm. Section 4 describes the hardware implementation and its benefits in comparison to software implementation. Section 5 presents the corresponding results. Finally, the conclusions are drawn in Section 6.

2. User Behaviour Profiling

Masquerade attacks occur when an attacker assumes the identity of a legitimate user of computer system. It can happen when an intruder obtains a password of the legitimate user or when the user leaves his workstation unattended and without any sort of locking mechanism in place. This attack is not trivial to detect because the attacker appears to be a normal user with valid authority and privileges. The damage that can be done like stolen/destroyed documents, data, email, makes this kind of attack a very serious threat to computer and network infrastructure.

User profiling can help in detection of a masquerader as being behavioral biometrics tool that can make a contrast between the real legitimate user and the intruder. Namely, each user possesses its own specific behavior that can be formalized in a so called user signature. Comparing the user signature with the behavior data of logged user, the matching can differentiate a legal use and an intrusion in computer system. The user signature is usually formed on the basis of a sequence of user commands but it can also contain an additional data related to user style of keyboard typing or specific mouse movements that characterize the user. Also physiological biometric features can be combined that would certainly improve the exactness of identification but would have additional cost in computational complexity.

The user profiles i.e. the signatures should be updated frequently since the user behavior can change over time. Namely the temporal component of the profiling is particularly important. In a small period of time the user can react in a different manner depending on various circumstances while in long periods the user behavior can change fundamentally. Physiological biometrics characteristics like the fingerprint or eye recognition are more reliable and unvarying over time but in remote systems these features can be useless. Therefore the combination of physiological and behavioural biometrics would offer better detection results but due to increased complexity it is difficult to achieve it in real time on general purpose computers.

Researchers have developed and applied with more or less success several algorithms to detect anomalies in user behaviour. The difficulties are that an intruder may happen to have similar behaviour and/or the user’s change in activity is not captured strongly in the signature. In [7] various masquerade detection techniques like Bayesian, Markov model etc. were compared in terms of missed attacks as false negatives and false alarms as false positives. The best results exhibited the bioinformatics approach that uses slight modification of the Smith-Waterman algorithm, originally published for the purpose of gene alignment.

3. Matching Algorithm

The sequence alignment was originally developed for the comparison of genetic material like DNA. It has also been used as local alignment algorithm for other sequences, strings or in this case user signatures. It is a well-studied tool utilized to quantify and visualize similarity between two or more sequences.

The Smith-Waterman algorithm is a database search algorithm developed for use in bioinformatics [3]. It is based on a previous calculation model named Needelman-Wunsch [10]. The Smith-Waterman algorithm applies a dynamic programming technique. This technique seizes alignments of any length, at any location, in any of two input sequences, and determines whether an optimal alignment can be found in two input sequences. Based on these calculations, scores or weights are assigned to each character-to-character comparison. Positive scores are attributed for exact matches and substitutions, while negative score are applied in the case of insertions and deletions. At the end, the algorithm reports the highest scoring alignment.

Many heuristic methods like BLAST and FASTA were developed to perform the same task as an alternative to Smith Waterman algorithm because of its high complexity. But the original Smith Waterman algorithm is superior to them as being more sensitive technique that searches a much larger field [9].

Individual pair-wise comparisons between letters are quite time consuming. Instead of looking at an entire sequence at once, the Smith Waterman algorithm compares multi-length segments, looking for whichever segment maximizes the scoring measure.

The algorithm (1) itself is recursive in nature.

\[
H_{i,j} = \max \left\{ \begin{array}{l}
H_{i-1,j-1} + s(a_i, b_j), \\
\max_{k\neq i} \left\{ H_{i,k-1} + W_k \right\}, \\
\max_{l\neq j} \left\{ H_{i-1,l} + W_l \right\} \end{array} \right. \tag{1}
\]

In (1), \( a_i \) and \( b_j \) are units of sequences that are being compared on positions \( i \) and \( j \), respectively, \( H_{i,j} \) is matrix value on position with coordinates \( i, j \), respectively, and \( W_k \) is weight for deletion of \( k \) sequence units [3].

The potential of an algorithm as a comparison tool depends on a scoring scheme that is used to rank different alignments. The scoring function assigns positive scores to aligned characters that either match or are known to be similar. Negative scores are assigned to both aligned characters that are dissimilar and to characters that are aligned with gaps.

The aim is to find the greatest possible length that can be obtained from both sequences by deleting zero or more characters. Alternatively, this is a problem of aligning two strings in order to maximize the number of matching characters by inserting gaps into either string in order to shift the characters into matching alignment.
Some possible alignments of two sequences are illustrated in Figure 1.

![Alignment Examples](Image)

**Figure 1.** Some possible alignments of two sequences AABBBCCC and ABBC

For the values of the parameters \( W_k = 1 + k / 3 \), \( s(a_i, b_j) = 1 \) for \( a_i = b_j \), and \( s(a_i, b_j) = -1 / 3 \) when not \( a_i \neq b_j \), the algorithm would generate the comparison score matrix given in Figure 2. After matrix generation the best alignment can be selected by choosing the maximum value in the matrix and following the path it was generated by.

![Edit-Distance Matrix](Image)

**Figure 2.** Edit-distance matrix for the strings AABBBCCC and ABBC

It is straightforward 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.

Instead of comparing nucleotides, the concept can be used to align the sequences of user commands. In this case, the user signature corresponds to one sequence and the test block corresponds to another.

If the algorithm is able to align some subsequences extracted from those two arrays, scoring them adequately, and if the score is high enough it could mean that the legitimate user is logged on. If the alignment score is not sufficient, the potential intruder is logged and the session should be interrupted.

The original algorithm formula can be modified as the following formula (2).

\[
M_{ij} = \max \left[ M_{i-1,j-1} + s_{gij}, (M_{i,j-1} + g_{h}) (M_{i-1,j} + g_{h}) 0 \right]
\]

In (2), \( M_{ij} \) 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.

In this case it is possible to apply the following principle. We want to be able to insert gaps into the tested block to simulate the insertion of commands between characteristic groups of commands in the user’s signature. This requires to provide a slightly lesser penalty for gaps in the tested block. Matches should positively influence the score of an alignment, and should be chosen so that matches are preferred to gaps.

Using the above mentioned 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.

Mismatches are kept at a constant score of 0, as a blanket reward or penalty for any mismatch would unfairly favor certain alignments, and would not disallow concept drift [2]. The scoring system rewards the alignment of commands in the user segment but does not necessarily penalize the misalignment of large portions of the signature that do not necessarily align with a segment of the user’s commands.

The scores are taken from [2], but any other scoring can be applied in our implementation 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 the idea is 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. Semi-global Smith-Waterman comparison algorithm [2], is very good starting point as it has hit-rate of 75.8%, while false positive being 7.7%.

Although having the best detection results in the masquerading intrusion, the sequentially implemented algorithm is quite slow. The complexity of the algorithm is \( O(mn) \) where \( m \) and \( n \) are the lengths of the user signature and test block respectively. This is not a problem for relatively short sequences. But if the physiological biometrics data is added, the sequences can even reach lengths up to a million elements. The fastest general purpose computer can not accomplish this task in real time. The important fact is that there are no multiplications in the algorithm that would additionally slow down the calculation but the amount of data that needs to be processed is enormous. This circumstances provoke the need to study alternative ways to carry out the algorithm.

4. **Software vs. Hardware Implementation**

The main drawback of the Smith-Waterman algorithm is its slowness in sequential (software) implementation. Thus many heuristic methods have been developed e.g. FASTA and BLAST but they produce too many errors in detecting the masquerading intrusions.

To illustrate a software implemented Smith-Waterman algorithm we programmed one in ANSI C, where integers are used instead of the originally used floating point values. This
way it uses twice less memory than the original floating point algorithm. On AMD AthlonXP 2GHz clock frequency and 256KB cache memory based PC processor with 256MB of RAM, the average execution time needed to calculate one score-matrix element was $13.3 \mu s$ and the worst case execution time of the same action was $16.4 \mu s$. In order to generate one score-matrix element two characters i.e. two-byte info needs to be introduced that means that throughput parallelism cannot be applied on general purpose machines.

The only way to reduce the complexity of the algorithm is to introduce the parallelism in algorithm’s calculations. The parallelism cannot be applied on general purpose machines where the processor has only one arithmetic-logic unit. But hardware devices like FPGAs have capabilities to carry out the necessary parallel calculations and therefore reduce the complexity of the algorithm, and set free other hardware parts that were used in processing the user signature and the test block.

The computation of a weight matrix can be implemented in several different ways. The first way can be like in sequential machines. For calculating the current element $M_{ij}$, we need to have the diagonal element $M_{i-1,j-1}$, the upper element $M_{i,j-1}$ and finally the left element, $M_{i-1,j}$. Using these three as well as reward/penalty constants we can produce the result. We can create a hardware cell, a simple processing element, and optimize it for weight matrix calculation. One such processing element is shown in Figure 3.

![Figure 3. Bioinformatics processing element](image)

The computation of a weight matrix can be implemented in several different ways. The first way can be like in sequential machines. For calculating the current element $M_{ij}$, we need to have the diagonal element $M_{i-1,j-1}$, the upper element $M_{i,j-1}$ and finally the left element, $M_{i-1,j}$. Using these three as well as reward/penalty constants we can produce the result. We can create a hardware cell, a simple processing element, and optimize it for weight matrix calculation. One such processing element is shown in Figure 3.

Using these three as well as reward/penalty constants we can produce the result.

![Figure 3. Bioinformatics processing element](image)

But, it is still possible to reduce the time complexity from quadratic to linear without letting area to increase that much with an increase of the sequence length for a matrix size $m \times n$ using a systolic array of at least $\min\{m,n\}$ processing elements. Although the 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 the area that this hardware occupies. The same processing element calculates distances in the column in which it is placed (Fig. 4).

![Figure 4. Parallelized matrix calculation](image)

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

5. Results and Discussion

In this work, the general Smith-Waterman algorithm is implemented with the score that can be any integer of fixed point decimal value.

One of the most popular FPGA families, Xilinx Virtex 4LX200 was used for an algorithm implementation. In structures like FPGAs the best way to gain speed is to create highly optimized regular cells and then simply replicate them.

It was assumed that there are 256 used commands, which in operating systems like, UNIX and Linux are more than enough. We also assumed that the characteristic mouse movements and keyboard typing styles can be represented
with eight bits each, although their expansion would not dramatically decrease performance.

The circuit was implemented using Xilinx ISE 7.1i tool. The systolic cells were first implemented in VHDL and then optimized by manually placing their inner components and creating relational-placed macros (RPM). The RPMs allowed an efficient use of the FPGA and higher processing speed.

Both area and speed were improved by an approximate factor of a 100%. The area was reduced exploiting the regularity of the systolic array through the creation of cells that were able to fit more precisely within the overall area. The speed was doubled reaching clock frequencies of 200 MHz due to the reduction of routing lines within each systolic cell and among cells.

In a Xilinx Virtex 4 one cell takes 100 logic cells, what is 50 slices (or 25 CLBs). This means that more than 2000 processing elements can be implemented on this chip. On the 8-bit wide bus, the equivalent throughput is 250MB/s or 2Gb/s that is a number of orders more than for sequential implementation.

Self equipped devices based on microprocessor and FPGA combining the network support gives the possibility to implement this design independently from personal computer, and monitor the network independently and thus free the server totally from this activity and, what is also important, to be placed somewhere else.

The results are obtained from the simulation with partially manually generated user signatures as the input. Exactly, we have written a script to encode and precisely enumerate the set of the commands available on the host system. The commands are thus ordered by their alphabetic order and an ordinal number is assigned to each and every one of them. The encoding script was written to enumerate the 256 mostly used commands. This has led to the matrix construction by the comparison of the 8-bits wide command elements. The members of our investigation group have been asked to track and provide their own profiles, which they have in most of the cases done manually, and these entries were then processed by the enumeration script and used as the simulation data. This process, however, has to be completely automated before being used on a real system.

The described simulation input data was fed to the comparison engine. After the construction of the matrix the decision logic analyses the obtained values. If the maximum value encountered in the matrix was higher than 80% of the length of the user signature, the logic permits the further interaction of the user and the host. Similarly, the other scenarios can include different decisions based on the maximum similarity value recorded in the matrix. In continuation we need to study the options for the design of the loop closure mechanism to be used for the dynamic determination of the threshold value, aforementioned and fixed 80%.

Within the network environment, each workstation would need to have its own masquerade detecting device. To exploit the throughput capabilities of the proposed schema the device will have to be attached to the PCI Express bus with at least 8 lanes and resulting throughput of 8x250Mbits=2Gbps. That would than allow an automated mechanism, sketched in the paragraph above to be used as continuous feeding engine of the detection system.

The server host, whatsoever, requires a different approach. Besides processing the data the same way the workstations do, locally it needs to consider the remote user actions and profiles derived from those. The monitoring of the secure sessions and encoding of the information would be unavoidable for such a case. Furthermore, the issues such as a variable latency that network provides and the influence of this phenomena to the user profiles generation process, and later comparison and decision processes needs to be thoroughly studied before having a server based system.

Yet, from the server perspective we note the importance of throughput performance the proposed “de-masquerading” schema is capable of delivering. We expect to undertake both the simulation based and practical tests of the system and obtain an objective efficiency measure.

6. Conclusion

We designed special intrusion detection hardware based on behavioural biometrics i.e. user profiling. The pattern recognition is carried out using the bioinformatics Smith-Waterman algorithm that exhibits high performances in detecting masquerading attack. The slowness of the algorithm is significantly reduced due to parallel hardware implementation.

We designed hardware architecture that can deploy any sequence alignment scoring in contrast to bioinformatics case where only fixed values are applied and optimized. The design was deployed in Virtex 4 FPGA and, as far as we know, there is no published hardware IDS implementations that uses this bioinformatics approach.

Each hardware unit reaches Gigabit processing rates thus satisfying contemporary data flow requirements and for several orders overcoming sequential software solutions. Due to optimized area design, the units can be massively replicated on FPGA devices and achieve a significant throughput increase.

Acknowledgment

The work was supported by Ministry of Science of Spain through the project TEC2009-14595-C02-01 and EC FP6 project IST 027095.

References


